

# Funkční bloky systému REXYGEN Referenční příručka

REX Controls s.r.o.

Verze 3.0.1  
1.3.2024  
Plzeň



# Obsah

<b>1</b>	<b>Úvod</b>	<b>17</b>
1.1	Jak číst tuto příručku . . . . .	17
1.2	Formát popisu funkčních bloků . . . . .	22
1.3	Konvence pojmenování proměnných, bloků a subsystémů . . . . .	23
1.4	Kvalita signálu používaná v OPC . . . . .	25
<b>2</b>	<b>EXEC – Konfigurace exekutivy reálného času</b>	<b>27</b>
	ALARMS – Hromadná definice alarmů . . . . .	29
	ARC – Archiv systému REXYGEN . . . . .	32
	EXEC – Exekutiva reálného času . . . . .	34
	HMI – * Konfigurace vizualizace . . . . .	37
	INFO – * Dodatečné informace o projektu . . . . .	38
	IODRV – Vstupně-výstupní ovladač systému REXYGEN . . . . .	39
	IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem . . . . .	41
	LPBRK – Rozpojení zpětné vazby . . . . .	42
	MODULE – Rozšiřující modul systému REXYGEN . . . . .	43
	OSCALL – Volání funkcí operačního systému . . . . .	44
	PROJECT – * Další nastavení projektu . . . . .	45
	QTASK – Rychlá úloha řídicího systému REXYGEN . . . . .	46
	SLEEP – Časovací blok pro Simulink . . . . .	47
	SRTF – Blok pro nastavování příznaků běhu . . . . .	48
	STATELOAD – Načtení stavů a parametrů více bloků . . . . .	50
	STATESAVE – Uložení stavů a parametrů více bloků . . . . .	52
	SYSEVENT – Čtení systémového logu . . . . .	54
	SYSLOG – Zápis do systémového logu . . . . .	56
	TASK – Standardní úloha řídicího systému REXYGEN . . . . .	57
	TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami . . . . .	59
	WWW – * Obsah pro interní webserver . . . . .	61
<b>3</b>	<b>INOUT – Bloky vstupů a výstupů</b>	<b>63</b>
	Display – * Zobrazení vstupní hodnoty . . . . .	64
	From, INSTD – Připojení signálu nebo vstupní signál . . . . .	65
	Goto, OUTSTD – Zdroj signálu nebo výstupní signál . . . . .	67

GotoTagVisibility – Viditelnost zdroje signálu . . . . .	69
Inport, Outport – Vstupní a výstupní port . . . . .	70
SubSystem – Subsystém . . . . .	72
INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů . . . . .	74
OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů . . . . .	76
OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací . . . . .	78
OUTRSTD – Výstupní signál s verifikací hodnoty . . . . .	79
QFC – Kódování příznaků kvality signálu . . . . .	80
QFD – Dekódování příznaků kvality signálu . . . . .	81
VIN – Ověření kvality vstupního signálu . . . . .	82
VOUT – Nastavení kvality výstupního signálu . . . . .	83
<b>4 MATH – Matematické bloky . . . . .</b>	<b>85</b>
ABS – Absolutní hodnota . . . . .	87
ADD – Součet dvou signálů . . . . .	88
ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů . . . . .	89
CNB – Booleovská (logická) konstanta . . . . .	91
CNE – Předdefinovaná konstanta . . . . .	92
CNI – Celočíselná konstanta . . . . .	93
CNR – Reálná konstanta . . . . .	94
DIF – Blok diference . . . . .	95
DIV – Dělení dvou signálů . . . . .	96
EAS – Rozšířené sčítání a odečítání . . . . .	97
EMD – Rozšířené násobení a dělení . . . . .	98
FNX – Výpočet hodnoty funkce jedné proměnné . . . . .	100
FNXY – Výpočet hodnoty funkce dvou proměnných . . . . .	102
GAIN – Násobení konstantou . . . . .	104
GRADS – Gradientní optimalizace . . . . .	105
IADD – Celočíselné sčítání . . . . .	107
IDIV – Celočíselné dělení . . . . .	109
IMOD – Zbytek po celočíselném dělení . . . . .	110
IMUL – Celočíselné násobení . . . . .	111
ISUB – Celočíselné odčítání . . . . .	113
LIN – Lineární interpolace . . . . .	115
MUL – Násobení dvou signálů . . . . .	116
POL – Vyhodnocení polynomu . . . . .	117
REC – Převrácená hodnota . . . . .	118
REL – Relační operace dvou signálů . . . . .	119
RTOI – Konverze reálného čísla na celé číslo . . . . .	120
SQR – Druhá mocnina . . . . .	122
SQRT – Druhá odmocnina . . . . .	123
SUB – Odčítání dvou signálů . . . . .	124
UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem . . . . .	125

<b>5</b>	<b>ANALOG – Zpracování analogových signálů</b>	<b>127</b>
	ABSROT – Víceotáčkové rozšíření čidla polohy . . . . .	129
	ASW – Přepínač s automatickou volbou vstupu . . . . .	131
	AVG – Filtr: vlečný průměr . . . . .	133
	AVS – Rozběhová jednotka . . . . .	134
	AVSI – Hladká interpolace trajektorie . . . . .	135
	BPF – Filtr: pásmová propust' . . . . .	138
	CMP – Komparátor s hysterezí . . . . .	139
	CNDR – Kompenzátor složité nelinearity . . . . .	140
	DEL – Dopravní zpoždění s inicializací . . . . .	142
	DELM – Dopravní zpoždění . . . . .	143
	DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků . . . . .	144
	EVAR – Vlečná střední hodnota a směrodatná odchylka . . . . .	145
	INTE – Řízený integrátor . . . . .	146
	KDER – Derivace a filtrace vstupního signálu . . . . .	148
	LPF – Filtr: dolní propust' . . . . .	150
	MINMAX – Vlečné minimum a maximum . . . . .	152
	NSCL – Kompenzátor jednoduché nelinearity . . . . .	153
	OSD – Jednokrokové zpoždění . . . . .	154
	RDFT – Vlečná diskretní Fourierova transformace . . . . .	155
	RLIM – Omezovač strmosti . . . . .	157
	S10F2 – Výběr jednoho ze dvou analogových vstupů . . . . .	158
	SAI – Zabezpečený analogový vstup . . . . .	161
	SEL – Selektor analogového signálu . . . . .	164
	SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu . . . . .	166
	SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot . . . . .	168
	SHLD – Vzorkovač (sample and hold) . . . . .	170
	SINT – Jednoduchý integrátor . . . . .	172
	SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů . . . . .	173
	SSW – Jednoduchý přepínač . . . . .	175
	SWR – Přepínač s rampovou funkcí . . . . .	176
	VDEL – Dopravní zpoždění s proměnnou délkou . . . . .	177
	ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací . . . . .	178
<b>6</b>	<b>GEN – Generátory signálů</b>	<b>183</b>
	ANLS – Řízený generátor po částech lineární funkce . . . . .	184
	BINS – Řízený generátor binární posloupnosti . . . . .	186
	BIS – Generátor binární posloupnosti . . . . .	188
	BISR – * Generátor binární posloupnosti s možností pozastavení . . . . .	189
	MP – Ručně generovaný pulz . . . . .	190
	PRBS – Pseudonáhodná binární posloupnost . . . . .	191
	SG, SGI – Řízený generátor signálu . . . . .	193

<b>7 REG – Bloky pro regulaci</b>	<b>195</b>
ARLY – Relé s předstihem . . . . .	197
FLCU – Fuzzy regulátor . . . . .	198
FRID – * Identifikace frekvenční charakteristiky . . . . .	200
I3PM – Identifikace modelu se třemi parametry . . . . .	202
LC – Derivační kompenzátor . . . . .	204
LLC – Integračně-derivační kompenzátor . . . . .	205
LPI – Hodnocení kvality regulační smyčky . . . . .	206
MCU – Jednotka pro ruční zadávání . . . . .	208
PIDAT – PID regulátor s reléovým autotunerem . . . . .	210
PIDE – PID regulátor se statikou . . . . .	213
PIDGS – PID regulátor s přepínáním sad parametrů . . . . .	215
PIDMA – PID regulátor s momentovým autotunerem . . . . .	217
PIDU – PID regulátor . . . . .	223
PIDUI – PID regulátor s parametry na vstupech . . . . .	226
POUT – Pulzní výstup . . . . .	228
PRGM – Programátor . . . . .	229
PSMPC – Prediktivní „pulse-step“ regulátor . . . . .	231
PWM – Blok šířkové modulace . . . . .	235
RLY – Relé s hysterezí . . . . .	237
SAT – Saturace výstupu s proměnnými mezemi . . . . .	238
SC2FA – Stavový regulátor systému 2. řádu s autotunerem . . . . .	240
SCU – Krokový regulátor s polohovou zpětnou vazbou . . . . .	247
SCUV – Krokový regulátor s rychlostním výstupem . . . . .	250
SELU – Selektor aktivního regulátoru . . . . .	254
SMHCC – Regulátor pro procesy s topením a chlazením . . . . .	255
SMHCCA – * Regulátor pro procesy s topením a chlazením s autotunerem . . . . .	258
SWU – Přepínač vstupu pro vysledování . . . . .	260
TSE – Třístavový prvek . . . . .	261
<b>8 LOGIC – Logické řízení</b>	<b>263</b>
AND – Logický součin dvou signálů . . . . .	264
ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů . . . . .	265
ATMT – Automat pro sekvenční řízení . . . . .	267
BDOCT, BDHEXD – Bitové demultiplexery . . . . .	270
BITOP – Bitová operace dvou celočíselných signálů . . . . .	272
BMOCT, BMHEXD – Bitové multiplexery . . . . .	273
COUNT – Řízený čítač . . . . .	275
EATMT – Rozšířený automat pro sekvenční řízení . . . . .	277
EDGE – Detekce hrany logického signálu . . . . .	280
EQ – Shodnost dvou signálů . . . . .	281
INTSM – Bitový posun a maska nad celým číslem . . . . .	282
ISSW – Jednoduchý přepínač celočíselných signálů . . . . .	283
ITOI – Transformace celých a binárních čísel . . . . .	284

NOT – Logická negace . . . . .	286
OR – Logický součet dvou signálů . . . . .	287
ORQUAD, OROCT, ORHEXD – Logický součet více signálů . . . . .	288
RS – Klopný obvod . . . . .	290
SR – Klopný obvod . . . . .	291
TIMER – Vícefunkční časovač . . . . .	292
<b>9 TIME – Bloky pro práci s časem . . . . .</b>	<b>295</b>
DATE – Aktuální datum . . . . .	296
DATETIME – Čtení, nastavování a konverze času . . . . .	297
TC – Řízení časovače . . . . .	299
TIME – Aktuální čas . . . . .	301
WSCH – Týdenní časovač . . . . .	302
<b>10 ARC – Archivace dat . . . . .</b>	<b>305</b>
10.1 Funkce archivačního subsystému . . . . .	306
10.2 Generování alarmů u a událostí . . . . .	307
ALB, ALBI – Alarmy pro logickou hodnotu . . . . .	307
ALM, ALMI – Aktivace alarmu . . . . .	309
ALN, ALNI – Alarmy pro číselnou hodnotu . . . . .	310
ARS – Uložení hodnoty do archivu . . . . .	313
10.3 Záznam trendů . . . . .	315
ACD – Archivní komprese s použitím delta kritéria . . . . .	315
TRND – Záznam trendů v reálném čase . . . . .	317
TRNDV – Záznam trendů v reálném čase (vektorová forma) . . . . .	320
10.4 Správa archivů . . . . .	322
AFLUSH – Vynucené zapsání archivu . . . . .	322
ACLEAR – * Vynucené smazání archivu . . . . .	323
<b>11 STRING – Bloky pro práci s řetězci . . . . .</b>	<b>325</b>
CNS – * Textová konstanta . . . . .	326
CONCAT – * Spojení stringů (podle vzoru) . . . . .	327
FIND – * Nalezení textu . . . . .	328
ITOS – Konverze celého čísla na text . . . . .	329
LEN – * Délka textu . . . . .	330
MID – * Výřez textu . . . . .	331
PJROCT – Získání číselných hodnot z textu ve formátu JSON . . . . .	332
PJSOCT – Získání textových hodnot z textu ve formátu JSON . . . . .	334
PJSEXOCT – Získání textových hodnot z textu ve formátu JSON . . . . .	336
REGEXP – * Regular expression parser . . . . .	337
REPLACE – * Náhrada textu . . . . .	338
RTOS – Konverze čísla na text . . . . .	339
SELSOCT – * Výběr textu z několika vstupů . . . . .	340
STOR – * Koverze textu na číslo . . . . .	341

TRIM – * Odstranění mezer na začátku a konci textu . . . . .	342
<b>12 PARAM – Bloky pro manipulaci s parametry</b>	<b>343</b>
GETPA – Blok pro vzdálené získání vektorového parametru . . . . .	344
GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru . . . . .	346
GETPS – * Blok pro vzdálené získání parametru typu string . . . . .	348
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu . . . . .	349
PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu . . . . .	350
PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu . . . . .	351
PARS – * Blok s parametrem typu string nastavitelným ze vstupu . . . . .	353
SETPA – Blok pro vzdálené nastavování vektorového parametru . . . . .	354
SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru . . . . .	356
SETPS – * Blok pro vzdálené nastavování parametru typu string . . . . .	358
SGSLP – Nastavování, čtení, ukládání a načítání parametrů . . . . .	359
SILO – Uložení vstupního signálu, načtení výstupního signálu . . . . .	363
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce . . . . .	365
<b>13 MODEL – Simulace dynamických systémů</b>	<b>367</b>
CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním . . . . .	369
CSSM – Stavový model spojitého lineárního systému . . . . .	372
DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním . . . . .	374
DFIR – * Filtr s konečnou impulzní odezvou . . . . .	376
DSSM – Stavový model diskrétního lineárního systému . . . . .	377
EKF – Rozšířený (nelineární) Kalmanův filtr . . . . .	379
FOPDT – Model systému 1. řádu s dopravním zpožděním . . . . .	382
IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry	383
IPEN2pu, IPEN3pu – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry . . . . .	386
MDL – Model procesu . . . . .	388
MDLI – Model procesu s proměnnými parametry . . . . .	389
MVD – Motorizovaný pohon ventilu . . . . .	390
NSSM – Nelineární stavový model . . . . .	391
SOPDT – Model systému 2. řádu s dopravním zpožděním . . . . .	394
<b>14 MATRIX – Bloky pro maticové a vektorové operace</b>	<b>397</b>
14.1 Poznámky k implementaci: . . . . .	399
CNA – * Konstantní pole (vektor/matice) . . . . .	400
MB_DASUM – * Součet absolutních hodnot . . . . .	401
MB_DAXPY – * Provádí $y := a*x + y$ pro vektory $x, y$ . . . . .	402
MB_DCOPY – * Kopíruje vektor $x$ do vektoru $y$ . . . . .	403
MB_DDOT – * Skalární součin dvou vektorů . . . . .	404



MB_DGEMM – * Provádí $C := \alpha \cdot \text{op}(A) \cdot \text{op}(B) + \beta \cdot C$ , where $\text{op}(X)$ = $X$ or $\text{op}(X) = X^T$ . . . . .	405
MB_DGEMV – * Provádí $y := \alpha \cdot A \cdot x + \beta \cdot y$ or $y := \alpha \cdot A^T \cdot x +$ $\beta \cdot y$ . . . . .	406
MB_DGER – * Provádí $A := \alpha \cdot x \cdot y^T + A$ . . . . .	407
MB_DNRM2 – * Eukleidovská norma vektoru . . . . .	408
MB_DRROT – * Rovinná rotace vektoru . . . . .	409
MB_DSCAL – * Násobení vektoru konstantou . . . . .	410
MB_DSWAP – * Záměna dvou vektorů . . . . .	411
MB_DTRMM – * Provádí $B := \alpha \cdot \text{op}(A) \cdot B$ or $B := \alpha \cdot B \cdot \text{op}(A)$ , where $\text{op}(X) = X$ or $\text{op}(X) = X^T$ pro trojúhelníkovou matici $A$ . . . . .	412
MB_DTRMV – * Provádí $x := A \cdot x$ or $x := A^T \cdot x$ pro trojúhelníkovou matici $A$ . . . . .	413
MB_DTRSV – * Řeší jednu ze soustav rovnic $A \cdot x = B$ nebo $A^T \cdot x = B$ pro trojúhelníkovou matici $A$ . . . . .	414
ML_DGEBAK – * Zpětná transformace k ML_DGEBAL levých nebo pravých vlastních vektorů . . . . .	415
ML_DGEBAL – * Vyvážení obecné reálné matice . . . . .	416
ML_DGEBRD – * Redukce obecné reálné matice do bidiagonální formy po- mocí ortogonální transformace . . . . .	417
ML_DGECON – * Odhad převrácené hodnoty čísla podmíněnosti obecné re- álné matice . . . . .	418
ML_DGEES – * Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů . . . . .	419
ML_DGEEV – * Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů . . . . .	420
ML_DGEHRD – * Redukce reálné obecné matice $A$ na horní Hessenbergovu formu . . . . .	421
ML_DGELQF – * Výpočet LQ factorizace reálné matice $A$ s rozměry $M \times N$ . . . . .	422
ML_DGELSD – * Výpočet řešení s minimální normou reálné lineární úlohy nejmenších čtverců . . . . .	423
ML_DGEQRF – * Výpočet QR factorizace reálné matice $A$ s rozměry $M \times N$ . . . . .	424
ML_DGESDD – * Výpočet singulární dekompozice (SVD) reálné matice $A$ s rozměry $M \times N$ . . . . .	425
ML_DLACPY – * Kopíruje celou nebo část matice do jiné matice . . . . .	426
ML_DLANGE – * Výpočet některé z maticových norem obecné matice . . . . .	427
ML_DLASET – * Inicializuje mimodiagonální a diagonální prvky matice na zadané hodnoty . . . . .	428
ML_DTRSYL – * Řešení reálné Sylvesterovy rovnice pro kvazitrojúhelníkové matice $A$ a $B$ . . . . .	429
MX_AT – * Hodnota prvku matice/vektoru . . . . .	430
MX_ATSET – * Nastavení hodnoty prvku matice/vektoru . . . . .	431
MX_CNADD – * Přičte skalár ke každému prvku matice/vektoru . . . . .	432
MX_CNMUL – * Vynásobí matici/vektor skalárem . . . . .	433

MX_CTODPA – * Discretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací . . . . .	434
MX_DIM – * Dimenze matice/vektoru . . . . .	435
MX_DIMSET – * Nastavení dimenze matice/vektoru . . . . .	436
MX_DSAGET – * Uložení submatice A do matice B . . . . .	437
MX_DSAREF – * Nastavení odkazu na submatici A do matice B . . . . .	438
MX_DSASET – * Uložení matice A do submatice v B . . . . .	439
MX_DTRNSP – * Transpozice obecné matice: $B := \alpha * A^T$ . . . . .	440
MX_DTRNSQ – * Transpozice čtvercové matice na místě: $A := \alpha * A^T$ . . . . .	441
MX_FILL – * Vyplnění reálné matice/vektoru . . . . .	442
MX_MAT – * Blok pro uložení dat matice . . . . .	443
MX_RAND – * Náhodně vygenerovaná matice nebo vektor . . . . .	444
MX_REFCOPY – * Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů . . . . .	445
MX_SLFS – Ukládání a čtení matice/vektoru do souboru nebo textového retězce . . . . .	446
MX_VEC – * Blok pro uložení dat vektoru . . . . .	449
MX_WRITE – * Výpis matice/vektoru do konzole/systemého logu . . . . .	450
RTOV – Vektorový multiplexer . . . . .	451
SWVMR – * Přepínač vektorového/maticevého/odkazovacího signálu . . . . .	453
VTOR – * Vektorový demultiplexer . . . . .	454
<b>15 OPTIM – Bloky pro optimalizaci</b>	<b>455</b>
QCEDPOPT – * Optimalizace tzv. Economic Dispatch problému . . . . .	456
QP_MPC2QP – * Převod úlohy prediktivního řízení na kvadratické programování . . . . .	458
QP_OASES – * Kvadratické programování pomocí metody aktivní množiny . . . . .	460
QP_UPDATE – * Aktualizace matic/vektorů kvadratického programování . . . . .	463
<b>16 SPEC – Speciální bloky</b>	<b>467</b>
EPC – Blok pro spouštění externích programů . . . . .	468
HTTP – * Blok pro generování požadavků HTTP GET a POST (zastaralý) . . . . .	471
HTTP2 – * Blok pro generování HTTP požadavků . . . . .	473
RDC – Komunikační blok . . . . .	475
REXLANG – Volně programovatelný blok . . . . .	480
SMTP – * Blok pro odesílání e-mailových oznámení přes SMTP . . . . .	501
STEAM – Přepočet vlastností páry . . . . .	503
UART – * Komunikační blok pro UART . . . . .	505
<b>17 LANG – Speciální bloky</b>	<b>507</b>
PYTHON – Volně programovatelný blok v jazyce Python . . . . .	508

<b>18 DSP – Zpracování speciálních signálů</b>	<b>515</b>
BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu	516
BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typu . . .	518
BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu	520
BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu . .	521
BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu . . . . .	522
FFT – Rychlá Fourierova transformace . . . . .	523
MOSS – Přesný senzor pohybu . . . . .	525
PSD – Výkonová spektrální hustota . . . . .	527
<b>19 MQTTDrv – Komunikace přes MQTT protokol</b>	<b>533</b>
MqttPublish – Odeslání zprávy protokolem MQTT . . . . .	534
MqttSubscribe – Odběr zpráv z MQTT topic . . . . .	536
<b>20 MC_SINGLE – Řízení pohybu v jedné ose</b>	<b>539</b>
MCP_AccelerationProfile – * Generování trajektorie (zrychlení) . . . . .	542
MCP_Halt – * Zastavení pohybu (přerušitelné) . . . . .	544
MCP_HaltSuperimposed – * Zastavení pohybu (přídavné a přerušitelné) . .	545
MCP_Home – * Nalezení výchozí polohy . . . . .	546
MCP_MoveAbsolute – * Pohyb do pozice (absolutní souřadnice) . . . . .	548
MCP_MoveAdditive – * Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	550
MCP_MoveContinuousAbsolute – * Pohyb do pozice (absolutní souřadnice)	552
MCP_MoveContinuousRelative – * Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	554
MCP_MoveRelative – * Kartézské transformace . . . . .	556
MCP_MoveSuperimposed – * Pohyb do pozice (přídavný pohyb) . . . . .	558
MCP_MoveVelocity – * Pohyb konstantní rychlostí . . . . .	559
MCP_PositionProfile – * Generování trajektorie (poloha) . . . . .	561
MCP_SetOverride – * Nastavení násobivých faktorů na ose . . . . .	563
MCP_Stop – * Zastavení pohybu . . . . .	564
MCP_TorqueControl – * Řízení síly/momentu . . . . .	565
MCP_VelocityProfile – * Generování trajektorie (rychlost) . . . . .	567
MC_AccelerationProfile, MCP_AccelerationProfile – Generování tra- jektorie (zrychlení) . . . . .	569
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné) . . . . .	573
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (pří- davné a přerušitelné) . . . . .	575
MC_Home, MCP_Home – Nalezení výchozí polohy . . . . .	576
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní sou- řadnice) . . . . .	578
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	581

MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	584
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	588
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění) . . . . .	592
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb) . . . . .	595
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí . . . . .	598
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha) . . . . .	601
MC_Power – Aktivace osy . . . . .	605
MC_ReadActualPosition – Skutečná poloha osy . . . . .	606
MC_ReadAxisError – Chyba osy . . . . .	607
MC_ReadBoolParameter – Čtení parametru (logická hodnota) . . . . .	608
MC_ReadParameter – Čtení parametru (číselná hodnota) . . . . .	609
MC_ReadStatus – Stav osy . . . . .	611
MC_Reset – Nulování chyb osy . . . . .	613
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose . . . . .	614
MC_Stop, MCP_Stop – Zastavení pohybu . . . . .	616
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu . . . . .	618
MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost) . . . . .	621
MC_WriteBoolParameter – Nastavení parametru (logická hodnota) . . . . .	625
MC_WriteParameter – Nastavení parametru (číselná hodnota) . . . . .	626
RM_Axis – Osa pro řízení pohybu . . . . .	627
RM_AxisOut – Výstupní blok osy . . . . .	634
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení) . . . . .	635
RM_HomeOffset – * Nastavení počáteční polohy pomocí offsetu . . . . .	640
RM_Track – Sledování a krokování . . . . .	641
<b>21 MC_MULTI – Řízení pohybu více os</b> . . . . .	<b>643</b>
MCP_CamIn – * Zapnutí vačky . . . . .	645
MCP_CamTableSelect – Definice vačky . . . . .	647
MCP_CombineAxes – * Kombinace pohybu dvou os do třetí . . . . .	649
MCP_GearIn – * Zapnutí konstantního převodového poměru . . . . .	651
MCP_GearInPos – * Zapnutí konstantního převodového poměru v zadané pozici . . . . .	653
MCP_PhasingAbsolute – * Vytvoření fázového posunu (absolutní souřadnice) . . . . .	655
MCP_PhasingRelative – * Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	657
MC_CamIn, MCP_CamIn – Zapnutí vačky . . . . .	659
MC_CamOut – Vypnutí vačky . . . . .	663
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí . . . . .	665

MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru . . .	668
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici . . . . .	671
MC_GearOut – Vypnutí konstantního převodového poměru . . . . .	676
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice) . . . . .	678
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	681
<b>22 MC_COORD – Koordinované řízení pohybu</b>	<b>683</b>
MCP_GroupHalt – * Zastavení koordinovaného pohybu (přerušitelné) . . .	687
MCP_GroupInterrupt – * Přerušení pohybu skupiny os . . . . .	688
MCP_GroupSetOverride – * Nastavení násobivých faktorů na osách ve skupině . . . . .	689
MCP_GroupSetPosition – * Nastavení polohového offsetu skupiny os . . .	690
MCP_GroupStop – * Zastavení koordinovaného pohybu . . . . .	691
MCP_MoveCircularAbsolute – * Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	692
MCP_MoveCircularRelative – * Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	694
MCP_MoveDirectAbsolute – * Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	696
MCP_MoveDirectRelative – * Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	698
MCP_MoveLinearAbsolute – * Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	700
MCP_MoveLinearRelative – * Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	702
MCP_MovePath – * Generování obecné trajektorie v prostoru . . . . .	704
MCP_MovePath_PH – * Generování obecné trajektorie v prostoru PH . . . .	706
MCP_SetCartesianTransform – * Kartézské transformace . . . . .	708
MCP_SetKinTransform_Arm – * Kinematické transformace robot ARM . . .	709
MCP_SetKinTransform_UR – * Kinematická transformace pro UR robot . .	711
MC_AddAxisToGroup – Přidání osy do skupiny os . . . . .	713
MC_GroupContinue – Pokračování v přerušeném pohybu . . . . .	714
MC_GroupDisable – Převedení skupiny do stavu GroupDisabled . . . . .	715
MC_GroupEnable – Převedení skupiny do stavu GroupStandby . . . . .	716
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné) . . . . .	717
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušení pohybu skupiny os	723
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os . . . .	724
MC_GroupReadActualPosition – Aktuální poloha skupiny os . . . . .	725
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os . . . . .	726
MC_GroupReadError – Chyby ve skupině os . . . . .	727

MC_GroupReadStatus – Stav skupin os . . . . .	728
MC_GroupReset – Nulování chyb os ve skupině . . . . .	730
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině . . . . .	731
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os . . . . .	734
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu . . . . .	736
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	739
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	744
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	749
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	753
MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	757
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	761
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru . . . . .	765
MC_MovePath_PH – * Generování obecné trajektorie v prostoru PH . . . . .	768
MC_ReadCartesianTransform – Přečtení použité kartézské transformace . . . . .	770
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace . . . . .	772
MC_UngroupAllAxes – Odebrání všech os ze skupiny . . . . .	775
RM_AxesGroup – Skupina os pro koordinované řízení pohybu . . . . .	776
RM_Feed – MC „krmič“ . . . . .	781
RM_Gcode – CNC řízení pohybu . . . . .	783
RM_GroupTrack – * Sledování polohy/rychlosti . . . . .	786
<b>23 CanDrv – Komunikace po sběrnici CAN</b>	<b>789</b>
CanItem – Další přijatá zpráva sběrnice CAN . . . . .	790
CanRecv – Přijetí zprávy sběrnice CAN . . . . .	791
CanSend – Odeslání zprávy na sběrnici CAN . . . . .	793
<b>24 OpcUaDrv – Komunikace pomocí OPC UA</b>	<b>795</b>
OpcUaReadValue – Čtení hodnoty protokolem OPC UA . . . . .	796
OpcUaServerValue – Vystavení hodnoty v podobě OPC UA uzlu . . . . .	798
OpcUaWriteValue – Zápis hodnoty protokolem OPC UA . . . . .	800
<b>A Typy licencí</b>	<b>803</b>
<b>B Seznam funkčních bloků a jejich licencování</b>	<b>805</b>

<i>OBSAH</i>	15
<b>C Chybové kódy systému REXYGEN</b>	<b>819</b>
<b>D Speciální signály systému REXYGEN</b>	<b>825</b>
<b>Literatura</b>	<b>827</b>
<b>Rejstřík</b>	<b>829</b>

*Poznámka:* U bloků označených \* je k dispozici pouze částečná dokumentace. Kompletní dokumentace může být dostupná v ostatních jazykových mutacích manuálu.





# Kapitola 1

## Úvod

Příručka „Funkční bloky systému REXYGEN“ je, jak už její název napovídá, referenční příručkou knihovny RexLib funkčních bloků řídicího systému REXYGEN. Kromě referenčního popisu jednotlivých tříd, popisuje (referenčním způsobem) všechny subsystémy řídicího systému REXYGEN.

### 1.1 Jak číst tuto příručku

Standardně dodávaná rozsáhlá knihovna funkčních bloků RexLib řídicího systému REXYGEN je rozdělena do menších skupin logicky příbuzných bloků, tzv. *kategorií* (podknihoven). Každá kategorie je popisována v samostatné kapitole, obsahující nejprve obecné vlastnosti celé kategorie a jejích funkčních bloků, následované postupně popisem všech funkčních bloků dané kategorie.

Jednotlivé kapitoly příručky obsahují:

#### 1 Úvod

Tato úvodní kapitola, seznamující s uspořádáním příručky a uvádějící formát (konvenci) popisu jednotlivých funkčních bloků.

#### 2 EXEC – Konfigurace exekutivy reálného času

Knihovna EXEC je nezbytná pro nastavení exekutivy v reálném čase v systému REXYGEN a zahrnuje klíčové bloky jako **EXEC**, **TASK**, **QTASK** a **HMI**. Tyto bloky jsou fundamentální pro správu provádění úkolů, určování priority procesů a pro interakci s rozhraními pro uživatele, čímž podstatně přispívají k efektivitě a ovladatelnosti aplikací v ekosystému REXYGEN.

#### 3 INOUT – Bloky vstupů a výstupů

Knihovna INOUT slouží jako klíčové rozhraní v systému REXYGEN, umožňuje plynulou interakci s ovladači vstupů/výstupů. Je navržena pro efektivní simultánní zpracování signálů, což je zásadní pro úlohy rychlého řízení. Tato knihovna zjednodušuje propojení mezi řídicími algoritmy a hardwarem, zajišťuje minimální

latenci. Zároveň poskytuje pokročilé funkce, jako jsou virtuální propojení (vlajky) signálů pro zvýšenou přehlednost schémat a flexibilitu podsystémů.

#### 4 MATH – Matematické bloky

Knihovna MATH nabízí ucelenou sbírku matematických operací a funkcí. Obsahuje základní aritmetické bloky jako **ADD**, **SUB**, **MUL** a **DIV** pro standardní výpočty a specializovanější bloky, jako je **ABS** pro absolutní hodnoty, **SQRT** pro odmocniny a **SQR** pro odmocnění. Pokročilé funkce poskytují bloky jako **LIN** pro lineární transformace, **POL** pro vyhodnocení polynomů a **FNX**, **FNXY** pro přizpůsobitelné matematické funkce. Knihovna také nabízí operace specifické pro celá čísla prostřednictvím bloků jako **IADD**, **IMUL**, **IDIV** a **IMOD**.

#### 5 ANALOG – Zpracování analogových signálů

Knihovna představuje univerzální škálu funkčních bloků určených pro aplikace řízení a zpracování signálů. Obsahuje bloky jako **ASW**, **AVG**, **BPF** a **DEL**, které nabízejí funkce od manipulace se signálem a průměrování po filtrování a složité podmíněné operace, čímž pokrývají široké spektrum požadavků a scénářů.

#### 6 GEN – Generátory signálů

Knihovna GEN je specializována na generování signálů. Obsahuje bloky jako **ANLS** pro generování po částech lineární funkce času nebo generátory binárních sekvencí **BINS**, **BIS**, **BISR**. Knihovna také nabízí **MP** pro manuální generování pulsních signálů, **PRBS** pro generování pseudonáhodných binárních sekvencí a **SG** pro generování periodických signálů. Tato knihovna poskytuje zásadní nástroje pro vytváření a manipulaci s různými typy signálů.

#### 7 REG – Bloky pro regulaci

Bloky pro regulaci tvoří nejrozsáhlejší podknihovnu knihovny RexLib a zahrnují bloky od jednoduchých dynamických kompenzátorů, přes bloky pro přepínání regulačních struktur, bloky pro přizpůsobení výstupů akčním členům (krokové regulátory, šířková modulace) až po několik verzí PID (P, I, PI, PD a PID) regulátorů. Mezi regulátory jsou např. blok **PIDGS**, umožňující za běhu přepínat několik sad parametrů (tzv. *gain scheduling*), **PIDMA** s vestavěným *momentovým autotunerem*, blok **PIDAT** s vestavěným reléovým autotunerem nebo blok fuzzy regulátoru **FLCU**, a další.

#### 8 LOGIC – Logické řízení

Knihovna LOGIC zahrnuje řadu bloků pro provádění logických a sekvenčních operací. Zahrnuje základní booleovské bloky jako **AND**, **OR**, **NOT** pro základní logické operace a pokročilé bloky jako **ATMT** pro konečné automaty. Bloky jako **COUNT** a **TIMER** rozšiřují funkčnost na obousměrné počítání pulsů a operace založené na čase. Další prvky jako **BITOP**, **BMOCT** a **BDOCT** nabízejí bitové operace a možnosti multiplexování/demultiplexování, čímž zvyšují všestrannost knihovny při ovládání kombinační a sekvenční logiky.

## 9 TIME – Bloky pro práci s časem

Knihovna TIME se specializuje na časově založené operace a plánování v systému REXYGEN. Obsahuje bloky jako [DATE](#), [TIME](#) a [DATETIME](#) pro manipulaci s datem a poskytuje základní nástroje pro práci s časovými daty. Knihovna obsahuje [TC](#) pro ovládání vnitřního časovače. Kromě toho se pro plánování používá [WSCH](#), což umožňuje efektivní správu časově závislých úloh. Tato knihovna je zvláště cenná pro systémy vyžadující přesné řízení času a možnosti plánování.

## 10 ARC – Archivace dat

Knihovna je nedílnou součástí archivačního subsystému systému REXYGEN, zaměřená na záznam historie řídicího algoritmu. Zahrnuje bloky pro generování alarmů a událostí, záznam trendů a správu archivů. Tato knihovna je zásadní pro udržení komplexní historie událostí, alarmů a trendů s podporou ukládání na různé typy zařízení jako RAM, zálohovaná paměť a disk, což zajišťuje univerzalitu v ukládání a přístupu k datům.

## 11 STRING – Bloky pro práci s řetězci

Knihovna STRING je věnována manipulaci a analýze řetězců v systému REXYGEN. Zahrnuje bloky jako [CONCAT](#) pro zřetězení řetězců, [FIND](#) pro vyhledávání v řetězcích a [REPLACE](#) pro nahrazování segmentů řetězců. Knihovna nabízí [LEN](#) a [MID](#) pro určení délky řetězce a extrakci podřetězců. Pokročilé porovnávání vzorů poskytuje [REGEXP](#). Konverzní bloky jako [ITOS](#), [STOR](#) a [RTOS](#) převádějí celá a reálná čísla na řetězce, zatímco jednoduchý blok [CNS](#) definuje řetězcovou konstantu. Knihovna navíc obsahuje bloky jako [PJROCT](#) pro analýzu JSON. Tato kolekce bloků je nezbytná pro manipulaci a zpracování řetězcových dat v různých aplikacích.

## 12 PARAM – Bloky pro manipulaci s parametry

Knihovna PARAM je určena pro správu parametrů a zpracování signálů v systému REXYGEN. Obsahuje bloky jako [PARR](#) a jeho varianty pro definování a úpravu různých typů parametrů. Vyhledávací bloky jako [GETPA](#) a [GETPS](#) umožňují extrakci těchto parametrů pro další zpracování. Naopak [SETPA](#), [SETPR](#) a [SETPS](#) se používají pro dynamické nastavení hodnot parametrů. Knihovna navíc obsahuje [SILO](#) a [SILOS](#) pro export a import hodnot ze souboru. Tato knihovna je klíčová pro systémy vyžadující dynamickou manipulaci s parametry a možnosti čtení/ukládání hodnot do souboru.

## 13 MODEL – Simulace dynamických systémů

Knihovna MODEL je zaměřena na modelování a simulaci systému. Zahrnuje bloky jako [CSSM](#) a [DSSM](#) pro spojitě a diskrétní modely stavového prostoru a [DFIR](#) pro digitální filtry s konečnou impulsní odezvou. Knihovna nabízí [EKF](#) pro implementace rozšířeného Kalmanova filtru a [FOPDT](#), [SOPDT](#) pro modely časového zpoždění procesu prvního a druhého řádu. Navíc poskytuje [FMUCS](#) a [FMUINFO](#) pro propojení s funkčními mock-upovými jednotkami a [MDL](#), [MDLI](#) pro rozhraní generických modelů. Pokročilé funkce jsou pokryty bloky jako [CDELSSM](#), [DDELSSM](#) pro stavové modely kontinuálních a diskrétních systémů se zpožděním a [MVD](#) pro modelové pro-

měnné zpoždění, které splňují širokou škálu požadavků na modelování. v systému REXYGEN.

#### 14 MATRIX – Bloky pro maticové a vektorové operace

Knihovna MATRIX je navržena pro pokročilé maticové výpočty a manipulace. Zahrnuje širokou škálu bloků, jako je `MB_DGEMM`, `MB_DTRMM` a `MB_DGER` pro operace typu matice-matice a matice-vektor. Knihovna obsahuje funkce pro rozklad matic (`ML_DGEBRD`, `ML_DGEQRF`), problémy s vlastními hodnotami (`ML_DGEEV`, `ML_DGEES`) a rozklad singulární hodnoty (`ML_DGESDD`). Kromě toho nabízí pomocné bloky jako `MX_MAT`, `MX_VEC` a `MX_FILL` pro vytváření matic a manipulaci, stejně jako specializované bloky, jako je např. `MX_DTRNSP` pro maticovou transpozici a `MX_RAND` pro generování náhodných matic. Tato knihovna je nezbytná pro složité matematické operace zahrnující matice v různých aplikacích.

#### 15 OPTIM – Bloky pro optimalizaci

Knihovna OPTIM je přizpůsobena pro optimalizační algoritmy a procesy. Zahrnuje `QCEDPOPT` pro optimalizace tzv. Economic Dispatch problémů, poskytující pokročilé nástroje pro řešení složitých optimalizačních problémů. Knihovna také obsahuje bloky jako `QP_MPC2QP` a `QP_OASES` pro kvadratické programování, které jsou nezbytné ve scénářích Model Predictive Control (MPC). Navíc je k dispozici `QP_UPDATE` pro aktualizaci parametrů kvadratického programu. Tato knihovna je zvláště užitečná v systémech vyžadujících řešení optimalizace na vysoké úrovni, jako jsou pokročilé řídicí a rozhodovací algoritmy.

#### 16 SPEC – Speciální bloky

Knihovna SPEC zahrnuje rozmanitou sadu funkčních bloků navržených pro integraci široké škály funkcí do automatizace, řídicích systémů a komunikačních protokolů. Od usnadnění přesných termodynamických výpočtů s blokem `STEAM` po umožnění bezproblémové datové komunikace prostřednictvím `UART` a `SMTP` slouží knihovna jako komplexní sada nástrojů pro inženýry a vývojáře. Obsahuje specializované bloky pro spouštění externích programů (`EPC`), zpracování webových požadavků (`HTTP2`). Navíc nabízí unikátní vstupně-výstupní řešení (`RDC`) a všestranné programovací prostředí s `REXLANG`.

#### 17 LANG – Speciální bloky

#### 18 DSP – Zpracování speciálních signálů

Knihovna DSP je přizpůsobena pro pokročilé zpracování digitálních signálů. Obsahuje bloky jako `FFT` pro operace rychlé Fourierovy transformace a `PSD` pro analýzu spektrální hustoty výkonu. Knihovna také nabízí `BSFIFO`, `BSGET`, `BSGETV`, `BSET` a `BSETV` pro ukládání do bufferu a jeho načítání, což umožňuje efektivní manipulaci s daty při úlohách zpracování signálů. Kromě toho `MOSS` poskytuje možnosti výběru signálů na základě více cílů. Tato kolekce bloků je nezbytná pro sofistikovanou analýzu a manipulaci se signály v digitálních systémech.

## 19 MQTTDrv – Komunikace přes MQTT protokol

Knihovna MQTTDrv je určena pro komunikaci IoT (Internet of Things) pomocí protokolu MQTT (Message Queuing Telemetry Transport). Skládá se ze dvou primárních bloků: `MqttPublish` a `MqttSubscribe`. Blok `MqttPublish` se používá pro odesílání zpráv brokerovi MQTT, umožňující publikování dat do MQTT témat. Naopak blok `MqttSubscribe` je určen pro přihlášení k odběru témat a příjem zpráv od brokera. Tato knihovna usnadňuje efektivní a efektivní datovou komunikaci v aplikacích IoT a využívá lehký a široce používaný protokol MQTT pro výměnu zpráv.

## 20 MC\_SINGLE – Řízení pohybu v jedné ose

Knihovna MC\_SINGLE je určena pro řízení pohybu v jednoosých systémech. Obsahuje bloky jako `MC_MoveAbsolute`, `MC_MoveRelative` a `MC_MoveVelocity` pro přesné polohování a ovládání rychlosti. Knihovna obsahuje `MC_Home` pro operace navádění a `MC_Power` pro řízení stavu napájení osy. Pokročilé funkce poskytují `MC_AccelerationProfile`, `MC_PositionProfile` a `MC_VelocityProfile` pro přizpůsobení profilů pohybu. Nabízí také možnosti monitorování a úpravy parametrů prostřednictvím `MC_ReadActualPosition`, `MC_ReadAxisError`, `MC_ReadParameter` a `MC_WriteParameter`. Knihovna navíc obsahuje bloky jako `MC_Halt`, `MC_Reset` a `MC_Stop` pro nouzové a kontrolní operace. Tato knihovna je nezbytná pro aplikace vyžadující přesný a řízený pohyb v jednoosých konfiguracích.

## 21 MC\_MULTI – Řízení pohybu více os

Knihovna MC\_MULTI se specializuje na řízení pohybu ve více osách. Obsahuje bloky jako `MC_CombineAxes` pro synchronizaci více os, `MC_GearIn` a `MC_GearOut` pro převody a `MC_PhasingAbsolute`, `MC_PhasingRelative` pro přesné fázování os. Knihovna nabízí `MC_CamIn` a `MC_CamOut` pro funkce vačky, které umožňují sledování složitých profilů pohybu. `MCP_CamTableSelect` navíc poskytuje flexibilitu při výběru vačkových tabulek a `MC_GearInPos` umožňuje řazení na základě polohy. Tato knihovna je nezbytná pro pokročilé aplikace vyžadující koordinované řízení pohybu napříč více osami.

## 22 MC\_COORD – Koordinované řízení pohybu

Knihovna MC\_COORD je speciálně navržena pro koordinaci víceosého řízení pohybu v rámci komplexních systémů. Zahrnuje řadu bloků, včetně `MC_MoveLinearAbsolute` pro provádění přesných lineárních pohybů, doplněných o `MC_MoveLinearRelative` pro relativní lineární pohyb. Pro provádění kruhového pohybu knihovna obsahuje bloky `MC_MoveCircularAbsolute` a `MC_MoveCircularRelative`, což zajišťuje podrobné kruhové trajektorie. V kontextu správy řízení skupinových os tato knihovna zavádí `MC_AddAxisToGroup`, který je dále podporován funkcemi jako `MC_GroupEnable` pro aktivaci, `MC_GroupDisable` pro deaktivaci a `MC_GroupHalt` pro okamžité zastavení seskupených os. Dále knihovna poskytuje `MC_MoveDirectAbsolute` a `MC_MoveDirectRelative`, které umožňují přímou kontrolu nad pohyby os. Pro navigaci po složitých cestách je k dispozici `MC_MovePath`. Základní funkce monitorování a ovládání jsou usnadněny pomocí `MC_GroupReadActualPosition` pro údaje o

poloze, `MC_GroupReadActualVelocity` pro přehled rychlosti, `MC_GroupReadError` pro detekci chyb a `MC_GroupReadStatus` pro aktualizace stavu. Knihovna navíc integruje `MC_ReadCartesianTransform` a `MC_SetCartesianTransform`, které jsou životně důležité pro procesy karteziánské transformace. Tato kolekce funkcí podtrhuje význam knihovny v aplikacích, které vyžadují synchronizované řízení více os, zejména v oblastech robotiky a automatizačních systémů.

### 23 CanDrv – Komunikace po sběrnici CAN

Knihovna CanDrv je určena pro zpracování komunikace po sběrnici CAN (Controller Area Network) v systému REXYGEN. Obsahuje `CanItem` pro správu datových položek CAN, `CanRecv` pro příjem zpráv ze sběrnice a `CanSend` pro odesílání zpráv. Tato knihovna poskytuje zásadní nástroje pro efektivní a spolehlivou komunikaci přes síť CAN, usnadňuje výměnu dat a řídicích příkazů mezi různými komponentami systému.

### 24 OpcUaDrv – Komunikace pomocí OPC UA

Knihovna OpcUaDrv se specializuje na propojení se servery OPC UA (Open Platform Communications Unified Architecture) pro průmyslovou automatizaci. Skládá se ze tří klíčových bloků: `OpcUaReadValue`, `OpcUaServerValue` a `OpcUaWriteValue`. Blok `OpcUaReadValue` je určen pro čtení dat ze serverů, takže je stěžejní pro sběr dat v automatizovaných systémech. Blok `OpcUaWriteValue` umožňuje zápis dat na servery, což umožňuje řízení a provádění příkazů. Blok `OpcUaServerValue` navíc usnadňuje monitorování a správu hodnot serveru. Tato knihovna slouží jako kritický nástroj pro bezproblémovou komunikaci a interakci se servery OPC UA, čímž rozšiřuje možnosti automatizačních systémů.

Jednotlivé kapitoly příručky na sebe navazují jen volně, a proto mohou být čteny téměř v libovolném pořadí, dokonce může být čtena vždy jen nezbytně nutná informace potřebná k pochopení funkce konkrétního funkčního bloku. Pro tento účel je vhodná zejména elektronická podoba příručky (ve formátu `.pdf`), vybavená hypertextovými záložkami a obsahem, které usnadňují rychlé nalezení příslušných bloků.

Přesto lze ještě doporučit přečtení následující podkapitoly, která popisuje konvence užívané při popisu bloků ve zbytku příručky.

## 1.2 Formát popisu funkčních bloků

Popis každého funkčního bloku se skládá z několika sekcí (v uvedeném pořadí):

Symbol bloku – graficky zobrazuje symbolickou značku bloku

Popis funkce – stručně popisuje funkci daného bloku, aniž by byly uváděny příliš detailní informace.

Vstupy – detailně popisuje všechny vstupy daného bloku

Výstupy – detailně popisuje všechny výstupy daného bloku

Parametry – detailně popisuje všechny parametry daného bloku

Příklady – graficky znázorňuje na jednoduchém příkladu použití daného bloku v kontextu ostatních bloků a často uvádí i obrázek s průběhem vstupních a výstupních signálů tak, aby chování bloku bylo přiblíženo co nejnázorněji.

Pokud je funkce bloku zřejmá, nemusí být sekce Příklady uvedena. V případě, že blok nemá žádný vstup nebo výstup nebo parametr, není ani příslušná sekce v popisu obsažena.

Vstupy, výstupy a parametry jsou popisovány v tabulkové formě:

<code>&lt;jmeno&gt;</code>	<code>[<i>jm</i>]</code> Podrobný popis vstupu (výstupu, parametru) <code>&lt;jmeno&gt;</code> .	<code>&lt;typ&gt;</code>
	Matematický symbol <i>jm</i> na pravé straně prvního sloupce je používán ve vzorcích v sekci Popis funkce a bude uváděn, pokud se od jména vstupu liší víc než jen typograficky. Pokud daná proměnná nabývá pouze několika vyjmenovaných hodnot, je význam těchto hodnot uveden v tomto sloupci.	
	<code>[⊙&lt;def&gt;]</code> [ <code>↓&lt;min&gt;</code> ] [ <code>↑&lt;max&gt;</code> ]	

Význam jednotlivých sloupců je celkem zřejmý. Ve třetím sloupci je vždy uveden pouze `<typ>`. Řídicí systém REXYGEN podporuje typy uvedené v tabulce 1.1. Standardní funkční bloky však nejčastěji používají pro logické proměnné typ `Bool`, pro celočíselné proměnné typ `Long` (I32) a pro reálné proměnné (v pohyblivé řádové čárce) typ `Double` (F64).

Každá takto popsaná proměnná (vstup, výstup či parametr) má v řídicím systému REXYGEN konkrétní implicitní (default) hodnotu `<def>`, uvozenou symbolem `⊙` a podobně i minimální příp. maximální přípustnou hodnotu, uvozenou symbolem `↓`, příp. `↑`. Všechny tyto tři hodnoty mohou být uvedeny ve druhém sloupci, ale nejsou povinné (jsou umístěny v `[ ]`). Pokud není uvedena hodnota `⊙<def>`, je vždy tato hodnota nulová. Není-li uvedena hodnota `↓<min>` příp. `↑<max>`, nabývá minimální příp. maximální hodnoty příslušného typu, viz tabulku 1.1<sup>1</sup>.

### 1.3 Konvence pojmenování proměnných, bloků a subsystémů

Pro usnadnění práce s řídicím systémem REXYGEN se používá několik konvencí. V předchozí podkapitole byly zavedeny všechny používané typy proměnných. Pod pojmem proměnná budeme mít v této podkapitole na mysli vstupy, výstupy a parametry bloků. Ve velké většině bloků se používají pouze tyto tři typy:

`Bool` – pro dvouhodnotové logické proměnné, např. zapnuto/vypnuto, ano/ne, pravda/nepravda, true/false, on/off, apod. V této příručce budeme hodnoty logické jedničky (ano, pravda, true, 1) zapisovat jako `on` a hodnoty logické nuly (ne, nepravda,

<sup>1</sup>Přesný rozsah typu `Large` je -9223372036854775808 až 9223372036854775807.

Typ	Význam	Minimum	Maximum
Bool	logická hodnota 0 nebo 1	0	1
Byte (U8)	8 bit. celé číslo bez znaménka	0	255
Short (I16)	16 bit. celé číslo se znaménkem	-32768	32767
Long (I32)	32 bit. celé číslo se znaménkem	-2147483648	2147483647
Large (I64)	64 bit. celé číslo se znaménkem	$-9.2234 \cdot 10^{18}$	$9.2234 \cdot 10^{18}$
Word (U16)	16 bit. celé číslo bez znaménka	0	65535
DWord (U32)	32 bit. celé číslo bez znaménka	0	4294967295
Float (F32)	32 bit. číslo v pohyblivé ř. čárce	$-3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$
Double (F64)	64 bit. číslo v pohyblivé ř. čárce	$-1.7 \cdot 10^{308}$	$1.7 \cdot 10^{308}$
String	znakový řetězec		

Tabulka 1.1: Typy proměnných systému REXYGEN.

false, 0) jako **off**. To platí i pro vývojové prostředí REXYGEN Studio. V dalších nástrojích a programech třetích stran mohou být jejich hodnoty zobrazovány jako 1 pro **on** a 0 pro **off**. Názvy logických proměnných používají velká písmena, např. RUN, YCN, R1, UP.

**Long (I32)** – pro celočíselné hodnoty, např. číslo sady parametrů, délka trendového bufferu, typ generovaného signálu, chybový kód, výstup čítače, apod. Názvy celočíselných proměnných jsou obvykle psány malými písmeny a počáteční písmeno (vždy malé) je nejčastěji jedno z písmen {i, k, l, m, n, o}, např. *ips*, *l*, *isig*, *iE*, apod. Existuje však několik výjimek z tohoto pravidla, např. *cnt* v bloku COUNT, *btype*, *pctype1*, *pfac* a *afac* v bloku TRND, apod.

**Double (F64)** – pro čísla v pohyblivé řádové čárce (reálná), např. zesílení, saturační meze, výsledky většiny matematických funkcí, parametry PID regulátorů, délky časových intervalů v sekundách, apod. Názvy proměnných v pohyblivé řádové čárce používají pouze malá písmena, např. *k*, *hilim*, *y*, *ti*, *tt*.

Typy funkčních bloků v řídicím systému jsou pojmenovávány velkými písmeny, uvnitř jména se mohou vyskytovat číslice a znak '\_' (podtržítka). Při vytváření uživatelských instancí bloků doporučujeme na začátku ponechat název typu bloku a doplnit jej o uživatelský název, kde doporučujeme používat všechny uvedené typy znaků a navíc malá písmena.

Výslovně se nedoporučuje používat v uživatelských názvech bloků a vytvořených subsystémů znaky s diakritikou a speciální znaky jako jsou mezery, znaky konce řádků, interpunkční znaménka, operátory, apod. Použití těchto znaků omezuje přenositelnost vytvořených algoritmů na různé platformy a může vést k velké nesrozumitelnosti. Jména jsou kontrolována překladačem REXYGEN Compiler a pokud obsahují některý z nevhodných znaků je hlášeno varování.



## 1.4 Kvalita signálu používaná v OPC

Každý signál (vstup, výstup, parametr) v řídicím systému REXYGEN má kromě své hodnoty některého z typů uvedených v tab. 1.1 ještě tzv. *příznaky kvality*. Příznaky kvality používané v řídicím systému REXYGEN jsou shodné s příznaky kvality používanými specifikacemi OPC (Open Platform Communications), viz [1] a obsahují jednobajtovou informaci, jejíž struktura je uvedena v tabulce 1.2.

Číslo bitu	7	6	5	4	3	2	1	0
Váha bitu	128	64	32	16	8	4	2	1
<b>Bitová pole</b>	<b>Kvalita</b>		<b>Substatus</b>				<b>Omezení</b>	
	Q	Q	S	S	S	S	L	L
Špatná (BAD)	0	0	S	S	S	S	L	L
Nejistá (UNCERTAIN)	0	1	S	S	S	S	L	L
(Nevyužito v OPC)	1	0	S	S	S	S	L	L
Dobrá (GOOD)	1	1	S	S	S	S	L	L

Tabulka 1.2: Struktura příznaků kvality

Základní druh kvality určují příznaky QQ v nejvyšších dvou bitech. Podle jejich kombinací uvedených v tabulce rozlišujeme kvalitu dobrou (GOOD), nejistou (UNCERTAIN) a špatnou (BAD). Jemnější rozlišení, tzv. substatus poskytují čtyři bity SSSS. Tyto bity mají různý význam pro různou základní kvalitu. Nejnižší dva bity LL informují o tom, zda daná veličina překročila své meze nebo zda má konstantní hodnotu. Podrobnosti a význam ostatních bitů lze nalézt v kap. 6.8 specifikace [1]. Seznam bloků propagujících kvalitu signálu je uveden v tabulce 1.3.

ABSROT	ABS	ADDHEXD	ADD	ANDHEXD	AND	ASW	ATMT
AVG	AVSI	AVS	BDHEXD	BITOP	BMHEXD	BPF	CMP
CNB	CNDR	CNI	CNR	COUNT	DELM	DEL	DER
DIF	DIV	EAS	EATMT	EDGE	EMD	EQ	EVAR
FNXY	FNX	GAIN	IADD	IDIV	IMOD	IMUL	INTE
INTSM	ISSW	ISUB	ITOI	KDER	LIN	LPF	MINMAX
MUL	NOT	NSCL	ORHEXD	OR	OSD	POL	RDFT
REC	REL	RLIM	RS	RTOI	SELHEXD	SHIFTOCT	SHLD
SINT	SPIKE	SQRT	SQR	SR	SSW	SUB	SWR
TIMER	UTOI	VDEL	ZV4IS				

Tabulka 1.3: Seznam bloků propagující kvalitu signálu

Princip propagace kvality mezi bloky je následující: Vybere se nejhorší kvalita ze všech *datových* vstupů přivedeného na blok a nastaví se na všechny *datové* výstupy. Jakýkoliv nepřipojený vstup má dobrou kvalitu (GOOD). Kvalita na *řídících* vstupech není monitorována, ale *řídící* vstupy mohou ovlivnit propagaci kvality z *datových* vstupů. Na *stavových* výstupech je kvalita vždy dobrá (GOOD).

Například pro blok **DEL** je vstup u *datový*, protože přenáší data. Vstup **R1** je *řídící*, protože slouží k ovládní bloku. Výstup **y** je *datový*, protože přenáší informaci z bloku dál. Výstup **RDY** je *stavový*, protože indikuje stav bloku.

Až na výjimky (**SAI**, **VIN**, **S10F2**) kvalita neovlivňuje algoritmus bloku (tj. aktuální hodnotu na výstupech). Některé bloky mohou mít v důsledku svého algoritmu horší kvalitu (**UNCERTAIN**, **BAD**) (např. **DEL** před naplněním vyrovnávací paměti nebo **DIV** při dělení 0).

## Kapitola 2

# EXEC – Konfigurace exekutivy reálného času

### Obsah

---

ALARMS – Hromadná definice alarmů . . . . .	29
ARC – Archiv systému REXYGEN . . . . .	32
EXEC – Exekutiva reálného času . . . . .	34
HMI – * Konfigurace vizualizace . . . . .	37
INFO – * Dodatečné informace o projektu . . . . .	38
IODRV – Vstupně-výstupní ovladač systému REXYGEN . . . . .	39
IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem	41
LPBRK – Rozpojení zpětné vazby . . . . .	42
MODULE – Rozšiřující modul systému REXYGEN . . . . .	43
OSCALL – Volání funkcí operačního systému . . . . .	44
PROJECT – * Další nastavení projektu . . . . .	45
QTASK – Rychlá úloha řídicího systému REXYGEN . . . . .	46
SLEEP – Časovací blok pro Simulink . . . . .	47
SRTF – Blok pro nastavování příznaků běhu . . . . .	48
STATELOAD – Načtení stavů a parametrů více bloků . . . . .	50
STATESAVE – Uložení stavů a parametrů více bloků . . . . .	52
SYSEVENT – Čtení systémového logu . . . . .	54
SYSLOG – Zápis do systémového logu . . . . .	56
TASK – Standardní úloha řídicího systému REXYGEN . . . . .	57
TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami	59
WWW – * Obsah pro interní webserver . . . . .	61

---

Knihovna EXEC je nezbytná pro nastavení exekutivy v reálném čase v systému REXYGEN a zahrnuje klíčové bloky jako EXEC, TASK, QTASK a HMI. Tyto bloky jsou fundamentální pro správu provádění úkolů, určování priority procesů a pro interakci s rozhraními pro uživatele, čímž podstatně přispívají k efektivitě a ovladatelnosti aplikací v ekosystému REXYGEN.

## ALARMS – Hromadná definice alarmů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok se umísťuje do *hlavního souboru projektu* a slouží pro hromadnou definici alarmů. Tyto alarmy se pak aktivují pomocí bloku [ALM](#) nebo [ALMI](#). Alarmy se definují pomocí externího souboru ve formátu `.csv`. Název tohoto souboru je v parametru `afile`. Alarmy lze aktivovat také pomocí bloků [ALB](#), [ALBI](#), [ALN](#), [ALNI](#), ty ale mají všechny parametry ve svém bloku a nepoužívají hromadnou definici pomocí tohoto bloku [ALARMS](#).

Soubor má následující sloupce:

<code>id</code>	... Identifikátor alarmu, přes který se odkazuje blok <a href="#">ALM</a> i záznamy v archivu.
<code>level</code>	... Položka level v archivu (pokud je ukládání do archivu povoleno).
<code>archives</code>	... Bitové pole, kde každý bit značí, zda se do archivu s tímto číslem ukládají události spojené s alarmem (jeho začátek, konec, potvrzení). Např. 0=do archivu se neukládá, 1=ukládá se do 1. archivu, 2=ukládá se do 2. archivu, 4=ukládá se do 3. archivu, 3=ukládá se do 1. a 2. archivu, apod.
<code>group</code>	... Rezervováno pro budoucí použití. Jedná se o další hodnotu (nebo bitové pole) kterou lze použít k filtraci alarmů ve vizualizaci.
<code>name</code>	... Název alarmu; může být použit jako identifikátor, proto by měl být jednoznačný.
<code>description</code>	... Obsahuje textový popis alarmu. Do textu je možné vkládat formátovací znaky pro vícejazyčné texty a pro vložení hodnoty spojené s alarmem (associated values), viz níže.

### Vícejazyčný popis

Popis alarmu musí být ve tvaru:

```
<ID_jazyk1><popis jazyk1>|<ID_jazyk2><popis jazyk2>|<ID_jazyk3><popis jazyk3>
```

Počet jazyků není omezen, ale celková délka textu je maximálně 32765 bajtů. Pokud uživatel nastaví neznámý jazyk, použije se `jazyk1` (tj. první možnost podle pořadí zadání). Pokud uživatel nastaví prázdný jazyk (tj. `""`), zobrazí se celý text, tj. všechny jazyky včetně formátovacích značek. Příklad: pro popis `en:High voltage alarm|cz:Přepětí` Uživatel, který si nastaví jazyk "en" uvidí popis alarmu "High voltage alarm", uživatel, který si nastaví jazyk "cz" uvidí "Přepětí". Ve všech ostatních případech (např. jazyk "de", ale také "eng", "CZ", "en-us", apod.) se zobrazí "High voltage alarm" (ne proto, že by se preferovala angličtina, ale proto, že tato jazyková verze je uvedena první).

## Associated values

Do textu (popisu alarmu), kam se má vložit hodnota asociované proměnné, se musí vložit formátovací text. Ten má tvar:

```
%<číslo proměnné>[<formát>][:<počet cifer>[:<přesnost>]],
```

příčemž formát je jeden znak s významem:

- b, B** ... binární hodnota (zobrazuje se jako `on`, `off`)
- d, D** ... celočíselná hodnota (zobrazuje se jako dekadické číslo), implicitní formát pro celočíselné typy
- x, X** ... celočíselná hodnota (zobrazuje se jako hexadecimální číslo)
- f, F** ... desetinné číslo v obvyklém tvaru, pokud je uvedena přesnost, je to počet desetinných míst
- e, E** ... desetinné číslo v exponenciálním tvaru, pokud je uvedena přesnost, je to počet desetinných míst
- g, G** ... desetinné číslo, podle hodnoty se automaticky zvolí formát **F** nebo **E**, implicitní formát pro reálnočíselné typy
- s, S** ... textový řetězec

Pokud formát neodpovídá typu, je ignorován a použije se implicitní formát pro daný typ. Pokud je počet znaků ve formátu příliš malý na zobrazení hodnoty, tak se použije více znaků, aby se hodnota zobrazila správně.

Příklady formátů:

- %2** ... hodnota 2. proměnné (`av2` u bloku **ALM** nebo **ALMI**)
- %1:8:2** ... hodnota 1. proměnné (`av1` u bloku **ALM** nebo **ALMI**) na 2 desetinná místa, doplněná na 8 znaků mezerami zleva
- %1e** ... hodnota 1. proměnné (`av1` u bloku **ALM** nebo **ALMI**) v exponenciálním tvaru

Bloky **ALB**, **ALBI** associated values nepoužívají. Pro blok **ALN** a **ALNI** jsou čísla associated values přiřazena takto:

- 1 ... hodnota vstupu `u`
- 2 ... hodnota parametru `h`
- 3 ... hodnota parametru `hh`
- 4 ... hodnota parametru `l`
- 5 ... hodnota parametru `ll`
- 6 ... hodnota parametru `tout`

## Poznámky:

- Soubor `.csv` může mít jako oddělovač čárku i středník. První řádka s názvy sloupců je nepovinná.
- Řádky v souboru musí být uspořádány podle sloupce `id` vzestupně.
- Při překladu se kontroluje, jestli `id` jsou unikátní a to i proti ostatním blokům zapisujícím do archivu (**TRND**, **ALB**, **ALN**, ...)

- Blok má speciální editor (tlačítko **Configure** v parametrickém dialogu). Toto tlačítko má funkci, že pokud soubor neexistuje, tak se vytvoří s příkladem ve správném formátu. Toto je vhodné využít poprvé pro zjištění správného formátu, i když později je možné soubor vytvářet jinak.
- Hodnoty spojené s alarmem se vyhodnocují při vzniku alarmu. Pokud se hodnota během trvání alarmu změní, v alarmovém okně se to nijak neprojeví.
- V komponentě na prohlížení alarmů je ještě sloupec jméno. To je název alarmového bloku (bez typového prefixu, který se často používá), který alarm generuje.

## Parametry

<code>afile</code>	Název <code>.csv</code> souboru s daty	<code>String</code>
--------------------	--	---------------------

## ARC – Archiv systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok ARC slouží v systému REXYGEN pro konfiguraci archivů, sloužících pro průběžné zaznamenávání alarmů, událostí a historických trendů přímo na cílovém zařízení. Vstup `prev` prvního z archivů se propojí s výstupem `Archives` bloku `EXEC`. Další archivy se přidávají propojováním vstupu `prev` s výstupem `next` předchozího archivu. Na každý výstup `next` smí být připojen nejvýše jeden vstup `prev` následujícího archivu, u posledního archivu zůstává výstup `next` nepřipojen. Vzniklá posloupnost určuje pořadí alokace a inicializace jednotlivých archivů v řídicím systému REXYGEN a také určuje index archivu, používaný v parametru `arc` archivačních bloků (viz kap. 10). Archivy jsou číslovány od 1 a jejich maximální počet je omezen na 15 (archiv č. 0 je interní systémový log).

Typ archivu z hlediska zachování dat i po restartu cílového zařízení je určen parametrem `atype`. Přípustné volby závisejí na možnostech cílového zařízení a lze je po úspěšném připojení k danému zařízení zjistit v diagnostice programu REXYGEN Studio.

Archivy jsou na cílovém zařízení tvořeny posloupností úložek proměnné délky (optimalizace paměti a disku), z nichž každá obsahuje časovou značku. Proto dalšími parametry archivu jsou celková velikost v bytech `asize` a maximální počet časových značek `nmarks` pro urychlení sekvenčního vyhledávání v archivu.

Frekvenci zápisu hodnot na disk lze ovlivnit parametrem `period`. U zařízení používajících flash paměť nebo SD karty jako disk není vhodné zapisovat hodnoty příliš často, proto je vhodné nastavit tento parametr na hodnotu v řádech minut. Dále je možné vybrat vhodný zdroj časových značek parametrem `timesrc`.

## Vstup

<code>prev</code>	Vstup sloužící pro připojení prvního archivu na výstup <code>Archives</code> bloku <code>EXEC</code> nebo k připojení na výstup <code>next</code> předchozího archivu	Long (I32)
-------------------	---	------------

## Výstup

<code>next</code>	Výstup sloužící pro zřetězování archivů připojením na vstup <code>prev</code> následujícího archivu	Long (I32)
-------------------	---	------------



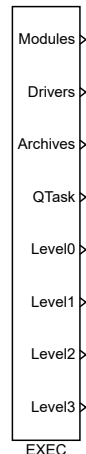
## Parametry

<b>atype</b>	Typ archivu	⊙1	Long (I32)
	1 ..... archiv je alokovan v paměti RAM (po restartu cílového zařízení je nenávratně ztracen)		
	2 ..... archiv je alokovan v zálohované paměti, např. CMOS (po restartu cílového zařízení zůstává zachován)		
	3 ..... archiv je alokovan na disku (zůstává zachován v souboru i po restartu)		
<b>asize</b>	Velikost archivu (v bytech)	↓256 ⊙102400	Long (I32)
<b>nmarks</b>	Počet časových značek pro urychlení sekvenčního vyhledávání v archivu	↓2 ⊙720	Long (I32)
<b>ldaymax</b>	Maximální velikost archivu za den [byte]	↓1000 ↑2147480000 ⊙1048576	Large (I64)
<b>period</b>	Perioda zapisování dat na disk [s]	⊙60.0	Double (F64)
<b>timesrc</b>	Zdroj časových značek	⊙1	Long (I32)
	1 ..... CORETIMER – technologický čas – aktuální tick		
	2 ..... CORETIMER-PRECISE – technologický čas – při spuštění bloku		
	3 ..... RTC – reálný čas z operačního systému – aktuální tick		
	4 ..... RTC-PRECISE – reálný čas z operačního systému – při spuštění bloku		
	4 ..... PFC – hrubý čas s vysokým rozlišením (PerFormanceCounter)		

## EXEC – Exekutiva reálného času

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **EXEC** tvoří základ tzv. *hlavního souboru projektu* ve formátu `.mdl`, kterým se konfiguruje jednotlivé subsystémy řídicího systému **REXYGEN**, a který nemá analogii v systému Matlab-Simulink. Konfigurace bloku **EXEC** a na něj navázané bloky nerealizují žádný výpočetní algoritmus, ale jsou zpracovány překladačem **REXYGEN Compiler** pro sestavení celé aplikace řídicího systému **REXYGEN**.

Konfigurace systému **REXYGEN** se skládá z modulů (**Modules**), vstupně-výstupních ovladačů (**Drivers**), archivačního subsystému (**Archives**) a subsystému reálného času, obsahujícího rychlou výpočetní úlohu (blíže viz blok **QTASK**) a čtyři prioritní úrovně (**Level0** až **Level3**) pro zařazování výpočetních úloh (blíže viz blok **TASK**).

Parametr `tick` určuje základní (nejkratší) periodu, se kterou bude možno spouštět jednotlivé úlohy. Zadaná hodnota je kontrolována překladačem **REXYGEN Compiler** podle zvoleného cílového zařízení. Obecně lze říci, že čím menší hodnota je zadána, tím je větší režie jádra řídicího systému **REXYGEN**.

Periody jednotlivých výpočetních úrovní **Level0** až **Level3** jsou určeny násobky parametrů `ntick0` až `ntick3` a základní periody `tick`. Parametry `pri0` až `pri3` jsou logickými prioritami odpovídajících výpočetních úrovní v systému **REXYGEN**. Poznamenejme, že systém **REXYGEN** používá 32 logických priorit, kterým jsou interně přiřazeny priority závislé na operačním systému cílového zařízení. Nejvyšší logická priorita systému **REXYGEN** je 0, nejnižší má hodnotu 31, přičemž platí, že pokud mají běžet dvě úlohy s různými prioritami, bude úloha s nižší prioritou (vyšší hodnotou) přerušena úlohou s vyšší prioritou (nižší hodnotou). Řídicí systém **REXYGEN** vychází z obecně přijímané myšlenky, že „rychlé“ úlohy (s krátkou periodou vzorkování) je vhodné spouštět s vyšší

prioritou než úlohy „pomalé“ (tzv. *Rate monotonic scheduling*). Proto přednastavené hodnoty priorit `pri0` až `pri3` není ve většině případů třeba měnit; neuvážená změna může vést k těžko předvídatelným důsledkům!

U zařízení s více CPU je možné přiřadit různé úrovně různým CPU. Přiřazení CPU se provádí pomocí parametrů `cpu0` až `cpu3`. CPU jsou číslovány od 0, přičemž -1 označuje výchozí nastavení.

## Výstupy

<code>Modules</code>	Výstup pro připojování rozšiřujících modulů systému REXYGEN, viz blok <a href="#">MODULE</a>	Long (I32)
<code>Drivers</code>	Výstup pro připojování vstupně výstupních ovladačů systému REXYGEN, viz bloky <a href="#">IODRV</a> a <a href="#">TIODRV</a>	Long (I32)
<code>Archives</code>	Výstup pro konfiguraci archivů, viz blok <a href="#">ARC</a>	Long (I32)
<code>QTask</code>	Výstup pro připojení rychlé úlohy (tzv. quick task) s nejvyšší prioritou a s nejkratší periodou, viz blok <a href="#">QTASK</a>	Long (I32)
<code>Level0</code>	Výpočetní úroveň pro zařazování úloh (viz blok <a href="#">TASK</a> ) s vysokou prioritou <code>pri0</code> a krátkou periodou určenou parametrem <code>ntick0</code>	Long (I32)
<code>Level1</code>	Výpočetní úroveň pro zařazování úloh se střední prioritou <code>pri1</code> a středně dlouhou periodou určenou parametrem <code>ntick1</code>	Long (I32)
<code>Level2</code>	Výpočetní úroveň pro zařazování úloh s nízkou prioritou <code>pri2</code> a dlouhou periodou určenou parametrem <code>ntick2</code>	Long (I32)
<code>Level3</code>	Výpočetní úroveň pro zařazování úloh s nejnižší prioritou <code>pri3</code> a nejdélší periodou určenou parametrem <code>ntick3</code>	Long (I32)

## Parametry

<code>target</code>	Cílové zařízení <input type="radio"/> Cílové zařízení <input type="radio"/> <code>Generic target device</code>	String
<code>tick</code>	Základní perioda (tik) jádra řídicího systému REXYGEN a současně též perioda rychlé úlohy <code>QTASK</code> (zadávaná ve vteřinách) <input type="radio"/> 0.05	Double (F64)
<code>ntick0</code>	Určuje základní periodu úloh zařazených do úrovně <code>Level0</code> podle vztahu <code>tick*ntick0</code> ↓1 <input type="radio"/> 10	Long (I32)
<code>ntick1</code>	Určuje základní periodu úloh zařazených do úrovně <code>Level0</code> podle vztahu <code>tick*ntick1</code> ↓ <code>ntick0</code> +1 <input type="radio"/> 50	Long (I32)
<code>ntick2</code>	Určuje základní periodu úloh zařazených do úrovně <code>Level0</code> podle vztahu <code>tick*ntick2</code> ↓ <code>ntick1</code> +1 <input type="radio"/> 100	Long (I32)
<code>ntick3</code>	Určuje základní periodu úloh zařazených do úrovně <code>Level0</code> podle vztahu <code>tick*ntick3</code> ↓ <code>ntick2</code> +1 <input type="radio"/> 1200	Long (I32)
<code>pri0</code>	Priorita všech úloh zařazených do úrovně <code>Level0</code> ↓3 ↑31 <input type="radio"/> 5	Long (I32)
<code>pri1</code>	Priorita všech úloh zařazených do úrovně <code>Level1</code> ↓ <code>pri0</code> +1 ↑31 <input type="radio"/> 9	Long (I32)
<code>pri2</code>	Priorita všech úloh zařazených do úrovně <code>Level2</code> ↓ <code>pri1</code> +1 ↑31 <input type="radio"/> 13	Long (I32)

<code>pri3</code>	Priorita všech úloh zařazených do úrovně <code>Level3</code> ↓ <code>pri2+1</code> ↑ <code>31</code> ⊙ <code>18</code>	Long (I32)
<code>cpu0</code>	Jádro procesoru přiřazené k úrovni 0 (-1=standardní, 0=jádro 0, 1=jádro 1, ...) ↓-1 ↑ <code>127</code> ⊙-1	Long (I32)
<code>cpu1</code>	Jádro procesoru přiřazené k úrovni 1 (-1=standardní, 0=jádro 0, 1=jádro 1, ...) ↓-1 ↑ <code>127</code> ⊙-1	Long (I32)
<code>cpu2</code>	Jádro procesoru přiřazené k úrovni 0 (-1=standardní, 0=jádro 0, 1=jádro 1, ...) ↓-1 ↑ <code>127</code> ⊙-1	Long (I32)
<code>cpu3</code>	Jádro procesoru přiřazené k úrovni 0 (-1=standardní, 0=jádro 0, 1=jádro 1, ...) ↓-1 ↑ <code>127</code> ⊙-1	Long (I32)

## HMI – \* Konfigurace vizualizace

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Parametry

<code>IncludeHMI</code>	Zahrnout soubory HMI do projektu	<input type="radio"/> on	Bool
<code>HmiDir</code>	Výstupní adresář pro soubory vizualizace (HMI)	<input type="radio"/> hmi	String
<code>SourceDir</code>	Zdrojový adresář	<input type="radio"/> hmisrc	String
<code>GenerateWebWatch</code>	Vygenerovat WebWatch vizualizaci z MDL souborů	<input type="radio"/> on	Bool
<code>GenerateRexHMI</code>	Při překladu projektu vygenerovat HMI ze SVG a JS souborů	<input type="radio"/> on	Bool
<code>RedirectToHMI</code>	Webserver bude automaticky přesměrovávat na stránku s HMI	<input type="radio"/> on	Bool
<code>Compression</code>	Aktivovat kompresi dat		Bool

**INFO – \* Dodatečné informace o projektu**

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

**Parametry**

<b>Title</b>	Název projektu	<b>String</b>
<b>Author</b>	Autor projektu	<b>String</b>
<b>Description</b>	Stručný popis projektu	<b>String</b>
<b>Customer</b>	Informace o zákazníkovi	<b>String</b>

## IODRV – Vstupně-výstupní ovladač systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Vstupně-výstupní ovladače jsou v systému REXYGEN implementovány jako rozšiřující moduly (viz blok [MODULE](#)). Modul může obsahovat několik ovladačů, které se do konfigurace systému přidávají pomocí bloků IODRV. Vstup `prev` prvního z ovladačů se propojí s výstupem `Drivers` bloku [EXEC](#). Další ovladače se přidávají propojováním vstupu `prev` s výstupem `next` předchozího ovladače. Na každý výstup `next` smí být připojen nejvýše jeden vstup `prev` následujícího ovladače, u posledního ovladače zůstává výstup `next` nepřipojen. Vzniklá posloupnost určuje pořadí inicializace jednotlivých ovladačů do řídicího systému REXYGEN (pořadí zavádění jednotlivých ovladačů je určeno pořadím modulů, v nichž jsou obsaženy, viz popis bloku [MODULE](#)).

Každý ovladač je v systému REXYGEN identifikován svým jménem, které se zadává v parametru `classname`. Pozor, parametr `classname` rozlišuje velká a malá písmena! Pokud se jméno ovladače liší od jména modulu, obsahujícího daný ovladač, musí se zadat i jméno modulu `module`, jinak se ponechá prázdné. Přesné nastavení těchto dvou parametrů je popsáno v příručce pro každý ovladač systému REXYGEN.

Většina ovladačů má svá vlastní konfigurační data uložena v souborech s příponou `.rio` (REXYGEN Input/Output), jejichž jméno určuje parametr `cfgname`. Soubory `.rio` se vytvářejí na stejném adresáři jako hlavní soubor projektu s příponou `.mdl` v němž je použit tento blok. Konfigurační data ovladačů (např. názvy vstupních/výstupních signálů, jejich připojení na konkrétní fyzické vstupy/výstupy, parametry komunikace se vstupně-výstupním zařízením, apod.) se zadávají ve vestavěných editorech poskytovaných přímo ovladači. V programu REXYGEN Studio systému REXYGEN se editory volají stisknutím tlačítka `Configure` v parametrickém dialogu bloku, v systému Simulink je pro stejnou funkci nutno zaškrtnout pomocné políčko `"Tick this checkbox to call IOdrv EDIT dialog"`.

Zbylé parametry bloku určují chování ovladače při běhu řídicího systému REXYGEN a mají význam jen tehdy, pokud ovladač implementuje vlastní úlohu (viz příručku k odpovídajícímu ovladači). Parametr `factor` je násobkem základní periody `tick` bloku [EXEC](#), určujícím periodu spouštění této úlohy (`factor*tick`). Parametr `stack` udává velikost zásobníku v bytech (není-li v příručce k ovladači napsáno jinak, není jej třeba měnit). Parametr `pri` určuje logickou prioritu úlohy ovladače. Nevhodná hodnota priority může kriticky ovlivnit výkonnost celého řídicího systému, proto doporučujeme konzultovat příručku k ovladači a poté si ověřit zatížení řídicího systému (ovladačů, výpočetních úrovní a úloh) v diagnostice programu REXYGEN Studio. Parametr `cpu` lze použít k určení, kde

má vlákno ovladače běžet na zařízeních s více CPU.

### Vstup

**prev** Vstup sloužící pro k připojení prvního ovladače na výstup **Drivers** bloku **EXEC** nebo k připojení na výstup **next** předchozího ovladače Long (I32)

### Výstup

**next** Výstup sloužící pro zřetězování ovladačů připojením na vstup **prev** následujícího ovladače Long (I32)

### Parametry

<b>module</b>	Jméno modulu, ve kterém je daný vstupně výstupní ovladač obsažen (nemusí se zadávat, je-li shodné s <b>classname</b> )	String
<b>classname</b>	Jméno třídy ovladače, rozlišuje malá a velká písmena!	String
<b>cfgname</b>	Jméno konfiguračního souboru ovladače	⊙DrvClass String
<b>factor</b>	Násobek parametru <b>tick</b> bloku <b>EXEC</b> určující periodu spouštění úlohy ovladače	⊙iodrv.rio Long (I32) ↓1 ⊙10
<b>stack</b>	Velikost zásobníku úlohy ovladače v bytech	↓1024 ⊙10240 Long (I32)
<b>pri</b>	Priorita úlohy ovladače	↓1 ↑31 ⊙3 Long (I32)
<b>cpu</b>	Jádro procesoru přiřazené úloze ovladače (−1=standardní, 0=jádro 0, 1=jádro 1, ...)	↓−1 ↑127 ⊙−1 Long (I32)



## IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Standardní úlohy systému REXYGEN jsou do konfigurace zařazovány pomocí bloku [TASK](#) nebo [QTASK](#). Takové úlohy jsou spouštěny systémovým časovačem, jehož tik (`tick`) se konfiguruje v bloku [EXEC](#).

V některých případech však využití systémového časovače nevyhovuje, např. z důvodu příliš dlouhé nejkratší periody spouštění nebo pokud má být úloha spouštěna od externí události (přerušení od vstupního signálu) apod. V takovém případě může úlohu [IOTASK](#) spouštět přímo vstupně-výstupní ovladač zkonfigurovaný pomocí bloku [TIODRV](#). Zda je uvedený způsob spouštění úloh v konkrétním ovladači implementován a za jakých podmínek, lze najít v uživatelské příručce daného ovladače.

### Vstup

`prev` Vstup sloužící pro k připojení první úlohy na výstup `Tasks` bloku [TIODRV](#) nebo k připojení na výstup `next` předchozí úlohy Long (I32)

### Výstup

`next` Výstup sloužící pro zřetězování úloh připojením na vstup `prev` následující úlohy Long (I32)

### Parametry

`factor` Parametr, který může být využit ovladačem pro určení periody úlohy, viz. uživatelská příručka daného ovladače Long (I32)  $\odot 1$

`stack` Velikost zásobníku (v bytech)  $\odot 10240$  Long (I32)

`filename` Jméno souboru s příponou `.mdl` obsahující algoritmus úlohy; není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněném příponou `.mdl` String

## LPBRK – Rozpojení zpětné vazby

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok LPBRK je pomocným blokem často používaným v řídicích schématech složených z bloků systému REXYGEN. Blok se obvykle umísťuje do všech zpětných vazeb ve schématu. Jeho chování je však v systémech Simulink a REXYGEN odlišné.

V systému Simulink funguje blok LPBRK jako zpoždění signálu o jeden krok. Kdyby nebyl tento blok vložen do každé zpětné vazby, vyhodnotil by systém Simulink (od verze Matlab 6.1), že schéma obsahuje tzv. „rychlou smyčku“ a simulace by po čase selhala.

V systému REXYGEN je při překladač schématu programem REXYGEN Compiler tento blok vypuštěn, avšak ještě před tím způsobí přerušení zpětnovazební smyčky v místě svého výskytu. Pokud po vypuštění všech bloků LPBRK ještě v řídicím schématu zbývá nějaká smyčka, vypíše překladač REXYGEN Compiler varovnou zprávu a zpětnou vazbu rozpojí v místě, které si sám určí. Pro dosažení co nejvyšší kompatibility mezi systémy REXYGEN a Simulink se doporučuje používat blok LPBRK i v konfiguraci řídicího systému REXYGEN.

Poznámka: Od zavedení kvality na výstupech většiny bloků (odvozuje se od kvality na vstupech a stavu bloku) není blok LPBRK z algoritmu vypouštění, ale má na výstupu kvalitu GOOD (signál v pořádku). Původní chování lze vynutit nastavením parametru RB = on. Propagace kvality byla zavedena ve verzi 3.0. Hlavní funkce bloku (indikace zpětnovazebního signálu pro určení pořadí vykonávání bloků) zůstává ve všech případech nezměněna.

### Vstup

u	Vstupní signál	Double (F64)
---	----------------	--------------

### Výstup

y	Výstupní signál	Double (F64)
---	-----------------	--------------

### Parametry

RB	Příznak vypuštění bloku	Bool
----	-------------------------	------

## MODULE – Rozšiřující modul systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Systém REXYGEN má otevřenou architekturu, jeho zabudované funkce lze tedy dále rozšiřovat a doplňovat. Toto rozšiřování je realizováno právě pomocí modulů. Každý modul je určen svým jménem (umístěným pod symbolem bloku). První rozšiřující modul se zařadí do projektu systému REXYGEN tím, že se jeho vstup **prev** propojí s výstupem **Modules** bloku [EXEC](#). Další moduly se přidávají propojováním vstupu **prev** s výstupem **next** předchozího modulu. Na každý výstup **next** smí být připojen nejvýše jeden vstup **prev** následujícího modulu, u posledního modulu zůstává výstup **next** nepřipojen. Vzniklá posloupnost určuje pořadí zavádění jednotlivých modulů a též pořadí jejich inicializace.

### Vstup

<b>prev</b>	Vstup sloužící pro připojení prvního modulu na výstup <b>Modules</b> bloku <a href="#">EXEC</a> nebo k připojení na výstup <b>next</b> předchozího modulu	Long (I32)
-------------	---	------------

### Výstup

<b>next</b>	Výstup sloužící pro zřetězování modulů připojením na vstup <b>prev</b> následujícího modulu	Long (I32)
-------------	---	------------

## OSCALL – Volání funkcí operačního systému

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok `OSCALL` je určen pro volání funkcí operačního systému ze systému `REXYGEN`. Zvolená operace je spuštěna vzestupnou hranou (`off`→`on`) na vstupu `TRG`. Na jednotlivých platformách však nemusí být podporovány všechny funkce. Výsledek operace a případný chybový kód jsou indikovány pomocí výstupů `E` a `iE`.

Pro volání externích programů je možno též využít blok [EPC](#).

### Vstup

<code>TRG</code>	Spuštění zvolené akce	<code>Bool</code>
------------------	-----------------------	-------------------

### Výstupy

<code>E</code>	Příznak chyby	<code>Bool</code>
<code>iE</code>	Kód chyby	<code>Long (I32)</code>
	<code>i</code> ..... obecná chyba systému <code>REXYGEN</code>	

### Parametr

<code>action</code>	Systémová funkce	⊙1 <code>Long (I32)</code>
	1 ..... restartovat systém	
	2 ..... vypnout systém	
	3 ..... zastavit systém ( <code>HALT</code> )	
	4 ..... synchronizace diskových jednotek	
	5 ..... zamknout systémovou partition	
	6 ..... odemknout systémovou partition	
	7 ..... povolit interní webserver	
	8 ..... zakázat interní webserver	

## PROJECT – \* Další nastavení projektu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Parametry

<code>CompileParams</code>	Parametry příkazového řádku programu REXYGEN Compiler		<code>String</code>
<code>SourcesOnTarget</code>	Uložit zdrojové soubory na cílové zařízení	<input type="radio"/> on	<code>Bool</code>
<code>TargetURL</code>	Výchozí adresa cílového zařízení (typicky IP adresa)		<code>String</code>
<code>LibraryPath</code>	Cesta ke knihovnám bloků. Může být absolutní nebo relativní vůči adresáři s projektem.		<code>String</code>

## QTASK – Rychlá úloha řídicího systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok **QTASK** slouží pro zařazení tzv. rychlé úlohy (quick task) s vysokou prioritou do exekutivy řídicího systému **REXYGEN**. Použití této úlohy je opodstatněné v případech, kdy je nutná co nejrychlejší zpracování vstupních signálů, např. pro číslicovou filtraci vstupních signálů zatížených šumem, nebo pro rychlou odezvu na stisk tlačítek připojených přes logické vstupy. Úloha se zařadí do exekutivy reálného času propojením vstupu **prev** s výstupem **QTask** bloku **EXEC**. Rychlá úloha se inicializuje před inicializací výpočetní úrovně **Level0** (viz blok **TASK**).

Zkonfigurovaná úloha **QTASK** běží s logickou prioritou č. 2 a může být v systému **REXYGEN** nejvýše jedna. Algoritmus této úlohy se konfiguruje stejným způsobem jako algoritmus standardní úlohy **TASK** v samostatném souboru s příponou **.mdl**.

Úloha běží s periodou danou součinem parametru **factor** tohoto bloku a parametru **tick** exekutivy **EXEC**. Pro hodnotu **factor=1** bude úloha spouštěna s nejkratší periodou **tick** a také zatížení systému bude největší. Pozor, v každé periodě se musí úloha **QTASK** stihnout za dobu kratší než **tick**, v opačném případě dojde k fatální chybě běhu exekutivy reálného času a vykonávání všech úloh se ukončí! Proto by úloha **QTASK** by měla být používána uvážlivě! Naštěstí lze dobu její exekuce zjistit v diagnostice programu **REXYGEN Studio**.

### Vstup

<b>prev</b>	Vstup, sloužící pro k připojení k výstupu <b>QTask</b> bloku <b>EXEC</b>	Long (I32)
-------------	--	------------

### Parametry

<b>factor</b>	Násobek času <b>tick</b> bloku <b>EXEC</b> určující periodu úlohy ( <b>factor * tick</b> )	Long (I32) ⊙1
<b>stack</b>	Velikost zásobníku (v bytech)	⊙10240 Long (I32)
<b>filename</b>	Jméno souboru s příponou <b>.mdl</b> obsahující algoritmus úlohy; není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněném příponou <b>.mdl</b>	String

## SLEEP – Časovací blok pro Simulink

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SLEEP** slouží k zajištění co nejpřesnější periody spouštění algoritmu. V řídicím systému **REXYGEN** je časování výpočetních úloh zajištěno systémovými prostředky (viz blok **EXEC**), a proto je blok **SLEEP** ignorován. V systému Matlab/Simulink se pracuje se simulačním časem, který může běžet rychleji nebo pomaleji než reálný čas (podle výkonu počítače a složitosti algoritmu).

Má-li simulace běžet v reálném čase, stačí do simulačního algoritmu zařadit blok **SLEEP**, který jej v každém kroku pozastaví na tak dlouho, aby byl jeho algoritmus volán s periodou danou parametrem **ts**. Mechanismus samozřejmě funguje jen v případě, že simulace běží rychleji než ve skutečnosti.

V současné době je blok **SLEEP** implementován pro systém Matlab/Simulink ve verzi pro operační systémy Windows. Vzhledem k tomu, že ve Windows běží obvykle ještě jiné úlohy, které přerušují simulaci, je vhodné nepoužívat příliš krátké periody v řádu milisekund, doporučená hodnota je od 100 ms. Pro správnou funkci je nutné v parametrech simulace **Solver options** nastavit parametr **Type** na **fixed-step, discrete (no continuous states)** a parametr **Fixed step size** na stejnou hodnotu, jako parametr **ts** bloku **SLEEP**. Blok **SLEEP** by měl být nejvýše jeden v celém simulačním schématu (počítáno včetně subsystémů).

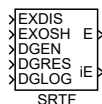
### Parametr

<b>ts</b>	Perioda spouštění simulační úlohy v sekundách	⊙0.1 Double (F64)
-----------	---	-------------------

## SRTF – Blok pro nastavování příznaků běhu

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok **SRTF** (Set Run-Time Flags) slouží pro nastavování příznaků určujících běh úloh, sekvencí (subsystémů) a bloků řídicího systému **REXYGEN**. Tento blok není určen pro Matlab-Simulink. V popisu tohoto bloku bude termín objekt označovat konkrétní objekt řídicího systému **REXYGEN** spouštěný v reálném čase, tj. vstupně-výstupní ovladač, některou z úloh (viz níže), výpočetní sekvenci (subsystém) nebo obyčejný blok systému **REXYGEN**.

Všechny níže uvedené operace jsou prováděny s objektem, jehož úplná cesta je uvedena v parametru **bname**. Není-li tento parametr zadán (prázdný řetězec), provádí se operace s nejbližším vlastníkem daného bloku, tj. pokud je blok obsažen v sekvenci (subsystému) pak s nejbližší nadřazenou sekvencí, jinak přímo s úlohou obsahující daný blok.

Příznaky bloku umožňují:

- **Zakázat spouštění** daného objektu vstupem **EXDIS = on**. Spouštění lze opětovně povolit (**EXDIS = off**). Vstup **EXDIS** nastavuje stejný příznak běhu jako tlačítko **Halt/Run** v pravém horním rohu záložky pracovního prostoru bloku (**Workspace**) v diagnostice programu **REXYGEN Studio**.
- **Jednorázově spustit** daný objekt. Pokud je spouštění objektu zakázáno příznakem **EXDIS = on** nebo je zakázáno z diagnostiky v programu **REXYGEN Studio**), lze vstupem **EXOSH = on** (One Shot Execution) spustit daný objekt právě jednou.
- **Povolit zjišťování diagnostických informací** pro objekt vstupem **DGEN = on**. Příznak je shodný s příznakem **Enable** nastavovaným z programu **REXYGEN Studio** z diagnostických záložek pro jednotlivé objekty (**I/O Driver, Level, Quick Task, Task, I/O Task, Sequence**).
- **Vynulovat diagnostické informace** pro daný objekt vstupem **DGRES = on**. Příznak je rovněž nastaven z programu **REXYGEN Studio** stisknutím tlačítka **Reset** v diagnostické záložce příslušného objektu. Po vynulování informací je v řídicím systému **REXYGEN** příznak automaticky shozen.

Následující tabulka ukazuje, jaké příznaky lze nastavovat pro různé druhy objektů řídicího systému **REXYGEN**.



Druh objektu	EXDIS	EXOSH	DGEN	DGRES
Vstupně výstupní ovladač (I/O Driver)	✓	✓	✓	✓
Výpočetní úroveň (Level)	✓	×	✓	✓
Výpočetní úloha (Task)	✓	✓	✓	✓
Rychlá úloha (Quick Task)	✓	✓	✓	✓
Úloha vstupně-výstupního ovladače (I/O Task)	✓	✓	✓	✓
Výpočetní sekvence (Sequence, subsystém)	✓	×	✓	✓
Obyčejný blok (Block)	✓	×	×	×

## Vstupy

EXDIS	Zakázání spuštění daného objektu	Bool
EXOSH	Jednorázové spuštění daného objektu	Bool
DGEN	Povolení shromažďování diagnostických informací o daném objektu	Bool
DGRES	Vynulování diagnostických údajů o objektu	Bool
DGLOG	Povolení rozšířené logování o objektu	Bool

## Výstupy

E	Příznak chyby off ... bez chyby on .... nastala chyba	Bool
iE	Kód chyby (při E = on) 0 ..... bez chyby 1 ..... objekt nebyl nalezen, neplatný parametr <b>bname</b> 2 ..... interní chyba systému REXYGEN (nesprávné ukazatele) 3 ..... příznak se nepodařilo nastavit (timeout)	Long (I32)

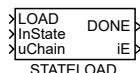
## Parametr

<b>bname</b>	Úplná cesta k bloku (objektu), rozlišuje malá a velká písmena. Jednotlivé vrstvy jsou oddělovány tečkami, názvy objektů kromě úloh ( <b>TASK</b> , <b>QTASK</b> ) začínají jedním z následujících speciálních znaků: ^ ..... výpočetní úroveň (Level), např. ^0 pro Level0 & ..... vstupně-výstupní ovladač (I/O Driver), např. &WcnDrv Jméno úlohy spuštěné vstupně-výstupním ovladačem ( <b>IOTASK</b> ) se zadává ve tvaru &<jmeno_ovladace>.<jmeno_ulohy>	String
--------------	--	--------

## STATELOAD – Načtení stavů a parametrů více bloků

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok **STATELOAD** znovu načte hodnoty stavů a parametrů ze souboru nebo řetězce. Soubor je specifikován parametrem **filename** a musí být ve formátu JSON, který je obvykle ukládán blokem **STATESAVE**. Je také možné načíst data ze vstupu **InState**, který je řetězcem ve formátu JSON stejně jako vstupní soubor. Vstup **InState** se používá, pokud je parametr **filename** prázdný.

Jsou načteny všechny hodnoty, které jsou uloženy v souboru podle konfigurace parametrů **blocks**, **depth** a **mask**. Pokud je parametr **Strict** nastaven na **on**, blok zkontroluje, zda konfigurované bloky a hodnoty odpovídají těm, které jsou uloženy v souboru - pokud není shoda, načtení hodnot se neprovede.

### Vstupy

<b>LOAD</b>	Načtení stavu	<b>Bool</b>
<b>InState</b>	Řetězec JSON k načtení, pokud je parametr <b>filename</b> prázdný	<b>String</b>
<b>uChain</b>	Tento vstup není blokem používán, ale je užitečný pro umístění bloku ve správném pořadí spuštění	<b>Long (I32)</b>

### Parametry

<b>filename</b>	Název souboru pro načtení stavu	<b>String</b>
<b>blocks</b>	Seznam bloků k načtení. Odkazy na bloky musí být relativní cesty (začínající tečkou) a jsou odděleny středníky. Všechny bloky (v aktuálním subsystému) jsou načteny, pokud je tento parametr prázdný	<b>String</b>
<b>depth</b>	Pokud je načítaný blok subsystémem, tento parametr určuje, kolik úrovní je také načteno. 0 = pouze aktuální úroveň, 1 = aktuální úroveň a bloky v subsystémech aktuální úrovně atd.	<b>Long (I32)</b>

↓0 ↑65535

<b>mask</b>	Vyberte, které objekty jsou načteny. Každý bit čísla znamená:	<b>Long (I32)</b>
	<ul style="list-style-type: none"> <li>• 1 ... vstupy</li> <li>• 2 ... výstupy</li> <li>• 4 ... parametry</li> <li>• 8 ... vnitřní stavy</li> <li>• 16 ... parametry pole</li> <li>• 32 ... stavy pole</li> <li>• 64 ... cyklické (trendové) buffery</li> <li>• 256 ... metadata (pouze STATESAVE)</li> </ul>	
		↓0 ↑65535 ⊙65535
<b>LoadOnInit</b>	Načtení během inicializace konfigurace	⊙on <b>Bool</b>
<b>STRICT</b>	Pokud je nastaveno, soubor je zkontrolován vůči aktuální konfiguraci a data jsou odmítnuta, pokud nedochází ke shodě	<b>Bool</b> ⊙on

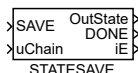
## Výstupy

<b>DONE</b>	Stav načten	<b>Bool</b>
<b>iE</b>	Kód chyby při selhání	<b>Error</b>

## STATESAVE – Uložení stavů a parametrů více bloků

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok **STATESAVE** ukládá hodnoty stavů a parametrů do souboru. Soubor je specifikován parametrem `filename` a je ve formátu JSON, který lze obvykle znovu načíst pomocí bloku **STATELOAD**. Je také možné ukládat data do výstupu `OutState`, který je řetězec JSON ve stejném formátu jako výstupní soubor. Výstup `OutState` se používá, pokud je parametr `filename` prázdný.

Ukládají se všechny hodnoty podle konfigurace parametrů `blocks`, `depth` a `mask`.

### Vstupy

<code>SAVE</code>	Uložení stavu	<code>Bool</code>
<code>uChain</code>	Tento vstup není blokem používán, ale je užitečný pro umístění bloku ve správném pořadí vykonávání.	<code>Long (I32)</code>

### Parametry

<code>filename</code>	Název souboru, kam se ukládá	<code>String</code>
<code>blocks</code>	Seznam bloků k uložení. Názvy bloků musí být relativní cesty (např. začínající tečkou) a jsou odděleny středníky. Všechny bloky (v rámci aktuálního subsystému) jsou uloženy, pokud je parametr prázdný.	<code>String</code>
<code>depth</code>	Pokud je uložený blok subsystémem, tento parametr určuje počet úrovní k uložení. 0 = pouze aktuální úroveň, 1 = aktuální úroveň a bloky přímo v subsystémech aktuální úrovně atd. ↓0 ↑65535	<code>Long (I32)</code>

**mask** Vyberte, které objekty se mají ukládat. Každý bit čísla představuje: **Long (I32)**

- 1 ... vstupy
- 2 ... výstupy
- 4 ... parametry
- 8 ... vnitřní stavy
- 16 ... parametry pole
- 32 ... stavy pole
- 64 ... cyklické (trendové) buffery
- 256 ... metadata (pouze STATESAVE)

↓0 ↑65535 ⊙65535

**SaveOnExit** Pokud je nastaveno, soubor se uloží při ukončení konfigurace. **Bool**  
 ⊙on

## Výstupy

**OutState** Řetězec JSON, kam jsou hodnoty uloženy (pouze pokud je parametr `filename` prázdný) **String**

**DONE** Stav uložení **Bool**

**iE** Kód chyby při selhání **Error**

## SYSEVENT – Čtení systémového logu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Tento blok slouží ke čtení záznamů ze systémového logu nebo archivu. Čtený archiv se vybere parametrem **arc**. Nezobrazují se všechny položky, ale jen ty co projdou filtrem. Filtrovat lze podle ID položky (u systémového logu nemá význam - momentálně mají všechny úložky **id=1**), podle úrovně alarmu/události (v případě systémového logu jsou tam kódovány kategorie) a v případě textové položky ještě podle hodnoty.

Filtr podle ID se nastavuje pomocí parametrů **idfrom** a **idto**, kterými se zvolí interval, který se zobrazuje. Pokud jsou obě hodnoty stejné, tak se zobrazuje jen jedno id a pokud je **idfrom>idto**, tak je filtrování podle id vypnuto a zobrazují se všechna id).

Filtr podle úrovně se nastavuje pomocí parametrů **lvlfrom** a **lvlto**, přičemž platí stejná pravidla jako v předchozím případě.

Filtr podle hodnoty se uplatňuje jen na textové položky (v systémovém logu jsou to všechny). Položka je zobrazena jen pokud je v ní obsažen text z parametru **filter**. Pokud je parametr prázdný, zobrazují se všechny položky. Na jiné než textové položky nemá tento parametr vliv a vždy se zobrazí (pokud vyhovují nastavení dalších filtrů).

Dokud jsou v archivu položky, které vyhovují filtru, tak se zobrazují tak, že v každém tiku je na výstupu jedna položka (v pořadí, jak jsou uloženy v archivu) a výstup **VALID=1**. Když už není další položka, na výstupech jsou hodnoty odpovídající poslední načtené položce, ale **VALID=0**. Výstup **sVal** obsahuje hodnotu textové položky (pro jiné druhy položek je prázdný), Výstup **sVal** obsahuje hodnotu celočíselné položky (pro jiné druhy položek je 0). Ve všech případech jsou všechny parametry (včetně hodnoty) uloženy ve formátu JSON na výstupu **sEvent**. Pro získání potřebných hodnot je možné použít blok **PJSOCT**, popřípadě **PJROCT**.

Poznámky:

- pokud se zařadí více bloků **sysevent**, každý prochází příslušný archiv samostatně. Podle nastaveného filtru se pak může stát, že určitá položka z archivu je na výstupu obou bloků, ale obvykle v jiný okamžik.

### Parametry

<b>arc</b>	Číslo čteného archivu (0=systémový logu)	↓0 ↑16	Long (I32)
<b>filter</b>	Text obsažený v položce		String

<code>idfrom</code>	Nejmenší ID položky, které se zobrazuje	↓0 ↑65535	Long (I32)
<code>idto</code>	Největší ID položky, které se zobrazuje	↓0 ↑65535 ⊙65655	Long (I32)
<code>lvlfrom</code>	Nejmenší úroveň položky, které se zobrazuje	↓0 ↑255	Long (I32)
<code>lvlto</code>	Největší ID položky, které se zobrazuje	↓0 ↑255 ⊙255	Long (I32)

## Výstupy

<code>VALID</code>	Platná (aktuální) výstupní data	Bool
<code>sEvent</code>	Archivní položka (JSON formát)	String
<code>sVal</code>	Hodnota archivní položky (pro text)	String
<code>iVal</code>	Hodnota archivní položky (pro celé číslo)	Long (I32)

**SYSLOG – Zápis do systémového logu**

Symbol bloku

Licence: **STANDARD****Popis funkce**

Blok **SYSLOG** je určený k zapisování libovolných zpráv do systémového logu **REXYGEN**. Lze jej využít na základní logování uživatelských událostí. Pro zápis je potřeba mít v konfiguraci systémového logu povolené zprávy dané úrovně (Target -> System Logs Configuration -> Function block messages).

**Vstupy**

<b>msg</b>	Zpráva, kterou chcete uložit do logu (max. 512 znaků)	<b>String</b>
<b>lvl</b>	Úroveň ukládané zprávy: 0 ..... Error 1 ..... Warning 2 ..... Info 3 ..... Verbose	<b>Long (I32)</b>
<b>RUN</b>	Spuštění zápisu. Zápis do logu probíhá, dokud má vstup <b>RUN</b> hodnotu <b>ON</b>	<b>Bool</b>



## TASK – Standardní úloha řídicího systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Algoritmy řídicích úloh (task) jsou do systému REXYGEN zařazovány pomocí bloků typu TASK. Aplikace řídicího systému může obsahovat několik úloh, které se v konfiguraci systému zařazují do jednotlivých výpočetních úrovní připojením na výstupy Level0 až Level13 bloku EXEC. Vstup `prev` první úlohy dané úrovně `<i>` se propojí s výstupem Level`<i>` bloku EXEC. Další úlohy této úrovně se přidávají propojováním vstupu `prev` s výstupem `next` předchozí úlohy. Na každý výstup `next` smí být připojen nejvýše jeden vstup `prev` následující úlohy stejné úrovně, u poslední úlohy zůstává výstup `next` nepřipojen. Vzniklá posloupnost úloh dané úrovně určuje pořadí inicializace a spouštění úloh této úrovně v řídicím systému REXYGEN. Jednotlivé úrovně se inicializují v pořadí od Level0 do Level13 (rychlá úloha QTASK se inicializuje před úrovní Level0).

Všechny úlohy na dané úrovni `<i>` jsou prováděny se stejnou prioritou, která je určena parametrem `pri<i>` bloku EXEC. Doba provádění úlohy je vypočítána jako násobek parametru `factor` a základní periody úrovně `ntick<i>*tick` v bloku EXEC.

Čas vyhrazený pro provádění úlohy začíná v tiku `start` a končí v tiku `stop`. Hodnoty `start` a `stop` mohou být pevné nebo automaticky určovány RexCore. Pro automatické určování RexCore lze parametry vyplnit následovně:

- `start = -1`: Provádění začíná ihned po skončení předchozí úlohy.
- `start = -2`: Provádění začíná v následujícím tiku po dokončení předchozí úlohy.
- `stop = -1`: Provádění úlohy musí skončit před koncem `ntick<i>*tick`.
- `stop = -2`: Provádění úlohy musí skončit v následujícím `tick`.

Pro pevné doby provádění by měly být hodnoty `start` a `stop` nezápornými celými čísly.

Kompilátor REXYGEN Compiler dodatečně ověřuje, že hodnota parametru `stop` předchozí úlohy je menší nebo rovna hodnotě parametru `stop` následující úlohy. To zajistí, že přidělené časové intervaly pro jednotlivé úlohy se nepřekrývají. Pokud časování jednotlivých úrovní není vhodné, úlohy mohou být přerušeny úlohami a jinými událostmi s vyšší prioritou. V takových případech provádění není zastaveno, ale pouze zpožděno (na rozdíl od bloku QTASK). Sekce Diagnostics programu REXYGEN Studio hodnotí, zda je zpoždění provádění příležitostné nebo trvalé (karty Level a Task).

## Vstup

<code>prev</code>	Vstup sloužící pro k připojení první úlohy na některý z výstupů <code>Level0</code> až <code>Level3</code> bloku <code>EXEC</code> nebo k připojení na výstup <code>next</code> předchozí úlohy dané úrovně	Long (I32)
-------------------	---	------------

## Výstup

<code>next</code>	Výstup sloužící pro zřetězování úloh dané úrovně připojením na vstup <code>prev</code> následující úlohy téže úrovně	Long (I32)
-------------------	--	------------

## Parametry

<code>factor</code>	Faktor spouštění, násobek periody <code>tick*ntick&lt;i&gt;</code> bloku <code>i</code> -té výpočetní úrovně bloku <code>EXEC</code> určující periodu úlohy ( <code>factor * tick * ntick&lt;i&gt;</code> )	Long (I32)
<code>start</code>	Číslo tiky periody dané výpočetní úrovně, na kterém má být úloha spuštěna	Long (I32)
<code>stop</code>	Číslo tiky periody dané výpočetní úrovně, do kterého má být úloha dokončena	Long (I32)
<code>stack</code>	Velikost zásobníku (v bytech)	Long (I32)
<code>filename</code>	Jméno souboru s příponou <code>.mdl</code> obsahující algoritmus úlohy. Není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněným příponou <code>.mdl</code> .	String

## TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok TIODRV slouží pro konfiguraci speciálních ovladačů řídicího systému REXYGEN, které jsou samy schopny spouštět úlohy konfigurované bloky [IOTASK](#), viz. uživatelská příručka konkrétního ovladače. První z úloh [IOTASK](#) se připojí svým vstupem `prev` na výstup `Tasks` bloku TIODRV. Pokud daný ovladač umožňuje spouštět více než jednu úlohu, připojí se další úloha svým vstupem `prev` na výstup `next` předchozí úlohy [IOTASK](#), atd. Počet připojených úloh a jejich pořadí nekontroluje překladač REXYGEN Compiler (jako v případě bloků [TASK](#)), ale přímo vstupně-výstupní ovladač.

Pokud ovladač nemůže pro některou z úloh zajistit periodické spouštění (např. úloha spouštěná od externí události), nastaví pro tuto úlohu odpovídající příznak. Taková úloha nesmí obsahovat bloky, vyžadující konstantní periodu vzorkování (např. většina regulátorů). V případě, že nějaký ze zakázaných bloků je přesto použit, zahlásí exekutiva chybu běhu úlohy, kterou lze zjistit v diagnostice programu REXYGEN Studio. Parametr `cpu` lze použít k určení, kde má vlákno ovladače běžet na zařízeních s více CPU.

### Vstup

<code>prev</code>	Vstup sloužící pro k připojení prvního ovladače na výstup <code>Drivers</code> bloku <a href="#">EXEC</a> nebo k připojení na výstup <code>next</code> předchozího ovladače	Long (I32)
-------------------	---	------------

### Výstupy

<code>next</code>	Výstup pro řetězení ovladačů (s úlohami)	Long (I32)
<code>Tasks</code>	Výstup sloužící pro zřetězování ovladačů připojením na vstup <code>prev</code> následujícího ovladače	Long (I32)

### Parametry

<code>module</code>	Jméno modulu, ve kterém je daný vstupně výstupní ovladač obsažen (nemusí se zadávat, je-li shodné s <code>classname</code> )	String
<code>classname</code>	Jméno třídy ovladače; rozlišuje malá a velká písmena!	String
		⊙DrvClass
<code>cfgname</code>	Jméno konfiguračního souboru ovladače	⊙iodrv.rio String

<b>factor</b>	Násobek parametru <code>tick</code> bloku <b>EXEC</b> určující periodu spouštění úlohy ovladače	↓1 ⊙10	Long (I32)
<b>stack</b>	Velikost zásobníku úlohy ovladače v bytech	↓1024 ⊙10240	Long (I32)
<b>pri</b>	Priorita úlohy ovladače	↓1 ↑31 ⊙3	Long (I32)
<b>cpu</b>	Jádro procesoru přiřazené úloze ovladače (-1=standardní, 0=core 0, 1=core 1, ...)	↓-1 ↑127 ⊙-1	Long (I32)

## WWW – \* Obsah pro interní webserver

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Parametry

<b>Source</b>	Zdrojový adresář	<b>String</b>
<b>Target</b>	Cílový adresář	<b>String</b>
<b>Compression</b>	Aktivovat kompresi dat	<b>Bool</b>



## Kapitola 3

# INOUT – Bloky vstupů a výstupů

### Obsah

---

Display – * Zobrazení vstupní hodnoty . . . . .	64
From, INSTD – Připojení signálu nebo vstupní signál . . . . .	65
Goto, OUTSTD – Zdroj signálu nebo výstupní signál . . . . .	67
GotoTagVisibility – Viditelnost zdroje signálu . . . . .	69
Inport, Outport – Vstupní a výstupní port . . . . .	70
SubSystem – Subsystém . . . . .	72
INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů . . . . .	74
OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů . . . . .	76
OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací . . . . .	78
OUTRSTD – Výstupní signál s verifikací hodnoty . . . . .	79
QFC – Kódování příznaků kvality signálu . . . . .	80
QFD – Dekódování příznaků kvality signálu . . . . .	81
VIN – Ověření kvality vstupního signálu . . . . .	82
VOUT – Nastavení kvality výstupního signálu . . . . .	83

---

Knihovna INOUT slouží jako klíčové rozhraní v systému REXYGEN, umožňuje plynulou interakci s ovladači vstupů/výstupů. Je navržena pro efektivní simultánní zpracování signálů, což je zásadní pro úlohy rychlého řízení. Tato knihovna zjednodušuje propojení mezi řídicími algoritmy a hardwarem, zajišťuje minimální latenci. Zároveň poskytuje pokročilé funkce, jako jsou virtuální propojení (vlajky) signálů pro zvýšenou přehlednost schémat a flexibilitu podsystémů.

Display – \* **Zobrazení vstupní hodnoty**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

u	Vstupní signál	Any
---	----------------	-----

## Parametr

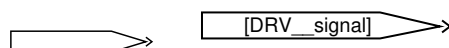
Format	Formát zobrazované hodnoty	⊙1	Long (I32)
	<b>Best fit</b> stejné jako <b>long</b> , ale pro extrémě malá nebo velká čísla jako <b>long_e</b> ; základní formát pro desetinná čísla		
	<b>short</b> běžné zobrazení desetinného čísla s nejvýše 3 ciframi za desetinnou čárkou (tečkou)		
	<b>long ..</b> běžné zobrazení desetinného čísla s nejvýše s plným počtem desetinných míst (tj. až 15)		
	<b>short_e</b> exponenciální (vědecké) zobrazení desetinného čísla s nejvýše 3 ciframi za desetinnou čárkou (tečkou)		
	<b>long_e</b> exponenciální (vědecké) zobrazení desetinného čísla s nejvýše s plným počtem desetinných míst (tj. až 15)		
	<b>bank ..</b> běžné zobrazení desetinného čísla s 2 ciframi za desetinnou čárkou (tečkou)		
	<b>dec ...</b> běžné zobrazení celého čísla (v desítkové soustavě); základní formát pro celá čísla		
	<b>hex ...</b> celé číslo v hexadecimálním formátu (šestnáctková soustava)		
	<b>bin ...</b> celé číslo v binárním formátu (dvojková soustava)		
	<b>oct ...</b> celé číslo v oktálovém formátu (osmičková soustava)		
Decimation	Po kolika periodách je hodnota zobrazována	↓1 ↑100000 ⊙1	Long (I32)
Suffix	Přípona		String
DispValue	Zobrazená hodnota		String



## From, INSTD – Připojení signálu nebo vstupní signál

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **From** (připojení signálu) a **INSTD** (standardní vstup) mají stejný symbol a slouží k připojení vstupního signálu do řídicího algoritmu.

V knihovně bloků naleznete pouze blok **From**. Ten je v případě potřeby při překladač projektu automaticky zkonvertován na blok **INSTD**. O tom, zda daný symbol bloku bude považován za blok **From** nebo **INSTD** rozhoduje překladač **REXYGEN Compiler** podle řetězcového parametru **GotoTag** následovně:

- Obsahuje-li parametr **GotoTag** oddělovač `__` (za sebou dva znaky `'_'`), jedná se o blok **INSTD**. Část parametru (substring) před tímto oddělovačem (v symbolu bloku výše **DRV**) je považována za jméno bloku typu **IODRV** obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program **REXYGEN Compiler** chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru **GotoTag** (za oddělovačem, zde **signal**) považována za jméno vstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná vstupní signál s uvedeným jménem, je vytvořena instance bloku **INSTD**, která bude za běhu v reálném čase získávat hodnotu daného vstupního signálu a přivádět ji při každém spuštění dané úlohy do řídicího algoritmu.
- Pokud parametr **GotoTag** oddělovač `__` neobsahuje, je daný blok považován za blok **From**. Při překladač programem **REXYGEN Compiler** se hledá odpovídající blok **Goto** se stejným parametrem **GotoTag** a požadovanou viditelností danou parametrem **TagVisibility** (viz popis bloku **Goto**). V případě, že není nalezen, oznámí překladač **REXYGEN Compiler** varovnou zprávu a blok **From** odstraní. V opačném případě se propojí odpovídající bloky **From** a **Goto**, jako by byly propojeny „neviditelným“ vodičem. Blok **From** se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

V případě bloku **INSTD** obsahuje parametr **GotoTag** symbol ovladače `<DRV>` a název signálu `<signal>` z daného ovladače:

```
<DRV>__<signal>
```

Například na první digitální vstup I/O zařízení s komunikací Modbus se může odkazovat pomocí `MBM_DI1`. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno `MBM__DI<id>` se bude týkat vstupu č. 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

## Výstup

<code>value</code>	Signál z I/O ovladače nebo bloku <a href="#">Goto</a> . Typ výstupu je určen typem signálu, který je na vlajku přiveden.	<code>Any</code>
--------------------	--	------------------

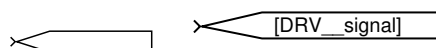
## Parametr

<code>GotoTag</code>	Odkaz na parametr <code>GotoTag</code> bloku <a href="#">Goto</a> , se kterým má být blok <code>From</code> propojen nebo odkaz na vstupní signál ovladače systému REXYGEN, který má být přiveden na výstup bloku.	<code>String</code>
----------------------	--	---------------------

## Goto, OUTSTD – Zdroj signálu nebo výstupní signál

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **Goto** (zdroj signálu) a **OUTSTD** (standardní výstup) mají stejný symbol a slouží k připojení výstupního signálu z řídicího algoritmu.

V knihovně bloků naleznete pouze blok **From**. Ten je v případě potřeby při překladu projektu automaticky zkonvertován na blok **OUTSTD**. O tom, zda daný symbol bloku bude považován za blok **Goto** nebo **OUTSTD** rozhoduje překladač **REXYGEN Compiler** podle řetězcového parametru **GotoTag** následovně:

- Obsahuje-li parametr **GotoTag** oddělovač `__` (za sebou dva znaky `'_'`), jedná se o blok **OUTSTD**. Část parametru (substring) před tímto oddělovačem (v symbolu bloku výše **DRV**) je považována za jméno bloku typu **IODRV** obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program **REXYGEN Compiler** chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru **GotoTag** (za oddělovačem, zde **signal**) považována za jméno výstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná výstupní signál s uvedeným jménem, je vytvořena instance bloku **OUTSTD**, která bude při každém spuštění dané úlohy v reálném čase nastavovat hodnotu daného výstupního signálu z řídicího algoritmu do ovladače.
- Pokud parametr **GotoTag** oddělovač `__` neobsahuje, je daný blok považován za blok **Goto**. Při překladu programem **REXYGEN Compiler** se hledá odpovídající blok **From** se stejným parametrem **GotoTag**, pro který je tento blok **Goto** viditelný (dosažitelný), viz dále. V případě, že není nalezen, oznámí překladač **REXYGEN Compiler** varovnou zprávu a blok **Goto** odstraní. V opačném případě se propojí odpovídající bloky **Goto** a **From**, jako by byly propojeny „neviditelným“ vodičem. Blok **Goto** se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

V případě bloku **OUTSTD** obsahuje parametr **GotoTag** symbol ovladače `<DRV>` a název signálu `<signal>` z :

```
<DRV>__<signal>
```

Například na první digitální výstup I/O zařízení s komunikací Modbus se může odkazovat pomocí `MBM__D01`. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno `MBM_DO<id>` se bude týkat výstupu č. 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

Druhý parametr `TagVisibility` bloku `Goto` určuje viditelnost daného bloku uvnitř souboru `.mdl`. Může nabývat hodnot `local`, `global` a `scoped`, jejichž význam je vysvětlen v tabulce parametrů níže. V případě, že je daný blok přeložen jako blok `OUTSTD` je tento parametr ignorován.

## Vstup

<code>value</code>	Signál odesílaný do I/O ovladače nebo bloku <a href="#">From</a> . V případě napojení na I/O ovladač systému REXYGEN, je typ vstupu určen ovladačem z parametru <code>GotoTag</code> .	Any
--------------------	--	-----

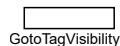
## Parametry

<code>GotoTag</code>	Odkaz na parametr <code>GotoTag</code> bloku <a href="#">From</a> , se kterým má být blok <code>Goto</code> propojen, nebo odkaz na výstupní signál ovladače systému REXYGEN, jehož hodnota je pak určena vstupem bloku.	String
<code>TagVisibility</code>	Viditelnost (dostupnost) daného bloku uvnitř <code>.mdl</code> souboru. Určuje podmínky pro umístění bloku <code>Goto</code> a k němu odpovídajícímu bloku <a href="#">From</a> tak, aby byly vzájemně dostupné:	String
	<ul style="list-style-type: none"> <li>⊙<code>local</code> oba bloky se musí nacházet ve stejném subsystému</li> <li><code>global</code> bloky mohou být umístěny kdekoliv v daném <code>.mdl</code> souboru</li> <li><code>scoped</code> bloky musí být umístěny ve stejném subsystému nebo v jakékoliv hierarchické úrovni pod umístěním bloku <a href="#">GotoTagVisibility</a> se stejným parametrem <code>GotoTag</code></li> </ul>	

## GotoTagVisibility – Viditelnost zdroje signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Bloky `GotoTagVisibility` upřesňují dostupnost (viditelnost) bloků `Goto` s viditelností `scoped`. Symbol (tag) specifikovaný v bloku `Goto` parametrem `GotoTag` je dostupný ze všech bloků `From` ze subsystému, který obsahuje odpovídající blok `GotoTagVisibility` a též ze všech subsystémů v hierarchii níže.

Blok `GotoTagVisibility` je požadován jen pro takové bloky `Goto`, jejichž parametr `TagVisibility` má hodnotu `scoped`. Pokud má parametr `TagVisibility` hodnoty `local` nebo `global`, není blok `GotoTagVisibility` třeba.

Blok `GotoTagVisibility` se používá jen při překladač projektu překladačem REXY-GEN Compiler a ve výsledné binární konfiguraci není obsažen, protože v reálném čase nevykonává žádnou činnost.

### Parametr

<code>GotoTag</code>	Odkaz na parametr <code>GotoTag</code> bloku <code>Goto</code> , jehož viditelnost je <code>String</code> dána umístěním tohoto bloku <code>GotoTagVisibility</code>
----------------------	--

## Inport, Outport – Vstupní a výstupní port

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky typů vstupní port (**Inport**) a výstupní port (**Outport**) slouží k propojování signálů mezi jednotlivými úrovněmi hierarchie. V systému REXYGEN se používají dvěma způsoby:

1. K připojení vstupů a výstupů subsystému. Bloky realizují přechod mezi symbolickou značkou subsystému a jeho vnitřkem (posloupností bloků skrytých v subsystému). Vlastní značka bloku **Inport** nebo **Outport** je obsažena uvnitř subsystému, jméno daného portu je znázorněno v symbolické značce subsystému v nadřazené hierarchické úrovni.
2. K propojení mezi výpočetními úlohami. V tomto případě jsou bloky obsaženy v nejvyšší hierarchické úrovni dané úlohy (souboru `.mdl`). Propojení vzájemně si jménem odpovídajících bloků **Inport** a **Outport** mezi různými úlohami zkontroluje a vytvoří překladač REXYGEN Compiler.

V obou případech je pořadí propojovaných vstupních a výstupních signálů určeno parametrem **Port** daného bloku. Číslování vstupních a výstupních portů je navzájem nezávislé, začíná od 1 a v obou případech se provádí v programu REXYGEN Studio automaticky. Čísla portů musí být navíc jednoznačná v dané hierarchické úrovni, a proto v případě ruční změny čísla portu jsou ostatní porty automaticky přechíslovány. Pozor, pokud jsou přechíslovány porty již připojeného subsystému, dojde v důsledku změny pořadí vstupů (nebo výstupů) k změně připojení signálů v nadřazené úrovni subsystému!

V blocích **Inport** a **Outport** je také možné napevno určit datový typ předávané hodnoty pomocí parametru **OutDataTypeStr**. Pokud není vybrána žádná hodnota, resp. je vybrána možnost **Inherit: auto**, je typ hodnoty určen automaticky.

Do parametru **Description** je možné doplnit textový popis bloku. Tento popis je zobrazen ve vlastnostech subsystému a knihovním bloku, pokud je **Inport** nebo **Outport** použit k definování vstupů a výstupů subsystému.

Poznámka: Tyto bloky nejsou vhodné k propojení polí a jiných odkazů mezi úlohami (výstupy typu odkaz často začínají `ref a` v diagnostice programu REXYGEN Studio mají typ `intptr`). V takovém případě není zajištěno konzistentní čtení a zápis hodnot. V případě pole například mohou být některé hodnoty z jedné periody úlohy a jiné až z další periody. Pro některé struktury může dokonce dojít k pádu RexCore. Pro pole lze využít bloky [SETPA](#) a [GETPA](#). Některé struktury mají toto ošetřené a tam je to výslovně uvedeno v dokumentaci (např. [RM\\_AxisSpline](#)).

## Vstup

`value`      Hodnota odcházející na výstupní připojení nebo do bloku `Inport`      `Any`

## Výstup

`value`      Hodnota přicházející ze vstupního připojení nebo bloku `Outport`      `Any`

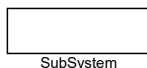
## Parametry

<code>Port</code>	Číslo portu bloku <code>Inport</code> nebo <code>Outport</code>	<code>Long (I32)</code>
<code>OutDataTypeStr</code>	Datový typ hodnoty	<code>String</code>
	Inherit: auto	
	<code>double</code>	
	<code>single</code>	
	<code>uint8</code>	
	<code>int16</code>	
	<code>uint16</code>	
	<code>int32</code>	
	<code>uint32</code>	
	<code>boolean</code>	
	<code>float</code>	
	<code>int64</code>	
	<code>string</code>	
	<code>array</code>	
<code>Description</code>	Textový popis portu	<code>String</code>

## SubSystem – Subsystém

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **Subsystem** je prostředkem pro budování hierarchických řídicích (a simulačních) algoritmů tím, že umožňuje vkládat subsystém do jiného systému (subsystému). Subsystém se skládá z jednotlivých funkčních bloků, jejich vzájemných propojení a případně z dalších subsystémů. Při běhu řídicího systému REXYGEN se subsystém vykonává jako seřazená posloupnost bloků, proto je někdy nazýván výpočetní posloupností (anglicky *sequence*). Mezi bloky této posloupnosti není vykonán žádný jiný blok z okolí subsystému. Ty jsou vykonávány buď striktně před nebo striktně po vyhodnocení celého subsystému.

Subsystém může být vytvořen dvěma způsoby:

- Zkopírováním bloku **Subsystem** z knihovny INOUT do daného schématu (soubor `.mdl`). Po otevření vytvořeného subsystému mohou být do něj přidávány bloky, včetně vstupních portů **Inport** a výstupních portů **Outport**.
- Označením skupiny bloků a volbou příkazu **Vytvoř subsystém** (**Create subsystem** z menu **Edit**). Vybrané bloky jsou nahrazeny subsystémem, po jehož otevření je možné vidět původní bloky a bloky **Inport** a **Outport**, zprostředkující spojení s bloky v nadřazené (původní) úrovni.

Jakmile je subsystém vytvořen, lze do něj vstoupit pomocí double-kliku.

Také je možno vytvořit tzv. masku subsystému a definovat parametry, jejichž hodnoty mohou být využity uvnitř subsystému. Vyberte subsystém a jděte do menu **Edit**→**Subsystem Mask**. Objeví se dialog, ve kterém můžete nadefinovat parametry a jejich popisky (významy).

Jakmile je pro subsystém nadefinována maska, začne se chovat jako standardní blok – po double-kliku se objeví dialog *Block properties*. Ten obsahuje parametry definované v masce subsystému. Pokud je potřeba editovat obsah subsystému s maskou, vyberte jej a jděte do menu **Edit**→**Open subsystem**.

Použití subsystémů je ilustrováno v příkladu 0101-02, který je součástí instalace vývojových nástrojů systému REXYGEN.

### Vstupy

Pořadí a jména vstupů subsystému jsou dána očíslováním a jmény bloků **Inport** použitých uvnitř subsystému.

### Výstupy



Pořadí a jména výstupů subsystému jsou dána očíslováním a jmény bloků `Outport` použitých uvnitř subsystému.

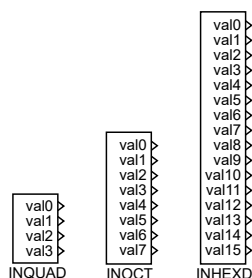
## Parametry

Parametry subsystému jsou definovány v tzv. masce subsystému.

## INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Řídicí systém REXYGEN umožňuje kromě čtení každého jednotlivého vstupu z řízeného procesu také současné čtení několika signálů jedním blokem (například vstupů celého modulu nebo zásuvné desky). Pro popsání způsob získávání vstupů slouží bloky **INQUAD**, **INOCT** a **INHEXD**, které se od sebe liší pouze maximálním počtem současně získaných signálů (po řadě 4, 8 a 16).

Symbol ovladače `<DRV>` a název signálu `<signal>` z daného ovladače je kódován přímo do jména každé instance některého z uvedených bloků ve tvaru:

```
<DRV>_<signal>
```

Kódování jména bloku umístěného přímo pod symbolem bloku v řídicím algoritmu (a tedy na první pohled viditelného ze schématu) dodržuje stejná pravidla jako kódování parametru `GotoTag` bloků **INSTD** a **OUTSTD**. Například na digitální vstupy I/O jednotky s komunikací Modbus se můžeme odkázat pomocí `MBM__DI`. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Použití těchto bloků vícenásobných vstupů minimalizuje režii potřebnou k získání signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc čte všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem jsou na jejich výstupech vyvedeny vstupy řídicího systému, lze nalézt v uživatelské příručce daného ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno `MBM__modul<id>` se bude týkat modulu 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

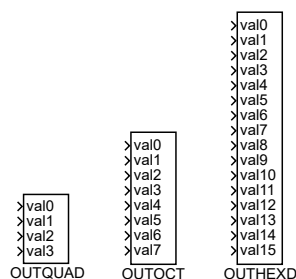
## Výstupy

*vali*      Vstupní signály z procesu přivedené prostřednictvím I/O ovladačů do řídicího algoritmu. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.      Any

## OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Řídicí systém REXYGEN umožňuje kromě zápisu každého jednotlivého výstupu z řízeného procesu také současný zápis několika signálů jedním blokem (například výstupů celého modulu nebo zásuvné desky). Pro popsání způsobu nastavování výstupů slouží bloky **OUTQUAD**, **OUTOCT** a **OUTHEXD**, které se od sebe liší pouze maximálním počtem současně zapisovaných signálů (po řadě 4, 8 a 16). Tyto bloky nemají obdobu v knihovně RexLib pro systém Matlab-Simulink.

Symbol ovladače <DRV> a název signálu <signal> z daného ovladače je kódován přímo do jména každé instance některého z uvedených bloků ve tvaru:

<DRV>\_\_<signal>

Kódování jména bloku umístěného přímo pod symbolem bloku v řídicím algoritmu (a tedy na první pohled viditelného ze schématu) dodržuje stejná pravidla jako kódování parametru `GotoTag` bloků **INSTD** a **OUTSTD**. Například na digitální výstupy I/O jednotky s komunikací Modbus se můžeme odkázat pomocí **MBM\_DD**. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Použití těchto bloků vícenásobných výstupů minimalizuje režii potřebnou k nastavení signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc zapisuje všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem se na jejich vstupy připojují výstupy řídicího systému lze nalézt v uživatelské příručce daného ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno **MBM\_modul<id>** se bude týkat modulu 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

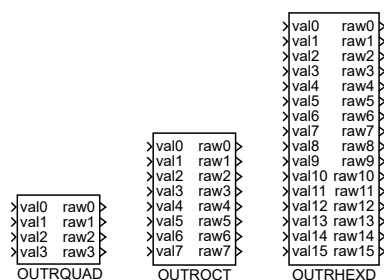
## Vstupy

*vali* Výstupní signály řídicího algoritmu do procesu nastavované prostřednictvím I/O ovladačů. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače. **Any**

## OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací

Symboly bloků

Licence: [ADVANCED](#)



### Popis funkce

Bloky **OUTRQUAD**, **OUTROCT** a **OUTRHEXD** se používají pro nastavování několika výstupů najednou podobně jako bloky **OUTQUAD**, **OUTOCT** a **OUTHEXD**. Navíc však umožňují získat pro každý  $i$ -tý výstup ovladače přivedený na vstup  $val_i$  zpětnou informaci o výsledku zápisu na odpovídajícím výstupu  $raw_i$  daného bloku.

Výstupy  $raw_i$  mohou být použity k informování řídicího algoritmu o výsledku zápisu dvojitým způsobem:

- Hodnotou tohoto výstupu, který může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je v názvu text **raw**).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem **VIN** a dále zpracovat blokem **QFD**.

Hodnota odpovídající danému zápisu se na výstupech  $raw_i$  nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

### Vstupy

$val_i$       výstupní signály řídicího algoritmu do procesu nastavované prostřednictvím I/O ovladačů. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.      Any

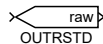
### Výstupy

$raw_i$       Zpětná informace od ovladače o výsledku nastavení odpovídajícího výstupu. Typ a význam signálů je popsán v uživatelské příručce příslušného ovladače.      Any

## OUTRSTD – Výstupní signál s verifikací hodnoty

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok `OUTRSTD` se používá pro nastavování výstupu z řídicího algoritmu podobně jako blok `OUTSTD`. Navíc však umožňuje získat zpětnou informaci o výsledku zápisu na výstupu `raw` daného bloku.

Výstup `raw` může být použit k informování řídicího algoritmu o výsledku zápisu dvojitým způsobem:

- Hodnotou tohoto výstupu, která může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je název `raw`).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem `VIN` a dále zpracovat blokem `QFD`.

Hodnota odpovídající danému zápisu se na výstupu `raw` nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

### Vstup

<code>value</code>	Výstupní signál řídicího algoritmu nastavovaný prostřednictvím ovladače do procesu. Typ a pojmenování signálu je popsáno v uživatelské příručce příslušného ovladače.	<code>Any</code>
--------------------	---	------------------

### Výstup

<code>raw</code>	Zpětná informace od ovladače o výsledku nastavení výstupu. Typ a význam signálu je popsán v uživatelské příručce příslušného ovladače.	<code>Any</code>
------------------	--	------------------

## QFC – Kódování příznaků kvality signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok QFC vytváří kombinací tří složek **iq**, **is** a **il** výsledný 8 bitový kód **iqf** příznaků kvality signálu. Příznaky kvality jsou součástí každého vstupního i výstupního signálu v řídicím systému REXYGEN. Bližší informace o jejich využití jsou uvedeny v kapitole 1.4 této příručky. Knihovna RexLib pro Matlab-Simulink příznaky kvality nepoužívá.

Blok QFC lze využít v kombinaci s blokem [VOUT](#) pro nastavení potřebných příznaků kvality danému signálu. Obrácenou funkci k bloku QFC provádí blok [QFD](#).

### Vstupy

<b>iq</b>	Základní příznaky kvality, viz tab. 1.2, str. 25	Long (I32)
<b>is</b>	Doplňující příznaky kvality, viz [1]	Long (I32)
<b>il</b>	Příznaky dosažení mezních úrovní, viz [1]	Long (I32)

### Výstup

<b>iqf</b>	Bitová kombinace vstupních signálů <b>iq</b> , <b>is</b> a <b>il</b>	Long (I32)
------------	--	------------



## QFD – Dekódování příznaků kvality signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok QFD rozkládá 8 bitové příznaky kvality `iqf` na jednotlivé složky `iq`, `is` a `il`. Příznaky kvality jsou součástí každého vstupního i výstupního signálu v řídicím systému REXYGEN. Bližší informace o jejich využití jsou uvedeny v kapitole 1.4 této příručky. Knihovna RexLib pro Matlab-Simulink příznaky kvality nepoužívá.

Blok QFD lze využít v kombinaci s blokem [VIN](#) pro detailní zpracování příznaků kvality vstupního signálu u bloku [VIN](#) v řídicím algoritmu. Obrácenou funkci k bloku QFD provádí blok [QFC](#).

### Vstup

<code>iqf</code>	Příznaky kvality, které mají být dekomponovány na složky <code>iq</code> , <code>is</code> a <code>il</code>	Long (I32)
------------------	--	------------

### Výstupy

<code>iq</code>	Příznaky základního typu kvality, viz tabulku 1.2, str. 25	Long (I32)
<code>is</code>	Doplňující příznaky kvality, viz [1]	Long (I32)
<code>il</code>	Příznaky dosažení mezních úrovní, viz [1]	Long (I32)

## VIN – Ověření kvality vstupního signálu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok VIN slouží pro ověření kvality vstupního signálu **u** v řídicím systému REXYGEN. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole 1.4 této příručky.

Blok průběžně odděluje příznaky kvality vstupu **u** a nastavuje je na výstup **iqf**. Na základě těchto příznaků a parametru **QU** (Good if Uncertain) jsou vstupní signály v bloku VIN dále zpracovány následujícím způsobem:

- Pro **QU = off** je hodnota výstupu **QG** nastavena na **on**, pouze pokud je kvalita vstupu dobrá (GOOD). V případě špatné (BAD) nebo nejisté (UNCERTAIN) kvality je nastaveno **QG = off**.
- Pro **QU = on** je hodnota výstupu **QG** nastavena na **on**, pokud je kvalita vstupu dobrá (GOOD) nebo nejistá (UNCERTAIN). V případě špatné (BAD) kvality je nastaveno **QG = off**.

Je-li vstupní signál **u** vyhodnocen jako kvalitní (**QG = on**, je přiveden na výstup **yg**. V případě problémů s kvalitou signálu je pro výstup použit náhradní signál ze vstupu **sv** (substitution variable).

### Vstupy

<b>u</b>	Vstupní signál, jehož kvalita se vyhodnocuje. Typ signálu je určen podle typu připojené hodnoty.	Any
<b>sv</b>	Náhradní hodnota pro případ chyby	Any

### Výstupy

<b>yg</b>	Validní výstupní signál ( <b>u</b> pro <b>QG = on</b> nebo <b>sv</b> pro <b>QG = off</b> )	Any
<b>QG</b>	Indikátor platnosti vstupního signálu	Bool
<b>iqf</b>	Úplné příznaky kvality oddělené od vstupu <b>u</b>	Long (I32)

### Parametr

<b>QU</b>	Přípustnost kvality UNCERTAIN <b>off</b> ... kvalita UNCERTAIN je nepřipustná <b>on</b> .... kvalita UNCERTAIN je připustná	Bool
-----------	---	------

## VOUT – Nastavení kvality výstupního signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok VOUT umožňuje signálu *u* nastavit (vnutit) příznaky kvality ze vstupu *iqf*. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole 1.4 této příručky.

### Vstupy

<i>u</i>	Vstup, jehož příznaky kvality mají být nahrazeny. Typ tohoto vstupu je určen podle připojeného signálu.	Any
<i>iqf</i>	Požadované příznaky kvality	Long (I32)

### Výstup

<i>yq</i>	Výsledný signál sestavený z hodnoty vstupu <i>u</i> a příznaků kvality daných hodnotou vstupu <i>iqf</i> . Typ výstupu je určen podle připojeného vstupního signálu <i>u</i> .	Any
-----------	--	-----



## Kapitola 4

# MATH – Matematické bloky

### Obsah

---

ABS – Absolutní hodnota . . . . .	87
ADD – Součet dvou signálů . . . . .	88
ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů . . . . .	89
CNB – Booleovská (logická) konstanta . . . . .	91
CNE – Předdefinovaná konstanta . . . . .	92
CNI – Celočíselná konstanta . . . . .	93
CNR – Reálná konstanta . . . . .	94
DIF – Blok difference . . . . .	95
DIV – Dělení dvou signálů . . . . .	96
EAS – Rozšířené sčítání a odečítání . . . . .	97
EMD – Rozšířené násobení a dělení . . . . .	98
FNX – Výpočet hodnoty funkce jedné proměnné . . . . .	100
FNXY – Výpočet hodnoty funkce dvou proměnných . . . . .	102
GAIN – Násobení konstantou . . . . .	104
GRADS – Gradientní optimalizace . . . . .	105
IADD – Celočíselné sčítání . . . . .	107
IDIV – Celočíselné dělení . . . . .	109
IMOD – Zbytek po celočíselném dělení . . . . .	110
IMUL – Celočíselné násobení . . . . .	111
ISUB – Celočíselné odčítání . . . . .	113
LIN – Lineární interpolace . . . . .	115
MUL – Násobení dvou signálů . . . . .	116
POL – Vyhodnocení polynomu . . . . .	117
REC – Převrácená hodnota . . . . .	118
REL – Relační operace dvou signálů . . . . .	119
RTOI – Konverze reálného čísla na celé číslo . . . . .	120

SQR – Druhá mocnina . . . . .	122
SQRT – Druhá odmocnina . . . . .	123
SUB – Odčítání dvou signálů . . . . .	124
UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem . . . . .	125

---

Knihovna MATH nabízí ucelenou sbírku matematických operací a funkcí. Obsahuje základní aritmetické bloky jako **ADD**, **SUB**, **MUL** a **DIV** pro standardní výpočty a specializovanější bloky, jako je **ABS** pro absolutní hodnoty, **SQRT** pro odmocniny a **SQR** pro odmocnění. Pokročilé funkce poskytují bloky jako **LIN** pro lineární transformace, **POL** pro vyhodnocení polynomů a **FNX**, **FNXY** pro přizpůsobitelné matematické funkce. Knihovna také nabízí operace specifické pro celá čísla prostřednictvím bloků jako **IADD**, **IMUL**, **IDIV** a **IMOD**.

## ABS – Absolutní hodnota

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **ABS** počítá absolutní hodnotu analogového vstupního signálu  $u$ . Na výstupu  $y$  je absolutní hodnota vstupu  $y = |u|$  a výstup **sgn** určuje znaménko vstupu,

$$\text{sgn} = \begin{cases} -1, & \text{pro } u < 0, \\ 0, & \text{pro } u = 0, \\ 1, & \text{pro } u > 0. \end{cases}$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Výstup

$y$	Absolutní hodnota vstupního signálu	Double (F64)
<b>sgn</b>	Indikátor znaménka vstupního signálu	Long (I32)

## ADD – Součet dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ADD počítá součet dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 + u2.$$

Pro sčítání a odečítání více signálů můžete použít blok [ADDOCT](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Výstup

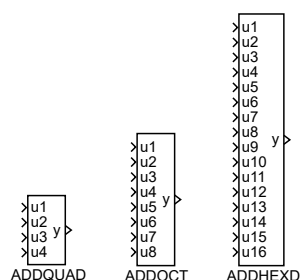
y	Součet vstupních signálů	Double (F64)
---	--------------------------	--------------



## ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky ADDQUAD, ADDOCT a ADDHEXD sčítají více (až 16) vstupních signálů. Parametr  $n1$  udává seznam vstupů, které se místo přičtení odečítají. Pokud je tento parametr prázdný, tak blok provádí funkci  $y = u1 + u2 + u3 + u4 + u5 + u6 + u7 + \dots + u16$ . Pokud bude například  $n1=2, 5, 7$ , tak bude realizována funkce  $y = u1 - u2 + u3 + u4 - u5 + u6 - u7 + \dots + u16$ .

Pro jednoduché operace sčítání a odečítání můžete použít bloky [ADD](#) a [SUB](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

u1	Analogový vstupní signál	Double (F64)
u2	Analogový vstupní signál	Double (F64)
u3	Analogový vstupní signál	Double (F64)
u4	Analogový vstupní signál	Double (F64)
u5	Analogový vstupní signál	Double (F64)
u6	Analogový vstupní signál	Double (F64)
u7	Analogový vstupní signál	Double (F64)
u8	Analogový vstupní signál	Double (F64)
u9	Analogový vstupní signál	Double (F64)
u10	Analogový vstupní signál	Double (F64)
u11	Analogový vstupní signál	Double (F64)
u12	Analogový vstupní signál	Double (F64)
u13	Analogový vstupní signál	Double (F64)
u14	Analogový vstupní signál	Double (F64)
u15	Analogový vstupní signál	Double (F64)
u16	Analogový vstupní signál	Double (F64)

## Parametr

n1	Seznam signálů k odečtení	Long (I32)
----	---------------------------	------------

## Výstup

y	Součet vstupních signálů	Double (F64)
---	--------------------------	--------------

## CNB – Booleovská (logická) konstanta

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok CNB slouží pro zadání Booleovské (logické) konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Parametr

YCN	Booleovská (logická) konstanta	⊙on Bool
	off ... zakázáno	
	on .... povoleno	

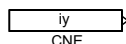
### Výstup

Y	Logický výstupní signál	Bool
---	-------------------------	------

## CNE – Předdefinovaná konstanta

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **CNE** umožňuje výběr celočíselné konstanty z předem připraveného seznamu. Rozbalovací seznam konstant je definován řetězcem **pupstr**, jehož syntaxe je zřejmá z počáteční hodnoty uvedené níže. Na výstupu bloku je celočíselná hodnota odpovídající číslu ze začátku vybrané položky. V případě, že formát řetězce **pupstr** není správný, je na výstupu bloku 0.

V Simulinku je připravena knihovna CNEs, ve které jsou připraveny bloky CNE s nejčastěji používanými seznamy konstant.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Parametr

<b>yenum</b>	Konstanta ze seznamu	⊙1: option A	String
<b>pupstr</b>	Definice seznamu konstant		String
		⊙1: option A 2: option B 3: option C	

### Výstup

<b>iy</b>	Celočíselný výstupní signál	Long (I32)
-----------	-----------------------------	------------

## CNI – Celočíselná konstanta

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok CNI slouží pro zadání celočíselné konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Parametr

icn	Celočíselná konstanta	⊙1	Long (I32)
vtype	Číselný typ	⊙4	Long (I32)
	2 . . . . . Byte (U8)		
	3 . . . . . Short (I16)		
	4 . . . . . Long (I32)		
	5 . . . . . Word (U16)		
	6 . . . . . DWord (U32)		
	10 . . . . . Large (I64)		

### Výstup

iy	Celočíselný výstupní signál	Long (I32)
----	-----------------------------	------------

## CNR – Reálná konstanta

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok CNR slouží pro zadání reálné konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Parametr

ycn	Reálná konstanta	©1.0 Double (F64)
-----	------------------	-------------------

### Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

## DIF – Blok difference

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok DIF počítá diferenci vstupního signálu  $u$  podle vztahu

$$y_k = u_k - u_{k-1},$$

kde  $u = u_k$ ,  $y = y_k$  a  $u_{k-1}$  je vstup  $u$  zpožděný o jeden krok (o periodu  $T_S$ , s níž je blok spouštěn).

Parametr ISSF nastavuje chování bloku v prvním cyklu vykonávání. Pokud je **off**, bude v prvním cyklu na výstupu  $y = u$ . Pro hodnotu **on** bude v prvním cyklu výstup  $y = 0$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u$	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

### Parametr

ISSF	Nulový výstup při spuštění <b>off</b> ... nenulový výstup v prvním cyklu <b>on</b> ... v prvním cyklu je výstup nulový	Bool
------	--	------

### Výstup

$y$	Diference vstupního signálu	Double (F64)
-----	-----------------------------	--------------

## DIV – Dělení dvou signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok DIV dělí dva vstupní analogové signály  $y = u1/u2$ . V případě, že je  $u2 = 0$ , nastaví se výstup  $E = on$  a na výstup  $y$  je dána náhradní hodnota  $y = yerr$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u1$	První analogový vstup bloku	Double (F64)
$u2$	Druhý analogový vstup bloku	Double (F64)

### Parametr

$yerr$	Náhradní hodnota pro případ chyby	⊙1.0 Double (F64)
--------	-----------------------------------	-------------------

### Výstup

$y$	Podíl vstupních signálů	Double (F64)
$E$	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on .... nastala chyba	



## EAS – Rozšířené sčítání a odečítání

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok EAS sčítá vstupní analogové signály  $u_1$ ,  $u_2$ ,  $u_3$  a  $u_4$  s příslušnými vahami  $a$ ,  $b$ ,  $c$  a  $d$ . Výstup  $y$  je pak dán vztahem

$$y = a * u_1 + b * u_2 + c * u_3 + d * u_4 + y_0.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u_1$	První analogový vstup bloku	Double (F64)
$u_2$	Druhý analogový vstup bloku	Double (F64)
$u_3$	Třetí analogový vstup bloku	Double (F64)
$u_4$	Čtvrtý analogový vstup bloku	Double (F64)

### Parametr

$a$	Váhový koeficient pro vstup $u_1$	⊙1.0	Double (F64)
$b$	Váhový koeficient pro vstup $u_2$	⊙1.0	Double (F64)
$c$	Váhový koeficient pro vstup $u_3$	⊙1.0	Double (F64)
$d$	Váhový koeficient pro vstup $u_4$	⊙1.0	Double (F64)
$y_0$	Aditivní konstanta (bias)		Double (F64)

### Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------

## EMD – Rozšířené násobení a dělení

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok EMD slouží k násobení a dělení vstupních analogových signálů  $u_1$ ,  $u_2$ ,  $u_3$  a  $u_4$  s příslušnými vahami  $a$ ,  $b$ ,  $c$  a  $d$ . Výstup  $y$  je pak dán vztahem

$$y = \frac{(a * u_1 + a_0)(b * u_2 + b_0)}{(c * u_3 + c_0)(d * u_4 + d_0)}. \quad (4.1)$$

V případě, že jmenovatel vztahu (4.1) je roven 0, nastaví se výstup  $E = \text{on}$  a na výstup  $y$  je dána náhradní hodnota  $y = \text{yerr}$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$u_1$	První analogový vstup bloku	Double (F64)
$u_2$	Druhý analogový vstup bloku	Double (F64)
$u_3$	Třetí analogový vstup bloku	Double (F64)
$u_4$	Čtvrtý analogový vstup bloku	Double (F64)

## Parametr

$a$	Váhový koeficient pro vstup $u_1$	⊙1.0	Double (F64)
$a_0$	Aditivní konstanta pro vstup $u_1$		Double (F64)
$b$	Váhový koeficient pro vstup $u_2$	⊙1.0	Double (F64)
$b_0$	Aditivní konstanta pro vstup $u_2$		Double (F64)
$c$	Váhový koeficient pro vstup $u_3$	⊙1.0	Double (F64)
$c_0$	Aditivní konstanta pro vstup $u_3$		Double (F64)
$d$	Váhový koeficient pro vstup $u_4$	⊙1.0	Double (F64)
$d_0$	Aditivní konstanta pro vstup $u_4$		Double (F64)
$yerr$	Náhradní hodnota pro případ chyby	⊙1.0	Double (F64)

## Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------

<b>E</b>	Indikátor chyby - dělení nulou	<b>Bool</b>
	<b>off</b> ... bez chyby	
	<b>on</b> ... nastala chyba	

## FNX – Výpočet hodnoty funkce jedné proměnné

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok FNX počítá hodnotu základních matematických funkcí jedné proměnné. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určena parametrem `ifn`.

Tabulka funkcí bloku FNX:

ifn: zkratka	funkce	omezení u
1: <code>acos</code>	arcus cosinus	$u \in \langle -1.0, 1.0 \rangle$
2: <code>asin</code>	arcus sinus	$u \in \langle -1.0, 1.0 \rangle$
3: <code>atan</code>	arcus tangens	–
4: <code>ceil</code>	zaokrouhlení na nejbližší vyšší celé číslo	–
5: <code>cos</code>	cosinus	–
6: <code>cosh</code>	cosinus hyperbolický	–
7: <code>exp</code>	exponenciální křivka $e^u$	–
8: <code>exp10</code>	exponenciální křivka $10^u$	–
9: <code>fabs</code>	absolutní hodnota	–
10: <code>floor</code>	zaokrouhlení na nejbližší nižší celé číslo	–
11: <code>log</code>	logaritmus	$u > 0$
12: <code>log10</code>	dekadický logaritmus	$u > 0$
13: <code>random</code>	náhodné číslo $z \in \langle 0, 1 \rangle$ (nezávisí na u)	–
14: <code>sin</code>	sinus	–
15: <code>sinh</code>	sinus hyperbolický	–
16: <code>sqr</code>	druhá mocnina	–
17: <code>sqrt</code>	druhá odmocnina	$u > 0$
18: <code>srand</code>	mění násadu pro funkci <code>random</code> na u	$u \in \mathbb{N}$
19: <code>tan</code>	tangens	–
20: <code>tanh</code>	tangens hyperbolický	–

*Poznámka:* Všechny trigonometrické funkce pracují s hodnotami v radiánech.

V případě, že vstup u je mimo povolený rozsah nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), např. výpočet odmocniny záporného čísla, je aktivován chybový výstup `E = on` a na výstup y je nastavena náhradní hodnota `y = yerr`.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

u Analogový vstupní signál Double (F64)

## Parametr

ifn Typ funkce ☉1 Long (I32)

- 1 ..... acos
- 2 ..... asin
- 3 ..... atan
- 4 ..... ceil
- 5 ..... cos
- 6 ..... cosh
- 7 ..... exp
- 8 ..... exp10
- 9 ..... fabs
- 10 .... floor
- 11 .... log
- 12 .... log10
- 13 .... random
- 14 .... sin
- 15 .... sinh
- 16 .... sqr
- 17 .... sqrt
- 18 .... srand
- 19 .... tan
- 20 .... tanh

yerr Náhradní hodnota pro případ chyby Double (F64)

## Výstup

y Výsledek vybrané funkce Double (F64)

E Příznak chyby Bool

- off ... bez chyby
- on .... nastala chyba

## FNXY – Výpočet hodnoty funkce dvou proměnných

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok **FNXY** počítá hodnotu základních matematických funkcí dvou proměnných. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určena parametrem **ifn**.

Tabulka funkcí bloku **FNXY**:

ifn: zkratka	funkce	omezení $u1, u2$
1: <b>atan2</b>	arcus tangens $u1/u2$	–
2: <b>fmod</b>	zbytek po dělení $u1/u2$	$u2 \neq 0.0$
3: <b>pow</b>	výpočet mocniny $y = u1^{u2}$	viz níže

Funkce **atan2** vrací funkční hodnotu v intervalu  $(-\pi, \pi)$ . Pro určení správného kvadrantu se využívá znamének obou vstupů  $u1$  a  $u2$ .

Funkce **fmod** počítá zbytek po dělení  $u1/u2$  tak, že platí  $u1 = i * u2 + y$ , kde  $i$  je celé číslo, výstup  $y$  má stejné znaménko jako vstup  $u1$  a pro absolutní hodnotu výstupu  $y$  platí:  $|y| < |u2|$ .

Výpočet mocniny funkcí **pow** se řídí následujícími pravidly:

- Nepracuje se vstupními hodnotami  $u1$  a  $u2$  většími než  $2^{64}$ ,
- $u1^0 = 1$  pro libovolné  $u1$  (i  $u1 = 0$ ),
- $0^{u2}$  vrací chybu pro  $u2 < 0$ .

V případě, že vstup  $u2$  nespĺňuje omezení nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), je aktivován chybový výstup **E = on** a na výstup  $y$  je nastavena náhradní hodnota  $y = yerr$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u1$	První analogový vstup bloku	Double (F64)
$u2$	Druhý analogový vstup bloku	Double (F64)

## Parametr

<code>ifn</code>	Typ funkce	⊙1	Long (I32)
	1 ..... atan2		
	2 ..... fmod		
	3 ..... pow		
<code>yerr</code>	Náhradní hodnota pro případ chyby		Double (F64)

## Výstup

<code>y</code>	Výsledek vybrané funkce		Double (F64)
<code>E</code>	Příznak chyby		Bool
	off ... bez chyby		
	on ... nastala chyba		

## GAIN – Násobení konstantou

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **GAIN** násobí analogový vstup  $u$  reálnou konstantou  $k$ . Výstup je pak

$$y = ku.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Parametr

$k$	Zesílení	©1.0 Double (F64)
-----	----------	-------------------

Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------



## GRADS – Gradientní optimalizace

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok **GRADS** umožňuje provádět jednodimenzionální minimalizaci funkce  $f(x, v)$  gradientní metodou, kde  $x \in \langle x_{\min}, x_{\max} \rangle$  je optimalizační proměnná a  $y$  je libovolná vektorová proměnná. Předpokládá se, že pro daný výstup  $x$  v kroku  $k$  je hodnota funkce  $f(x, v)$  vyčíslena na vstupu  $f$  v kroku  $(k + n)$ . To značí, že jednotlivé iterace gradientní metody jsou prováděny s periodou  $n \cdot T_S$ , kde  $T_S$  je perioda spouštění bloku **GRADS**. Délka kroku gradientní metody je určována podle vztahu

$$\begin{aligned} grad &= (f_i - f_{i-1}) * (dx)_{i-1} \\ (dx)_i &= -gamma * grad, \end{aligned}$$

kde  $k$  značí číslo iterace. Je-li krok  $((dx)_i < d_{\min})$  nebo  $((dx)_i > d_{\max})$ , potom je příslušně omezen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>f</b>	Hodnota minimalizované funkce $f(\cdot)$ v bodě $x$	Double (F64)
<b>x0</b>	Startovní bod optimalizace	Double (F64)
<b>START</b>	Spouštěcí signál (náběžná hrana)	Bool
<b>BRK</b>	Signál pro přerušení	Bool

### Parametr

<b>xmin</b>	Dolní mez přípustného intervalu optimální proměnné $x$	Double (F64)
<b>xmax</b>	Horní mez přípustného intervalu optimální proměnné $x$	$\odot 10.0$ Double (F64)
<b>gamma</b>	Koeficient gradientní metody určující velikost kroku	$\odot 0.3$ Double (F64)
<b>d0</b>	Počáteční krok gradientní metody	$\odot 0.05$ Double (F64)
<b>dmin</b>	Minimální krok gradientní metody	$\odot 0.01$ Double (F64)
<b>dmax</b>	Maximální krok gradientní metody	$\odot 1.0$ Double (F64)
<b>n</b>	Perioda jedné iterace (v periodách vzorkování bloku)	$\odot 100$ Long (I32)
<b>itermax</b>	Maximální počet iterací před ukončením	$\odot 20$ Long (I32)

## Výstup

<code>x</code>	Aktuální hodnota optimalizované proměnné	Double (F64)
<code>xopt</code>	Výsledná optimální hodnota proměnné $x$	Double (F64)
<code>fopt</code>	Výsledná optimální hodnota funkce $f(x)$	Double (F64)
<code>BSY</code>	Příznak probíhající operace	Bool
<code>iter</code>	Číslo aktuální iterace	Long (I32)
<code>E</code>	Příznak chyby	Bool
	<code>off ...</code> bez chyby	
	<code>on ....</code> nastala chyba	
<code>iE</code>	Kód chyby	Long (I32)
	<code>1 .....</code> $x$ je mimo meze	
	<code>2 .....</code> $x$ je na mezi	

## IADD – Celočíselné sčítání

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok IADD sečte dva vstupní celočíselné signály  $n = i1 + i2$ . V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem `vtype`. Pokud se součet vejde do rozsahu proměnné, je výsledkem normální součet. V opačném případě výsledek závisí na hodnotě parametru `SAT`.

Pro `SAT = off` se přetečení rozsahu nekontroluje, tj. nastaví se výstup `E = off` a výstup `n` tak, jak počítá procesor. Například pro typ `Short`, který má rozsah `-32768..+32767`, dostaneme `30000 + 2770 = -32766`.

Pro `SAT = on` se při přetečení rozsahu nastaví výstup `E = on` a na výstup `n` je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme `30000 + 2770 = 32767`).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>i1</code>	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
<code>i2</code>	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

### Parametr

<code>vtype</code>	Číselný typ	⊙4	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		
<code>SAT</code>	Kontrola přetečení		Bool
	<code>off</code> ... přetečení se nekontroluje		
	<code>on</code> ... přetečení se kontroluje		

### Výstup

<code>n</code>	Celočíselný součet vstupních signálů		Long (I32)
----------------	--------------------------------------	--	------------

E	Příznak chyby	Bool
	off ... bez chyby	
	on ... nastala chyba	

## IDIV – Celočíselné dělení

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok IDIV dělí dva vstupní celočíselné signály  $n = i1 \div i2$ , kde  $\div$  označuje operátor celočíselného dělení. Pokud je obyčejný (neceločíselný, normální) podíl obou operandů celé číslo je tato hodnota i výsledkem celočíselného dělení. V opačném případě je výsledkem hodnota, která vznikne „odříznutím“ desetinné části normálního podílu k nejbližšímu celému číslu směrem blíže k nule. V případě, že  $i2 = 0$ , nastaví se výstup  $E = on$  a na výstup  $n$  je dána náhradní hodnota  $n = nerr$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$i1$	První celočíselný vstup bloku	$\downarrow -9.22E+18$ $\uparrow 9.22E+18$	Long (I32)
$i2$	Druhý celočíselný vstup bloku	$\downarrow -9.22E+18$ $\uparrow 9.22E+18$	Long (I32)

### Parametr

$vtype$	Číselný typ	$\odot 4$	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		
$nerr$	Náhradní hodnota pro případ chyby	$\odot 1$	Long (I32)

### Výstup

$n$	Celočíselný podíl vstupních signálů	Long (I32)
$E$	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on .... nastala chyba	

## IMOD – Zbytek po celočíselném dělení

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok IMOD dělí dva vstupní celočíselné signály  $n = i1 \% i2$ , kde  $\%$  označuje operátor zbytku celočíselného dělení (modulo). Pokud jsou obě čísla kladná a dělitel větší než jedna, je výsledek buď nula (pro soudělná čísla) nebo kladné číslo menší než dělitel. V případě, že je jedno z čísel záporné, má výsledek znaménko dělence, např.  $15 \% 10 = 5$ ,  $15 \% (-10) = 5$ , ale  $(-15) \% 10 = -5$ . V případě, že  $i2 = 0$ , nastaví se výstup  $E = on$  a na výstup  $n$  je dána náhradní hodnota  $n = nerr$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$i1$	První celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)
$i2$	Druhý celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)

### Parametr

$vtype$	Číselný typ	$\odot 4$	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		
$nerr$	Náhradní hodnota pro případ chyby	$\odot 1$	Long (I32)

### Výstup

$n$	Zbytek po celočíselném dělení	Long (I32)
$E$	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on ... nastala chyba	

## IMUL – Celočíselné násobení

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok IMUL vynásobí dva vstupní celočíselné signály  $n = i1 * i2$ . V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem `vtype`. Pokud se součin vejde do rozsahu proměnné, je výsledkem normální součin. V opačném případě výsledek závisí na hodnotě parametru `SAT`.

Pro `SAT = off` se přetečení rozsahu nekontroluje, tj. nastaví se výstup `E = off` a výstup `n` tak jak počítá procesor (například pro typ `Short`, který má rozsah `-32768..+32767` dostaneme `2000 * 20 = -25536`).

Pro `SAT = on` se při přetečení rozsahu nastaví výstup `E = on` a na výstup `n` je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme `2000 * 20 = 32767`).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>i1</code>	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
<code>i2</code>	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

### Parametr

<code>vtype</code>	Číselný typ	⊙4	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		
<code>SAT</code>	Kontrola přetečení		Bool
	<code>off</code> ... přetečení se nekontroluje		
	<code>on</code> ... přetečení se kontroluje		

### Výstup

<code>n</code>	Celočíselný součin vstupních signálů		Long (I32)
----------------	--------------------------------------	--	------------

E	Příznak chyby	Bool
	off ... bez chyby	
	on ... nastala chyba	



## ISUB – Celočíselné odčítání

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok ISUB sečte dva vstupní celočíselné signály  $n = i1 - i2$ . V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem `vtype`. Pokud se rozdíl vejde do rozsahu proměnné, je výsledkem normální rozdíl. V opačném případě výsledek závisí na hodnotě parametru `SAT`.

Pro `SAT = off` se přetečení rozsahu nekontroluje, tj. nastaví se výstup `E = off` a výstup `n` tak jak počítá procesor (například pro typ `Short`, který má rozsah `-32768..+32767` dostaneme `30000 - -2770 = -32766`).

Pro `SAT = on` se při přetečení rozsahu nastaví výstup `E = on` a na výstup `n` je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme `30000 - -2770 = 32767`).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>i1</code>	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
<code>i2</code>	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

### Parametr

<code>vtype</code>	Číselný typ	⊙4	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		
<code>SAT</code>	Kontrola přetečení		Bool
	<code>off</code> ... přetečení se nekontroluje		
	<code>on</code> ... přetečení se kontroluje		

### Výstup

<code>n</code>	Celočíselný rozdíl vstupních signálů		Long (I32)
----------------	--------------------------------------	--	------------

E	Příznak chyby	Bool
	off ... bez chyby	
	on ... nastala chyba	

## LIN – Lineární interpolace

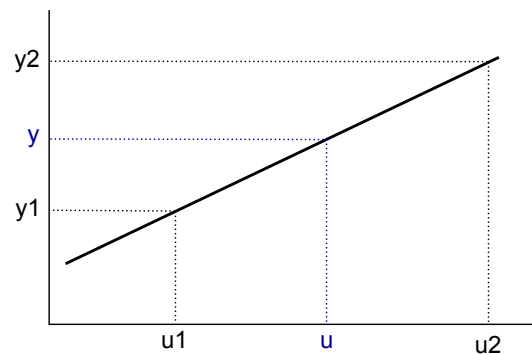
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LIN počítá lineární interpolaci. Následující obrázek ilustruje výpočet výstupu  $y$  ze vstupu  $u$  a ze zadaných bodů  $[u1, y1]$  a  $[u2, y2]$ .



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Parametr

$u1$	Souřadnice prvního bodu v ose x	Double (F64)
$y1$	Souřadnice prvního bodu v ose y	Double (F64)
$u2$	Souřadnice druhého bodu v ose x	⊙1.0 Double (F64)
$y2$	Souřadnice druhého bodu v ose y	⊙1.0 Double (F64)

Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------

## MUL – Násobení dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MUL násobí dva vstupní analogové signály  $y = u1 \cdot u2$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Výstup

y	Součin vstupních signálů	Double (F64)
---	--------------------------	--------------

## POL – Vyhodnocení polynomu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok POL počítá hodnotu polynomiální funkce ve tvaru:

$$y = a_0 + a_1u + a_2u^2 + a_3u^3 + a_4u^4 + a_5u^5 + a_6u^6 + a_7u^7 + a_8u^8.$$

Pro zajištění numerické robustnosti je polynom interně vyhodnocen pomocí Hornerova schématu.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
---	--------------------------	--------------

Parametr

a0	Koeficient $u^0$	Double (F64)
a1	Koeficient $u^1$	⊙1.0 Double (F64)
a2	Koeficient $u^2$	Double (F64)
a3	Koeficient $u^3$	Double (F64)
a4	Koeficient $u^4$	Double (F64)
a5	Koeficient $u^5$	Double (F64)
a6	Koeficient $u^6$	Double (F64)
a7	Koeficient $u^7$	Double (F64)
a8	Koeficient $u^8$	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

## REC – Převrácená hodnota

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok REC počítá převrácenou hodnotu vstupního signálu  $u$ . Výstup je pak

$$y = \frac{1}{u}.$$

Pokud je vstup  $u = 0$ , nastaví se chybový výstup  $E = \text{on}$  a na výstupu  $y$  je náhradní hodnota `yerr`.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Parametr

<code>yerr</code>	Náhradní hodnota pro případ chyby	⊙1.0 Double (F64)
-------------------	-----------------------------------	-------------------

Výstup

$y$	Analogový výstupní signál	Double (F64)
$E$	Indikátor chyby - dělení nulou	Bool
	<code>off</code> ... bez chyby	
	<code>on</code> ... nastala chyba	

## REL – Relační operace dvou signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok REL vyhodnocuje binární relaci  $u1 \circ u2$  z hodnot vstupů a podle výsledku relace „ $\circ$ “ je nastavována hodnota výstupu  $Y$  na hodnotu **on** (relace platí) nebo **off** (relace neplatí). Kód binární operace je uveden v parametru **irel** popsaném níže.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u1</b>	První analogový vstup bloku	Double (F64)
<b>u2</b>	Druhý analogový vstup bloku	Double (F64)

### Parametr

<b>irel</b>	Typ relace	⊙1 Long (I32)
1	..... rovnost (==)	
2	..... nerovnost (!=)	
3	..... menší než (<)	
4	..... více než (>)	
5	..... menší nebo rovno (<=)	
6	..... větší nebo rovno (>=)	

### Výstup

<b>Y</b>	Logický výstupní signál	Bool
----------	-------------------------	------

## RTOI – Konverze reálného čísla na celé číslo

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok **RTOI** převádí reálné číslo  $r$  na celé číslo  $i$  se znaménkem. Výsledná zaokrouhlená hodnota je určena vztahem:

$$i = \begin{cases} -2147483648, & \text{pro } r \leq -2147483648.0, \\ \text{round}(r), & \text{pro } -2147483648.0 < r \leq 2147483647.0, \\ 2147483647, & \text{pro } r > 2147483647.0, \end{cases}$$

kde  $\text{round}(r)$  je zaokrouhlení na nejbližší celé číslo. Čísla ve tvaru  $n + 0.5$  ( $n$  celé) zaokrouhluje k číslu s vyšší absolutní hodnotou, např.  $\text{round}(1.5) = 2$ ,  $\text{round}(-2.5) = -3$ .

Poznamenejme, že čísla  $-2147483648$  a  $2147483647$  odpovídají po řadě nejmenšímu a největšímu číslu se znaménkem zobrazitelným ve formátu s 32 bity (v jazyku C zapsanými v šestnáctkové soustavě jako  $0x7FFFFFFF$  a  $0x80000000$ ). Tyto limity platí, pokud parametr **vtype** má výchozí hodnotu 4 (long). Jinak platí limity pro příslušný datový typ (viz níže).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

**r**                      Analogový vstupní signál    Double (F64)

### Parametr

**vtype**                      Číselný typ     $\odot 4$  Long (I32)

2 ..... Byte (U8)

3 ..... Short (I16)

4 ..... Long (I32)

5 ..... Word (U16)

6 ..... DWord (U32)

10 .... Large (I64)

**SAT**                      Kontrola přetečení     $\odot \text{on}$  Bool

off ... přetečení se nekontroluje

on ... přetečení se kontroluje



## Výstup

i

Zaokrouhlený a zkonvertovaný vstupní signál

Long (I32)

**SQR – Druhá mocnina**

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok SQR počítá druhou mocninu analogového vstupu  $u$ . Výstup je pak

$$y = u^2.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Výstup

$y$	Druhá mocnina vstupního signálu	Double (F64)
-----	---------------------------------	--------------

## SQRT – Druhá odmocnina

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SQRT** počítá druhou odmocninu analogového vstupu **u**. Výstup je pak

$$y = \sqrt{u}.$$

V případě  $u < 0$  je aktivován chybový výstup  $E = \text{on}$  a výstup  $y$  je nastaven na hodnotu parametru **yerr**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<b>u</b>	Analogový vstupní signál	Double (F64)
----------	--------------------------	--------------

Parametr

<b>yerr</b>	Náhradní hodnota pro případ chyby	⊖1.0 Double (F64)
-------------	-----------------------------------	-------------------

Výstup

<b>y</b>	Odmocnina ze vstupní hodnoty	Double (F64)
<b>E</b>	Příznak chyby	Bool
	<b>off</b> ... bez chyby	
	<b>on</b> ... nastala chyba	

## SUB – Odčítání dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SUB počítá rozdíl dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 - u2.$$

Pro sčítání a odečítání více signálů můžete použít blok [ADDOCT](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	Analogový vstupní signál	Double (F64)
u2	Analogový vstupní signál	Double (F64)

Výstup

y	Rozdíl mezi vstupními signály	Double (F64)
---	-------------------------------	--------------

## UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok UTOI umožňuje převod neznaménkového celého čísla na znaménkové číslo v reprezentaci dvojkového doplňku, což je běžná reprezentace používaná v procesorech. Například, v 8-bitové reprezentaci je číslo -1 reprezentováno jako 255, a v 16-bitové reprezentaci jako 65535. Rozlišení, kolikabitová reprezentace se má použít, určuje parametr `bits`.

Primární využití tohoto bloku je v situacích, kdy hodnota získaná z ovladače obsahuje více signálů, které se izolují vymaskováním (typicky pomocí bloků [INTSM](#) nebo [BITOP](#)). Výsledkem tohoto vymaskování je vždy neznaménkové (kladné) číslo. Pokud však má být signál interpretován jako číslo se znaménkem, je pro získání správné hodnoty nutné použít tento blok.

Vzhledem k tomu, že procesory se mohou lišit ve způsobu, jakým ukládají vícebajtová čísla (nejčastěji se setkáváme s tzv. little-endian formátem, kde na nižší adrese najdeme méně významný bajt, existují však i procesory s big-endian formátem, kde je to právě naopak), blok UTOI nabízí možnost prohození pořadí bajtů, pokud to již není řešeno na úrovni ovladače. Pro tuto úpravu slouží parametr `SWAP`.

Upozornění: Prohození pořadí bajtů (nastavením `SWAP=on`) typicky řeší problémy s rozdílným uspořádáním bajtů v procesoru pouze pro hodnoty `bits=16` nebo `bits=32`.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>u</code>	Vstupní signál bez znaménka	↓-9.22337E+18 ↑9.22337E+18	Large (I64)
----------------	-----------------------------	----------------------------	-------------

### Parametr

<code>bits</code>	Počet platných bitů ve vstupním signálu	↓2 ↑64 ○16	Long (I32)
<code>SWAP</code>	Otočení pořadí bajtů vstupu		Bool

### Výstup

<code>i</code>	Zkonvertovaný (se znaménkem) vstupní signál		Large (I64)
----------------	---	--	-------------



## Kapitola 5

# ANALOG – Zpracování analogových signálů

### Obsah

---

ABSR0T – Víceotáčkové rozšíření čidla polohy . . . . .	129
ASW – Přepínač s automatickou volbou vstupu . . . . .	131
AVG – Filtr: vlečný průměr . . . . .	133
AVS – Rozběhová jednotka . . . . .	134
AVSI – Hladká interpolace trajektorie . . . . .	135
BPF – Filtr: pásmová propust' . . . . .	138
CMP – Komparátor s hysterezí . . . . .	139
CNDR – Kompenzátor složité nelinearity . . . . .	140
DEL – Dopravní zpoždění s inicializací . . . . .	142
DELM – Dopravní zpoždění . . . . .	143
DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků . . . . .	144
EVAR – Vlečná střední hodnota a směrodatná odchylka . . . . .	145
INTE – Řízený integrátor . . . . .	146
KDER – Derivace a filtrace vstupního signálu . . . . .	148
LPF – Filtr: dolní propust' . . . . .	150
MINMAX – Vlečné minimum a maximum . . . . .	152
NSCL – Kompenzátor jednoduché nelinearity . . . . .	153
OSD – Jednokrokové zpoždění . . . . .	154
RDFT – Vlečná diskrétní Fourierova transformace . . . . .	155
RLIM – Omezovač strmosti . . . . .	157
S10F2 – Výběr jednoho ze dvou analogových vstupů . . . . .	158
SAI – Zabezpečený analogový vstup . . . . .	161
SEL – Selektor analogového signálu . . . . .	164
SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu . . . . .	166

SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot . . . .	168
SHLD – Vzorkovač (sample and hold) . . . . .	170
SINT – Jednoduchý integrátor . . . . .	172
SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů . . . .	173
SSW – Jednoduchý přepínač . . . . .	175
SWR – Přepínač s rampovou funkcí . . . . .	176
VDEL – Dopravní zpoždění s proměnnou délkou . . . . .	177
ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací . . . .	178

---

Knihovna představuje univerzální škálu funkčních bloků určených pro aplikace řízení a zpracování signálů. Obsahuje bloky jako **ASW**, **AVG**, **BPF** a **DEL**, které nabízejí funkce od manipulace se signálem a průměrování po filtrování a složité podmíněné operace, čímž pokrývají široké spektrum požadavků a scénářů.



## ABSROT – Víceotáčkové rozšíření čidla polohy

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok ABSROT se typicky používá v případech, kdy máme na nějaké hřídeli absolutní čidlo úhlu natočení v rozsahu např.  $5^\circ$  až  $355^\circ$  (popř.  $-175^\circ$  až  $+175^\circ$ ), ale potřebujeme řídit pohyb o několik otáček. Blok předpokládá spojitý signál, takže při přechodu z například  $355^\circ$  na  $5^\circ$  předpokládá, že nastala další otáčka a úhel je ve skutečnosti  $365^\circ$ .

Protože v případě dlouhodobého otáčení jedním směrem by došlo k ztrátě přesnosti, je možné vstupem R1 nastavit výstup y zpět do základního intervalu. Pokud je nastaven příznak RESR = on, dojde i k vynulování čítače otáček irev. V každém případě je však potřeba zároveň resetovat všechny související signály (např. signál sp připojeného regulátoru).

Výstup MPI (mid-point indicator) detekuje střední polohu čidla, což může být vhodný okamžik k resetování bloku. Výstup OLI (off-limits indicator) informuje o tom, že čidlo natočení je v tzv. mrtvém úhlu, kdy neposkytuje platná data.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

u	Signál z absolutního snímače polohy	Double (F64)
R1	Reset bloku	Bool

### Parametr

lolim	Dolní mez pro údaj z čidla	$\ominus 3.14159265$	Double (F64)
hilim	Horní mez pro údaj z čidla	$\odot 3.14159265$	Double (F64)
tol	Rozsah pro indikaci středové pozice	$\odot 0.5$	Double (F64)
hys	Hystereze pro indikaci středové pozice		Double (F64)
RESR	Příznak pro resetování počítadla otáček		Bool
	off ... resetovat pouze vypočtenou polohu		
	on .... vynulovat i počítadlo otáček		

### Výstup

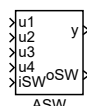
y	Vypočtená poloha	Double (F64)
irev	Počet otáček	Long (I32)

MPI	Indikátor středové polohy	Bool
OLI	Indikátor polohy mimo rozsah senzoru	Bool

## ASW – Přepínač s automatickou volbou vstupu

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok ASW ukládá na výstup  $y$  hodnotu jednoho ze vstupů vstup  $u_1, \dots, u_4$  nebo jeden z parametrů  $p_1, \dots, p_4$ . Pokud je na vstupu  $iSW$  jedna z hodnot  $\{1, 2, 3, 4\}$ , je na výstupu  $y$  hodnota příslušného vstupu. Pokud je na vstupu  $iSW$  jedna z hodnot  $\{-1, -2, -3, -4\}$ , je na výstupu  $y$  hodnota příslušného parametru (tj. pro  $iSW = -1$  je na výstupu  $y$  hodnota  $p_1$ , pro  $iSW = 3$  je na výstupu  $y$  hodnota  $u_3$  atd.). Pokud je na vstupu  $iSW$  jiná hodnota (tj.  $iSW = 0$  nebo  $iSW < -4$  nebo  $iSW > 4$ ), je na výstupu  $y$  hodnota toho ze vstupů  $u_1, \dots, u_4$  nebo parametrů  $p_1, \dots, p_4$ , který se naposledy změnil. Pokud se změní více hodnot najednou, pak se použije hodnota podle následujícího pořadí  $p_4, p_3, p_2, p_1, u_4, u_3, u_2, u_1$ . Hodnota se považuje za změněnou, pokud se změnila o více než udává parametr  $\delta$  od minulé detekce změny na příslušném vstupu resp. parametru (tj. změny se uvažují integrálně nikoliv diferenciálně od minulého vzorku). Ve všech režimech je na výstupu  $oSW$  číslo vstupu (resp. číslo parametru, pokud je hodnota záporná), který se použil pro generování výstupu  $y$ .

Blok ASW má dále tu speciální vlastnost, že nová hodnota  $y$  se kopíruje na parametry  $p_1, \dots, p_4$  (stejná vlastnost je i u bloků PARR, PARI, PARB). To má za následek, že všechny externí nástroje jako hodnotu všech těchto vstupů přečtou stejnou hodnotu  $y$ . To se hodí zejména v nadřazených systémech, které používají metodu nastav a sleduj (např. "potenciometr" v Iconics Genesis). Tato vlastnost není implementována ve verzi bloku ASW pro Simulink, protože tam není možnost používat externí programy pro čtení vstupu bloku.

**POZOR!** Pokud je blok zařazen ve schématu v nějaké smyčce, může se stát, že jeden ze vstupů  $u_1, \dots, u_4$  je o krok zpožděn, čímž se zdánlivě ignoruje priorita (výstup  $oSW$  pak zcela nepochopitelně signalizuje, že poslední změna nastala na tomto o krok zpožděném vstupu). Dalším důsledkem tohoto stavu je, že externí nástroje na zpožděném vstupu nezobrazují hodnotu  $y$ . Takovému chování lze zabránit vhodným použitím bloků LPBRK (např. za oba výstupy).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

$u_1$	První analogový vstup bloku	Double (F64)
$u_2$	Druhý analogový vstup bloku	Double (F64)

u3	Třetí analogový vstup bloku	Double (F64)
u4	Čtvrtý analogový vstup bloku	Double (F64)
iSW	Volba aktivního signálu nebo parametru	Long (I32)

### Parametr

delta	Práh pro detekci změny	⊙1e-06	Double (F64)
p1	Parametr p1 pro výběr		Double (F64)
p2	Parametr p2 pro výběr		Double (F64)
p3	Parametr p3 pro výběr		Double (F64)
p4	Parametr p4 pro výběr		Double (F64)

### Výstup

y	Zvolený signál nebo parametr	Double (F64)
oSW	Identifikátor použitého vstupu nebo parametru	Long (I32)

## AVG – Filtr: vlečný průměr

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **AVG** počítá vlečný průměr z posledních  $n$  vzorků,  $n < N$  ( $N$  závisí na implementaci), podle vztahu

$$y_k = \frac{1}{n}(u_k + u_{k-1} + \dots + u_{k-n+1}).$$

Není-li ještě k dispozici všech  $n$  vzorků (po startu algoritmu), je výstup  $y$  nastaven na hodnotu průměru ze všech dostupných vzorků.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u$	Vstupní signál filtru	Double (F64)
-----	-----------------------	--------------

### Parametr

$n$	Počet vzorků pro průměrování	↓1 ↑10000000 ⊙10	Long (I32)
$n_{\max}$	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙100	Long (I32)

### Výstup

$y$	Filtrovaný výstupní signál	Double (F64)
-----	----------------------------	--------------

## AVS – Rozběhová jednotka

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok *AVS* generuje časově optimální trajektorii pohybu z klidové polohy 0 do klidové polohy *sm* při omezení *am* na maximální zrychlení, *dm* na maximální zpomalení a *vm* na maximální rychlost. Při náběžné hraně vstupu *SET* (*off*→*on*) se provede inicializace (výpočet trajektorie) pro aktuální vstupy *am*, *dm*, *vm* a *sm*. Před první inicializací a po dobu inicializace má výstup *RDY* hodnotu *off*, potom *on*. Při náběžné hraně vstupu *START* (*off*→*on*) se spustí generování trajektorie pohybu na výstupech *a*, *v*, *s*, *tt*, přičemž tyto výstupy mají po řadě význam zrychlení, rychlosti, polohy a času. Po dobu generování trajektorie má výstup *BSY* hodnotu *on*, jinak *off*.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<i>START</i>	Spouštěcí signál (náběžná hrana)	Bool
<i>SET</i>	Inicializace/výpočet trajektorie	Bool
<i>am</i>	Maximální povolené zrychlení [ $m/s^2$ ]	Double (F64)
<i>dm</i>	Maximální povolené zpomalení [ $m/s^2$ ]	Double (F64)
<i>vm</i>	Maximální povolená rychlost [ $m/s$ ]	Double (F64)
<i>sm</i>	Žádaná konečná poloha [ $m$ ]	Double (F64)

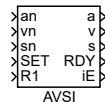
### Výstup

<i>a</i>	Zrychlení [ $m/s^2$ ]	Double (F64)
<i>v</i>	Rychlost [ $m/s$ ]	Double (F64)
<i>s</i>	Poloha [ $m$ ]	Double (F64)
<i>tt</i>	Čas [ $s$ ]	Double (F64)
<i>RDY</i>	Příznak platnosti/připravenosti výstupu	Bool
<i>BSY</i>	Příznak probíhající operace	Bool

## AVSI – Hladká interpolace trajektorie

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok **AVSI** - Akcelerace ( $A$ ), Rychlost ( $V$ ), Vzdálenost ( $S$ ) Interpolace ( $I$ ) - je navržen pro interpolaci signálů, zejména v aplikacích řízení pohybu. Jeho hlavní účel spočívá v generování hladkých sekvencí polohy (vzdálenosti), rychlosti a zrychlení na základě diskrétních vstupních hodnot. Tento blok je inspirován funkcionalitou z [RM\\_AxisSpline](#) a nabízí podobné interpolační metody.

Blok **AVSI** přijímá na svých vstupech **sn** (polohu), **vn** (rychlost) a **an** (zrychlení), které jsou generovány externí funkcí (mimo tento blok) s určitou periodou definovanou parametrem **RemTs**. Hodnoty se na vstupu bloku aktualizují na náběžné hraně signálu **SET**.

Interpolace mezi jednotlivými vstupy se provádí s cílem vytvořit plynulé přechody a zajistit kontinuální řízení pohybu nebo jiné aplikace vyžadující regulaci signálu a jeho derivací. Blok podporuje různé interpolační metody určené parametrem **Mode**, které odpovídají možnostem v [RM\\_AxisSpline](#). Podporované metody jsou:

- 1: **linear**: Poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0 (tj. rychlost je po částech konstantní funkce se skoky).
- 2: **cubic spline**: Poloha je polynom 3. řádu, vypočtený na základě polohy a rychlosti na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 3: **quintic spline**: Poloha je polynom 5. řádu, vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 4: **cubic approximation (B-spline)**: Poloha je polynom 3. řádu, vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 5: **quintic approximation (B-spline)**: Poloha je polynom 5. řádu, vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.

- 6: **all linear**: Poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti.
- 7: **all cubic**: Poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy.
- 8: *rezervováno pro pozdější použití.*
- 9: *rezervováno pro pozdější použití.*

Vzhledem k principu činnosti blok AVSI zavádí zpoždění signálu, kde aktivní generování hodnot začíná až po dvou kompletních periodách **RemTs** od první náběžné hrany signálu **SET**. Pro B-spline interpolační metody je potřeba větší množství vzorků pro zahájení interpolace.

Blok AVSI je primárně určen pro aplikace řízení pohybu, ale lze jej použít i pro jiné typy signálů a jejich derivace. Jeho implementace umožňuje efektivnější a plynulejší přechody mezi jednotlivými stavovými hodnotami bez nutnosti složitějšího externího řízení.

Při použití je důležité správně nastavit periodu **RemTs** odpovídající generátoru vstupních hodnot a zvolit vhodný **Mode** pro požadovaný typ interpolace.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<b>an</b>	Zrychlení v příštím (velkém) tiku [unit/s <sup>2</sup> ]	Double (F64)
<b>vn</b>	Rychlost v příštím (velkém) tiku [unit/s]	Double (F64)
<b>sn</b>	Poloha v příštím (velkém) tiku [unit]	Double (F64)
<b>SET</b>	Vstupy jsou použity při náběžné hraně	Bool
<b>R1</b>	Reset bloku	Bool
<b>dm</b>	Maximální povolené zpomalení při chybě[m/s <sup>2</sup> ]	Double (F64)

## Parametr

<b>RemTs</b>	Perioda pomalejšího (vzdáleného) signálu	⊙0.0	Double (F64)
<b>Mode</b>	Algoritmus pro interpolaci	⊙9	Long (I32)
	1 ..... lineární		
	2 ..... polynom 3. řádu		
	3 ..... polynom 5. řádu		
	4 ..... aproximace polynomem 3.řádu (B-spline)		
	5 ..... aproximace polynomem 5.řádu (B-spline)		
	6 ..... vše lineární		
	7 ..... vše 3.řád		
	8 ..... —		
	9 ..... —		



## Výstup

a	Zrychlení [unit/s <sup>2</sup> ]	Double (F64)
v	Rychlost [unit/s]	Double (F64)
s	Poloha [unit]	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool
iE	Kód chyby	Bool

## BPF – Filtr: pásmová propušť

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok BPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{2\xi as}{a^2 s^2 + 2\xi as + 1},$$

kde  $a$  a  $\xi$  jsou po řadě parametry bloku **fm** a **xi**. Parametr **fm** určuje střed frekvenčního pásma propustnosti a **xi** je součinitel relativního tlumení.

Je-li **ISSF = on**, potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu **u**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u</b>	Vstupní signál filtru	Double (F64)
<b>R1</b>	Reset bloku	Bool
<b>HLD</b>	Pozastavení	Bool

### Parametr

<b>fm</b>	Střed pásma propustnosti [Hz]	⊙1.0	Double (F64)
<b>xi</b>	Součinitel relativního tlumení	⊙0.707	Double (F64)
<b>ISSF</b>	Ustálený stav při spuštění		Bool
	off ... nulový počáteční stav		
	on .... ustálený počáteční stav		

### Výstup

<b>y</b>	Filtrovaný výstupní signál	Double (F64)
----------	----------------------------	--------------

## CMP – Komparátor s hysterezí

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok CMP provádí komparaci vstupu **u1** a **u2** s hysterezí **h** podle následujících vztahů:

$$\begin{aligned} Y_{-1} &= 0, \\ Y_k &= \mathit{hyst}(e_k), \quad k = 0, 1, 2, \dots \end{aligned}$$

kde

$$e_k = u1_k - u2_k$$

a

$$\mathit{hyst}(e_k) = \begin{cases} 0 & \text{pro } e_k \leq -h \\ Y_{k-1} & \text{pro } e_k \in (-h, h) \\ 1 & \text{pro } e_k \geq h \quad (e_k > h \text{ pro } h = 0) \end{cases}$$

Indexované proměnné odpovídají hodnotám dané veličiny v cyklu, který udává index  $k$ , tzn.  $Y_{k-1}$  značí hodnotu výstupu v minulém kroku/cyklu. Hodnota  $Y_{-1}$  je použita pouze jednou při inicializaci bloku ( $k = 0$ ), pokud je rozdíl vstupních signálů nepřekročí mez hystereze.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u1</b>	První analogový vstup bloku	Double (F64)
<b>u2</b>	Druhý analogový vstup bloku	Double (F64)

### Parametr

<b>hys</b>	Hystereze	↓0.0 ⊙0.5 Double (F64)
------------	-----------	------------------------

### Výstup

<b>Y</b>	Logický výstupní signál	Bool
----------	-------------------------	------

## CNDR – Kompenzátor složité nelinearity

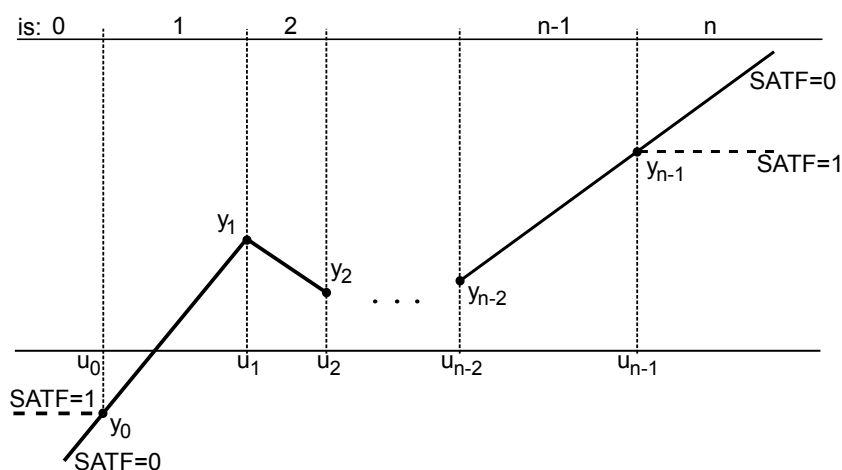
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok CNDR je určen pro kompenzaci složitých nelinearit pomocí po částech lineární transformace zobrazené na níže uvedeném obrázku.



V této souvislosti je důležité upozornit, že v případech  $u < u_0$  a  $u > u_{n-1}$  je výstup definován v závislosti na parametru SATF.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

$u$  Analogový vstupní signál Double (F64)

Parametr

$n_{max}$	Rezervovaná paměť pro pole	↓4 ⊙10	Long (I32)
SATF	Saturace	⊙on	Bool
	off ... signál není omezen		
	on ... saturační meze jsou aktivní		
$u_p$	Vektor rostoucích $u$ souřadnic uzlů		Double (F64)
		⊙[0.0 3.9 3.9 9.0 14.5 20.0]	
$y_p$	Vektor $y$ souřadnic uzlů	⊙[0.0 0.0 15.8 38.4 72.0 115.0]	Double (F64)

## Výstup

y	Analogový výstupní signál	Double (F64)
is	Sektor nelinearity odpovídající vstupu u	Long (I32)

## DEL – Dopravní zpoždění s inicializací

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok DEL realizuje zpoždění vstupního signálu  $u$  o  $n$  vzorků, tj.

$$y_k = u_{k-n}.$$

Příslušné časové zpoždění je  $n \cdot T_S$ , kde  $T_S$  je perioda spouštění bloku.

Jestliže po spuštění nebo restartu ( $R1$ :  $\text{off} \rightarrow \text{on} \rightarrow \text{off}$ ) dosud není zapamatovaných  $n$  minulých vzorků ( $RDY = \text{off}$ ), potom

$$y_k = y_0,$$

kde  $y_0$  je inicializační vstup bloku.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

$u$	Analogový vstupní signál	Double (F64)
$R1$	Reset bloku	Bool
$y_0$	Počáteční hodnota výstupu	Double (F64)

### Parametr

$n$	Zpoždění (počet vzorků)	$\downarrow 0 \uparrow 10000000 \odot 10$	Long (I32)
$n_{\max}$	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 10000000 \odot 100$	Long (I32)

### Výstup

$y$	Zpožděný vstupní signál	Double (F64)
$RDY$	Příznak platnosti/připravenosti výstupu	Bool

## DELM – Dopravní zpoždění

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok DELM realizuje časové zpoždění vstupního signálu  $u$  o čas, který vznikne zaokrouhlením parametru `del` na nejbližší celočíselný násobek periody  $T_S$  spouštění bloku. Po spuštění bloku do času `del` je výstup  $y = 0$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>u</code>	Analogový vstupní signál	Double (F64)
----------------	--------------------------	--------------

### Parametr

<code>del</code>	Časové zpoždění [s]	⊙1.0	Double (F64)
<code>nmax</code>	Rezervovaná paměť pro pole	↓10 ↑10000000	⊙100 Long (I32)

### Výstup

<code>y</code>	Zpožděný vstupní signál	Double (F64)
----------------	-------------------------	--------------

DER – Derivace, filtrace a predikce z posledních  $n+1$  vzorků

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok DER prokládá posledních  $n + 1$  vzorků ( $n \leq N - 1$ ,  $N$  závisí na implementaci) vstupního signálu  $u$  přímkou  $y = at + b$  metodou nejmenších čtverců. Počátek časové osy je v každém kroku umístěn do aktuálního okamžiku vzorkování vstupu  $u$ .

Ze získaných parametrů přímky  $a$  a  $b$  se počítají v případě  $RUN = \text{on}$  výstupy  $y$  a  $z$  podle vztahů:

$$\begin{aligned} \text{Derivace: } & y = a \\ \text{Filtrace: } & z = b, \text{ pro } t_p = 0 \\ \text{Predikce: } & z = at_p + b, \text{ pro } t_p > 0 \\ \text{Postdikce: } & z = at_p + b, \text{ pro } t_p < 0 \end{aligned}$$

Je-li  $RUN = \text{off}$  nebo blok nemá k dispozici posledních  $n + 1$  vzorků vstupního signálu ( $RDY = \text{off}$ ), potom  $y = 0$ ,  $z = u$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$u$	Analogový výstupní signál	Double (F64)
$RUN$	Povolení běhu algoritmu off ... sledování on ... filtrace	Bool
$t_p$	Časový okamžik pro predikci/filtraci	Double (F64)

## Parametr

$n$	Počet vzorků pro interpolaci	↓1 ↑10000000 ⊙10	Long (I32)
$nmax$	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙100	Long (I32)

## Výstup

$y$	Odhad derivace vstupního signálu	Double (F64)
$z$	Predikovaný/filtrovaný výstupní signál	Double (F64)
$RDY$	Příznak platnosti/přípravenosti výstupu	Bool



## EVAR – Vlečná střední hodnota a směrodatná odchylka

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EVAR** počítá střední hodnotu  $\mu$  ( $\mu$ ) a směrodatnou odchylku  $\sigma$  ( $\sigma$ ) z posledních  $n$  vzorků vstupního signálu  $u$  podle vztahů

$$\mu_k = \frac{1}{n} \sum_{i=0}^{n-1} u_{k-i}$$

$$\sigma_k = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} u_{k-i}^2 - \mu_k^2}$$

kde  $k$  značí aktuální okamžik vzorkování.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Parametr

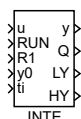
$n$	Počet vzorků pro výpočet statistických ukazatelů	Long (I32)
		↓2 ↑10000000 ⊙100
$n_{\max}$	Rezervovaná paměť pro pole	Long (I32)
		↓10 ↑10000000 ⊙200

Výstup

$\mu$	Střední hodnota	Double (F64)
$\sigma$	Směrodatná odchylka	Double (F64)

## INTE – Řízený integrátor

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok INTE realizuje řízený integrátor s proměnnou integrační časovou konstantou  $t_i$  a indikací dvou úrovní výstupu  $y_{\min}$  a  $y_{\max}$ . Je-li  $RUN = \text{on}$  a  $R1 = \text{off}$ , potom

$$y(t) = \frac{1}{T_i} \int_0^t u(\tau) d\tau + C,$$

kde hodnota  $C = y_0$ . Je-li  $RUN = \text{off}$  a  $R1 = \text{off}$ , je výstup  $y$  zmrazen na jeho poslední hodnotu před sestupnou hranou vstupu  $RUN$ . Je-li  $R1 = \text{on}$ , potom je výstup  $y$  resetován na počáteční hodnotu  $y_0$ . Integrace se provádí lichoběžníkovou metodou podle vztahu

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde  $T_S$  je perioda spouštění bloku. Pokud je  $T_i = 0$ , místo integrace se provádí sumace podle vztahu

$$y_k = y_{k-1} + u_k.$$

Pro  $T_i < 0$  je chování nedefinované.

Pro integraci je také možno použít blok [SINT](#), jehož jednodušší struktura a funkčnost může být pro základní úlohy dostačující.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

$u$	Analogový vstupní signál	Double (F64)
$RUN$	Povolení běhu algoritmu off ... integrace je pozastavena on ... integrace probíhá	Bool
$R1$	Reset bloku	Bool
$y_0$	Počáteční hodnota výstupu	Double (F64)
$t_i$	Integrační časová konstanta	Double (F64)

## Parametr

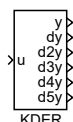
ymin	Definice spodní úrovně	⊖-1.0	Double (F64)
ymax	Definice horní úrovně	⊕1.0	Double (F64)
SAT	Omezení výstupu, když je dosažena krajní úroveň		Bool

## Výstup

y	Výstup integrátoru		Double (F64)
Q	Příznak probíhající integrace		Bool
LY	Příznak dosažení spodní meze		Bool
HY	Příznak dosažení horní meze		Bool

## KDER – Derivace a filtrace vstupního signálu

Symbol bloku

Licence: [ADVANCED](#)

## Popis funkce

Blok KDER je speciálně navržený Kalmanův filtr řádu `norder` tak, aby poskytoval odhady časových derivací řádu 0 až `norder - 1` lokálně polynomiálních signálů, jejichž měření je zatíženo šumem. Blok je možné využít pro odhad derivací téměř libovolného vstupního signálu  $u = u_0(t) + v(t)$  za předpokladu, že užitečný signál  $u_0(t)$  a šum  $v(t)$  mají odlišné frekvenční spektrum.

Blok se nastavuje pouze pomocí dvou parametrů `pbeta` a `norder`. Parametr `pbeta` je závislý na vzorkovací periodě  $T_S$ , frekvenčních vlastnostech vstupního signálu  $u$  a rovněž frekvenčních vlastnostech a úrovni obsaženého šumu. Platí pro něj přibližný vztah  $\text{pbeta} \approx T_S \omega_0$ . Pro správnou funkci bloku KDER by se frekvenční spektrum vstupního signálu  $u$  mělo nacházet hluboko pod zlomovou frekvencí filtru  $\omega_0$ . Naopak frekvenční spektrum šumů by mělo být co možná nejdále od frekvence  $\omega_0$ . Pro vyšší potlačení šumů je nutné volit nižší zlomovou frekvenci  $\omega_0$  a tím i parametr `pbeta`.

Druhý parametr `norder` je nutné volit převážně s ohledem na řád odhadovaných derivací. Ve většině případů by mělo stačit použít standardní hodnotu pro 3. řád. Vyšší hodnoty řádu derivačního filtru poskytují o něco lepší odhady derivací nepolynomiálních vstupních signálů za cenu delší doby vysledování (naladění) a vyšších výpočetních nároků.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>u</code>	Vstupní signál filtru	Double (F64)
----------------	-----------------------	--------------

## Parametr

<code>norder</code>	Řád derivačního filtru	↓2 ↑10 ⊙3	Long (I32)
<code>pbeta</code>	Šířka pásma derivačního filtru	↓0.0 ⊙0.1	Double (F64)

## Výstup

<code>y</code>	Filtrovaný vstupní signál	Double (F64)
<code>dy</code>	Odhad 1. derivace vstupního signálu	Double (F64)

d2y	Odhad 2. derivace vstupního signálu	Double (F64)
d3y	Odhad 3. derivace vstupního signálu	Double (F64)
d4y	Odhad 4. derivace vstupního signálu	Double (F64)
d5y	Odhad 5. derivace vstupního signálu	Double (F64)

## LPF – Filtr: dolní propuř

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok LPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{1}{a^2 s^2 + 2\xi a s + 1},$$

kde

$$a = \frac{\sqrt{\sqrt{2}\sqrt{2\xi^4 - 2\xi^2 + 1} - 2\xi^2 + 1}}{2\pi f_b}$$

a  $f_b$  a  $\xi = xi$  jsou parametry bloku. Frekvence  $f_b$  [Hz] určuje šířku pásma filtru a parametr  $xi$  součinitel relativního tlumení filtru. Útlum na frekvenci  $f_b$  je 3 dB, na  $10 \cdot f_b$  přibližně 40 dB. Pro správnou funkci filtru musí platit  $f_b < \frac{1}{10T_s}$ , kde  $T_s$  je perioda spouštění bloku. Doporučená hodnota pro Butterworthův filtr je  $xi = 0,71$  a pro Besselův filtr  $xi = 0,87$ .

Je-li  $ISSF = on$ , potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu  $u$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u$	Vstupní signál filtru	Double (F64)
$R1$	Reset bloku	Bool
$HLD$	Pozastavení	Bool

### Parametr

$f_b$	Šířka pásma [Hz]	⊙1.0	Double (F64)
$xi$	Součinitel relativního tlumení	⊙0.707	Double (F64)
$ISSF$	Ustálený stav při spuštění		Bool
	$off$ ... nulový počáteční stav		
	$on$ ... ustálený počáteční stav		

Výstup

y

Filtrovaný výstupní signál

Double (F64)

## MINMAX – Vlečné minimum a maximum

Symbol bloku

Licence: **STANDARD**



### Popis funkce

Blok MINMAX vyhodnocuje minimum a maximum z posledních  $n$  vzorků vstupního signálu  $u$ . Pokud není k dispozici  $n$  vzorků, je nastaveno  $RDY = \text{off}$  a minimum a maximum se hledá mezi dostupnými vzorky.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

$u$	Analogový vstupní signál	Double (F64)
$R1$	Reset bloku	Bool

### Parametr

$n$	Počet prvků (délka bufferu)	↓1 ↑10000000 ⊕100	Long (I32)
$nmax$	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊕200	Long (I32)

### Výstup

$ymin$	Nalezená minimální hodnota	Double (F64)
$ymax$	Nalezená maximální hodnota	Double (F64)
$RDY$	Příznak platnosti/připravenosti výstupu	Bool



## NSCL – Kompenzátor jednoduché nelinearity

Symbol bloku

Licence: [STANDARD](#)

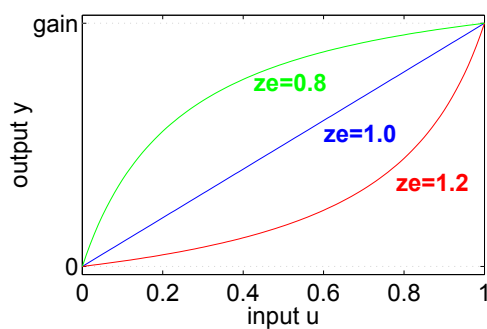


Popis funkce

Blok NSCL kompenzuje v praxi často se vyskytující nelinearity (např. nelinearitu servoventilu) pomocí vztahu

$$y = \text{gain} \frac{u}{ze + (1 - ze)u},$$

kde  $\text{gain}$  a  $ze$  jsou parametry bloku. Volbou  $ze$  v intervalu  $(0, 1)$  obdržíme konkávní transformaci, zatímco je-li  $ze > 1$ , dostaneme transformaci konvexní.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u$  Analogový vstupní signál Double (F64)

Parametr

$\text{gain}$  Zesílení  $\odot 1.0$  Double (F64)  
 $ze$  Tvarovací parametr  $\odot 1.0$  Double (F64)

Výstup

$y$  Analogový výstupní signál Double (F64)

## OSD – Jednokrokové zpoždění

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **OSD** realizuje jednokrokové zpoždění vstupního signálu **u**. Délka zpoždění (vyjádřena v sekundách) je dána periodou tasku (blíže viz blok [EXEC](#)).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u</b>	Analogový vstupní signál	<b>Any</b>
----------	--------------------------	------------

### Parametr

<b>LB</b>	Funkce rozpojení smyčky	<b>Bool</b>
-----------	-------------------------	-------------

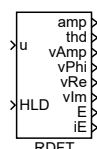
### Výstup

<b>y</b>	Analogový výstupní signál	<b>Any</b>
----------	---------------------------	------------

## RDFT – Vlečná diskrétní Fourierova transformace

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok RDFT počítá diskrétní Fourierovu transformaci vstupního signálu pro základní frekvenci `freq` (a případně několik dalších) z posledních `m` vzorků vstupního signálu `u`, kde  $m = nper / freq / T_S$ , tj. z časového okna o délce odpovídající `nper` periodám základní frekvence.

Pokud je `nharm > 0`, je počet vyčíslovaných vyšších harmonických frekvencí dán právě tímto parametrem. Pokud je `nharm = 0`, další vyčíslované frekvence určuje vektorový parametr `freq2`.

Pro každou frekvenci se vyčísluje amplituda (výstup `vAmp`), fáze (výstup `vPhi`), reálná/kosinová složka (výstup `vRe`) a imaginární/sinová složka (výstup `vIm`). Výstupy bloku jsou vektorové, takže obsahují příslušné hodnoty pro všechny analyzované frekvence. Hodnoty pro jednotlivé frekvence se získají pomocí bloků `VTOR`. Výstup `thd` udává celkové harmonické zkreslení, tj. podíl základní a vyšších harmonických frekvencí (jen pokud `nharm ≥ 1`).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>u</code>	Analogový vstupní signál	Double (F64)
<code>HLD</code>	Pozastavení	Bool

### Parametr

<code>freq</code>	Základní frekvence	↓1e-09 ↑1e+09 ⊙1.0	Double (F64)
<code>nper</code>	Počet period signálu na kterých provádět výpočet	↓1 ↑10000 ⊙10	Long (I32)
<code>nharm</code>	Počet monitorovaných harmonických frekvencí	↓0 ↑16 ⊙3	Long (I32)
<code>ifrunite</code>	Jednotky pro frekvenci	⊙1	Long (I32)
	1 ..... Hz		
	2 ..... rad/s		
<code>iphunit</code>	Jednotky pro fázový posun	⊙1	Long (I32)
	1 ..... stupně		
	2 ..... radiány		

<code>nmax</code>	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊖8192	Long (I32)
<code>freq2</code>	Vektor uživatelem definovaných frekvencí	⊖[2.0 3.0 4.0]	Double (F64)

### Výstup

<code>amp</code>	Amplituda základní frekvence	Double (F64)
<code>thd</code>	Celkové harmonické zkreslení	Double (F64)
<code>vAmp</code>	Vektor amplitud pro zadané frekvence	Reference
<code>vPhi</code>	Vektor fázových posunů pro zadané frekvence	Reference
<code>vRe</code>	Vektor reálných částí pro zadané frekvence	Reference
<code>vIm</code>	Vektor imaginárních částí pro zadané frekvence	Reference
<code>E</code>	Příznak chyby	Bool
<code>iE</code>	Kód chyby	Error

## RLIM – Omezovač strmosti

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **RLIM** kopíruje vstup  $u$  na výstup  $y$ , avšak maximální dovolená rychlost změny signálu je omezena. Omezení jsou definována časovými konstantami  $t_p$  a  $t_n$  podle následujících vztahů:

$$\begin{aligned} \text{maximální nárůst za sekundu:} & \quad 1/t_p \\ \text{maximální pokles za sekundu:} & \quad -1/t_n \end{aligned}$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u$	Vstupní signál filtru	Double (F64)
-----	-----------------------	--------------

### Parametr

$t_p$	Časová konstanta určující maximální růst	⊙2.0 Double (F64)
$t_n$	Časová konstanta určující maximální pokles	⊙2.0 Double (F64)

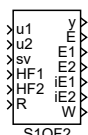
### Výstup

$y$	Filtrovaný výstupní signál	Double (F64)
-----	----------------------------	--------------

## S10F2 – Výběr jednoho ze dvou analogových vstupů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok S10F2 určuje odděleně platnost signálů  $u1$  a  $u2$  stejným způsobem jako blok SAI. Je-li signál  $u1$  (nebo  $u2$ ) neplatný, potom má výstup E1 (nebo E2) hodnotu **on** a kód chyby je na výstupu iE1 (nebo iE2). Dále se v bloku S10F2 vyhodnocuje odchylka vstupu  $u1$  a  $u2$  a nastavuje vnitřní příznak D, který má hodnotu **on** tehdy, jestliže posledních  $nd$  vzorků odchylek  $|u1 - u2|$  splňuje nerovnost:

$$|u1 - u2| > pdev \frac{vmax - vmin}{100},$$

kde  $vmin$  a  $vmax$  jsou po řadě dolní a horní mez vstupů  $u1$  a  $u2$  a  $pdev$  je dovolená procentuální odchylka signálů  $u1$  a  $u2$  z celkového rozsahu. Na základě zjištěné platnosti vstupů (příznaky E1 a E2) a příznaku odchýlení D se určuje zabezpečený výstup  $y$  následujícím způsobem:

(i) **Je-li E1 = off a E2 = off a D = off**, pak výstup  $y$  je podle parametru **mode** dán vztahem:

$$y = \begin{cases} \frac{u1+u2}{2}, & \text{pro mode} = 1, \\ \min(u1, u2), & \text{pro mode} = 2, \\ \max(u1, u2), & \text{pro mode} = 3. \end{cases}$$

a výstup ER má hodnotu **off**, nebyl-li již dříve nastaven na **on**.

(ii) **Je-li E1 = off a E2 = off a D = on**, potom  $y = sv$  a ER = **on**.

(iii) **Je-li E1 = on a E2 = off (E1 = off a E2 = on)**, potom  $y = u2$  ( $y = u1$ ) a výstup ER = **off** nebyl-li již dříve nastaven na **on**.

(iv) **Je-li E1 = on a E2 = on**, potom  $y = sv$  a ER = **on**.

Vstup R resetuje vnitřní příznaky chyb F1–F4 (viz. blok SAI) a příznak D. Je-li trvale R = **on**, potom v případě rozpoznání neplatnosti vstupu  $u1$  ( $u2$ ) je výstup E1 (E2) nahozen pouze po dobu jednoho cyklu. Naproti tomu při R = **off** je E1 = **on** (E2 = **on**) až do následného resetování (náběžná hrana R = **off** → **on**). Pro výstup ER platí obdobné pravidlo. Je-li trvale R = **on**, pak v případě náběžné hrany vnitřního příznaku D (**off** → **on**) je výstup ER nahozen pouze po dobu jednoho cyklu. Při R = **off** je nastaveno ER = **on** až

do následného resetování. Výstup *W* má hodnotu *on* pouze v případech (iii) a (iv), tzn. pokud alespoň jeden z výstupů *E1* a *E2* má hodnotu *on*, tedy pokud je alespoň jeden ze vstupních signálů označen za neplatný.

Parametr *nb* udává počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálů *u1* a *u2*. Parametr *nc* uvádí počet vzorků pro testování neměnnosti (viz blok [SAI](#), podmínka F2). Počet vzorků pro testování variability (viz blok [SAI](#), podmínka F3) je dán parametrem *nr*. Maximální předpokládaná procentuální změna vstupu *u1* (*u2*) z celkového rozsahu *vmax* – *vmin* za *nr* vzorků vstupu *u1* (*u2*) (viz blok [SAI](#)) je určena *prate*. Parametr *nv* vyjadřuje počet vzorků pro testování překročení rozsahu (viz blok [SAI](#), podmínka F4).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

<i>u1</i>	První analogový vstup bloku	Double (F64)
<i>u2</i>	Druhý analogový vstup bloku	Double (F64)
<i>sv</i>	Náhradní hodnota pro případ chyby	Double (F64)
<i>HF1</i>	Příznak hardwarové chyby vstupu <i>u1</i>	Bool
	<i>off</i> ... vstupní modul signálu pracuje normálně	
	<i>on</i> ... došlo k hardwarové chybě vstupního modulu	
<i>HF2</i>	Příznak hardwarové chyby vstupu <i>u2</i>	Bool
	<i>off</i> ... vstupní modul signálu pracuje normálně	
	<i>on</i> ... došlo k hardwarové chybě vstupního modulu	
<i>R</i>	Vynulování vnitřních chybových příznaků	Bool

## Parametr

<i>nb</i>	Počet vynechaných vzorků po restartu	⊙10	Long (I32)
<i>nc</i>	Počet vzorků pro testování neměnnosti	⊙10	Long (I32)
<i>nbits</i>	Počet bitů A/D převodníku	⊙12	Long (I32)
<i>nr</i>	Počet vzorků pro testování variability	⊙10	Long (I32)
<i>prate</i>	Maximální procentuální změna	⊙10.0	Double (F64)
<i>nv</i>	Počet vzorků pro testování překročení rozsahu	⊙1	Long (I32)
<i>vmin</i>	Spodní omezení na vstupní signál	⊙-1.0	Double (F64)
<i>vmax</i>	Horní omezení na vstupní signál	⊙1.0	Double (F64)
<i>nd</i>	Počet vzorků pro vyhodnocování odchýlení	⊙5	Long (I32)
<i>pdev</i>	Maximální povolená procentuální odchylka signálů	⊙10.0	Double (F64)
<i>mode</i>	Způsob výpočtu výstupu při platnosti obou vstupů	⊙1	Long (I32)
	1 ..... průměr		
	2 ..... minimum		
	3 ..... maximum		

## Výstup

<i>y</i>	Analogový výstupní signál	Double (F64)
----------	---------------------------	--------------

E	Indikátor neplatnosti výstupního signálu off ... signál je platný on .... signál není platný	Bool
E1	Indikátor neplatnosti vstupu u1 off ... signál je platný on .... signál není platný	Bool
E2	Indikátor neplatnosti vstupu u2 off ... signál je platný on .... signál není platný	Bool
iE1	Důvod neplatnosti vstupu u1 0 ..... signál je platný 1 ..... signál mimo rozsah 2 ..... signál se mění příliš málo 3 ..... signál se mění jen málo a je mimo rozsah 4 ..... signál se mění příliš mnoho 5 ..... signál se mění příliš mnoho a je mimo rozsah 6 ..... signál se mění příliš málo a příliš mnoho 7 ..... signál se mění příliš málo a příliš mnoho a je mimo rozsah 8 ..... hardwarová chyba	Long (I32)
iE2	Důvod neplatnosti vstupu u2 0 ..... signál je platný 1 ..... signál mimo rozsah 2 ..... signál se mění příliš málo 3 ..... signál se mění jen málo a je mimo rozsah 4 ..... signál se mění příliš mnoho 5 ..... signál se mění příliš mnoho a je mimo rozsah 6 ..... signál se mění příliš málo a příliš mnoho 7 ..... signál se mění příliš málo a příliš mnoho a je mimo rozsah 8 ..... hardwarová chyba	Long (I32)
W	Varování (neplatný vstupní signál) off ... oba vstupní signály jsou platné on .... alespoň jeden vstupní signál je neplatný	Bool



## SAI – Zabezpečený analogový vstup

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

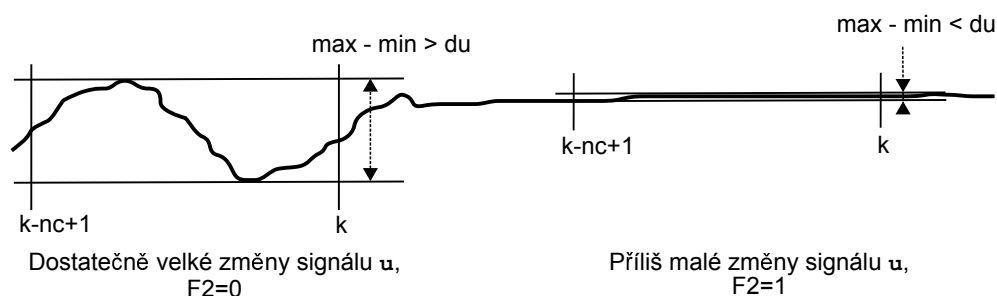
Blok SAI testuje vstupní signál  $u$  s cílem rozpoznání jeho platnosti. Vstupní signál  $u$  se považuje za neplatný (výstup  $E = \text{on}$ ) v následujících případech:

F1: Hardwarová chyba. Vstupní signál  $\text{HWF} = \text{on}$ .

F2: Vstupní signál  $u$  se mění příliš málo. Posledních  $nc$  vzorků vstupu  $u$  leží v intervalu délky  $du$ ,

$$du = \begin{cases} \frac{v_{\max} - v_{\min}}{2^{\text{nbits}}}, & \text{pro nbits} \in \{8, 9, \dots, 16\} \\ 0, & \text{pro nbits} \notin \{8, 9, \dots, 16\}. \end{cases}$$

kde  $v_{\min}$  a  $v_{\max}$  jsou po řadě dolní a horní mez vstupu  $u$  a  $\text{nbits}$  je počet bitů příslušného A/D převodníku. Situace, kdy je splněna podmínka příliš malé změny  $u$ , je zobrazena na následujícím obrázku:



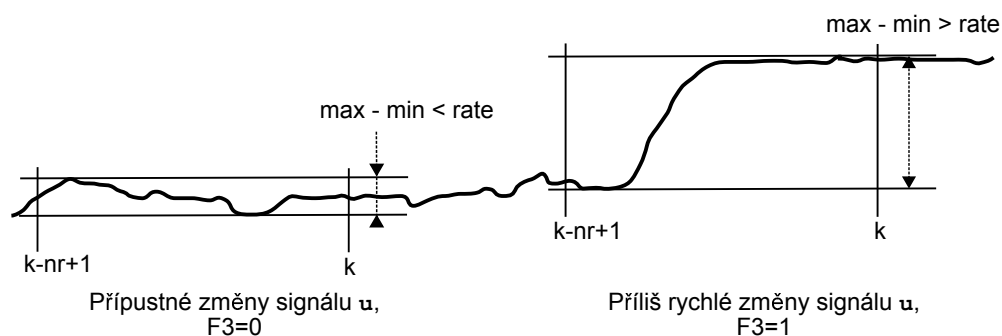
Jestliže je nastaveno  $nc = 0$ , potom podmínka F2 není splněna nikdy.

F3: Vstupní signál  $u$  se mění příliš rychle. Posledních  $nr$  vzorků vstupu  $u$  filtrovaného filtrem [SPIKE](#) neleží v intervalu délky  $\text{rate}$ ,

$$\text{rate} = \text{prate} \frac{v_{\max} - v_{\min}}{100},$$

kde  $\text{prate}$  vyjadřuje dovolenou procentuální změnu signálu  $u$  z celkového rozsahu během  $nr$  vzorků (s ohledem na celkový rozsah vstupního signálu  $u \in (v_{\min}, v_{\max})$ ). V bloku je zařazený [SPIKE](#) filtr s pevnými parametry  $\text{mingap} = (v_{\max} - v_{\min})/100$  a  $q = 2$  odstraňující ze signálu úzké špičky, které by mohly způsobovat nežádoucí splnění této podmínky (blíže viz popis bloku [SPIKE](#)). Situace, kdy je splněna

podmínka příliš rychlé změny vstupního signálu  $u$ , je zobrazena na následujícím obrázku:



Jestliže je nastaveno  $nr = 0$ , potom podmínka F3 není splněna nikdy.

F4: Vstupní signál  $u$  je mimo rozsah. Posledních  $nv$  vzorků vstupu  $u$  leží mimo přípustný interval  $\langle v_{\min}, v_{\max} \rangle$ . Jestliže je nastaveno  $nv = 0$ , potom podmínka F4 není splněna nikdy.

Je-li signál  $u$  platný, potom je beze změny kopírován na výstup  $y$ . V opačném případě je do výstupu  $y$  dosazena náhradní hodnota ze vstupu  $sv$ . V tomto případě má výstup  $E$  hodnotu  $on$  a výstup  $iE$  udává kód rozpoznané chyby vstupu  $u$  (viz tabulka níže). Vstup  $R$  resetuje vnitřní příznaky chyb F1–F4. Je-li trvale  $R = on$ , potom v případě rozpoznání neplatnosti vstupu  $u$  je výstup  $E$  nahozen pouze po dobu jednoho cyklu. Naproti tomu při  $R = off$  je  $E = on$  až do následného resetování (náběžná hrana  $R: off \rightarrow on$ ).

Tabulka kódů chyb  $iE$  podle vnitřních příznaků F1–F4:

F1	F2	F3	F4	$iE$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	*	*	*	8

Parametr  $nb$  určuje počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálu  $u$ . Doporučuje se volit  $nb \geq 5$  z důvodu odeznění počátečních podmínek SPIKE filtru.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$u$

Analogový vstupní signál

Double (F64)

sv	Náhradní hodnota pro případ chyby	Double (F64)
HWF	Příznak hardwarové chyby off ... vstupní modul signálu pracuje normálně on .... došlo k hardwarové chybě vstupního modulu	Bool
R	Vynulování vnitřních chybových příznaků	Bool

## Parametr

nb	Počet vynechaných vzorků po restartu	⊙10	Long (I32)
nc	Počet vzorků pro testování neměnnosti	⊙10	Long (I32)
nbits	Počet bitů A/D převodníku	⊙12	Long (I32)
nr	Počet vzorků pro testování variability	⊙10	Long (I32)
prate	Maximální procentuální změna	⊙10.0	Double (F64)
nv	Počet vzorků pro testování překročení rozsahu	⊙1	Long (I32)
vmin	Spodní omezení na vstupní signál	⊙-1.0	Double (F64)
vmax	Horní omezení na vstupní signál	⊙1.0	Double (F64)

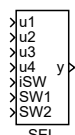
## Výstup

y	Analogový výstupní signál	Double (F64)
yf	Výstupní signál filtrovaný SPIKE algoritmem	Double (F64)
E	Indikátor neplatnosti výstupního signálu off ... signál je platný on .... signál není platný	Bool
iE	Důvod neplatnosti signálu 0 ..... signál je platný 1 ..... signál mimo rozsah 2 ..... signál se mění příliš málo 3 ..... signál se mění jen málo a je mimo rozsah 4 ..... signál se mění příliš mnoho 5 ..... signál se mění příliš mnoho a je mimo rozsah 6 ..... signál se mění příliš málo a příliš mnoho 7 ..... signál se mění příliš málo a příliš mnoho a je mimo rozsah 8 ..... hardwarová chyba	Long (I32)

## SEL – Selektor analogového signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SEL** nebude dále podporován, nahraďte jej blokem [SELQUAD](#), [SELOCT](#) nebo [SELHEXD](#). Pozor na změnu významu vstupních signálů *SWn*.

Blok **SEL** realizuje výběr zvoleného signálu ze čtyř vstupních signálů *u1*, *u2*, *u3* a *u4* a kopíruje ho na výstup *y*. Výběr se provádí podle vstupu *iSW*, jestliže je *BINF = off* nebo podle binárních vstupů *SW1* a *SW2* (*BINF = on*) dle následující tabulky:

<i>iSW</i>	<i>SW1</i>	<i>SW2</i>	<i>y</i>
0	off	off	<i>u1</i>
1	off	on	<i>u2</i>
2	on	off	<i>u3</i>
3	on	on	<i>u4</i>

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<i>u1</i>	První analogový vstup bloku	Double (F64)
<i>u2</i>	Druhý analogový vstup bloku	Double (F64)
<i>u3</i>	Třetí analogový vstup bloku	Double (F64)
<i>u4</i>	Čtvrtý analogový vstup bloku	Double (F64)
<i>iSW</i>	Selektor aktivního signálu	Long (I32)
<i>SW1</i>	Binární vstup pro výběr	Bool
<i>SW2</i>	Binární vstup pro výběr	Bool

### Parametr

<i>BINF</i>	Výběr pomocí binárních vstupů off ... zakázáno (výběr přes <i>iSW</i> ) on .... povoleno (výběr přes <i>SW1</i> a <i>SW2</i> )	Bool
-------------	--	------

Výstup

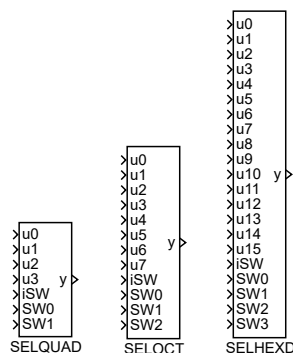
y

Zvolený vstupní signál

Double (F64)

## SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu

Symboly bloků

Licence: [STANDARD](#)

## Popis funkce

Bloky SELQUAD, SELOCT a SELHEX realizují výběr zvoleného signálu ze vstupních signálů a kopírují ho na výstup y. Jediný rozdíl mezi bloky je v počtu vstupních signálů. Výběr aktivního vstupu u0...u15 se provádí podle vstupu iSW, jestliže je BINF = off nebo podle binárních vstupů SW0...SW3 (BINF = on) dle následující tabulky:

iSW	SW0	SW1	SW2	SW3	y
0	off	off	off	off	u0
1	on	off	off	off	u1
2	off	on	off	off	u2
3	on	on	off	off	u3
4	off	off	on	off	u4
5	on	off	on	off	u5
6	off	on	on	off	u6
7	on	on	on	off	u7
8	off	off	off	on	u8
9	on	off	off	on	u9
10	off	on	off	on	u10
11	on	on	off	on	u11
12	off	off	on	on	u12
13	on	off	on	on	u13
14	off	on	on	on	u14
15	on	on	on	on	u15

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

u0	Analogový vstupní signál	Double (F64)
u1	Analogový vstupní signál	Double (F64)
u2	Analogový vstupní signál	Double (F64)
u3	Analogový vstupní signál	Double (F64)
u4	Analogový vstupní signál	Double (F64)
u5	Analogový vstupní signál	Double (F64)
u6	Analogový vstupní signál	Double (F64)
u7	Analogový vstupní signál	Double (F64)
u8	Analogový vstupní signál	Double (F64)
u9	Analogový vstupní signál	Double (F64)
u10	Analogový vstupní signál	Double (F64)
u11	Analogový vstupní signál	Double (F64)
u12	Analogový vstupní signál	Double (F64)
u13	Analogový vstupní signál	Double (F64)
u14	Analogový vstupní signál	Double (F64)
u15	Analogový vstupní signál	Double (F64)
iSW	Selektor aktivního signálu	Long (I32)
SW0	Binární vstup pro výběr	Bool
SW1	Binární vstup pro výběr	Bool
SW2	Binární vstup pro výběr	Bool
SW3	Binární vstup pro výběr	Bool

## Parametr

BINF	Výběr pomocí binárních vstupů	Bool
------	-------------------------------	------

## Výstup

y	Zvolený vstupní signál	Double (F64)
---	------------------------	--------------

## SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok realizuje funkci posuvného registru s osmi výstupy pro libovolný typ signálů.

Je-li aktivní vstup RUN, je v každém tiku algoritmu provedeno následující přiřazení:

$$y_i = y_{i-1}, \quad i = 1..7,$$

$$y_0 = u,$$

tedy hodnota na každém z výstupů  $y_0$  až  $y_6$  je posunuta na výstup v pořadí následující, a hodnota vstupu  $u$  je přenesena na výstup  $y_0$ .

Blok pracuje s libovolným datovým typem signálu přivedeného na vstup  $u$ . Požadovaný datový typ je třeba nastavit parametrem  $vtype$ . Výstupy  $y_0$  až  $y_7$  jsou pak shodného datového typu.

Pokud potřebujete posun dat v registru provádět na základě triggeru, vložte před vstup RUN blok [EDGE](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

$u$	Analogový vstupní signál	Any
RUN	Povoluje posun výstupů	Bool

## Parametr

$vtype$	Typ výstupů	⊙8 Long (I32)
1	.... Bool	
2	.... Byte (U8)	
3	.... Short (I16)	
4	.... Long (I32)	
5	.... Word (U16)	
6	.... DWord (U32)	
7	.... Float (F32)	
8	.... Double (F64)	
10	.... Large (I64)	



## Výstup

y0	První analogový výstup bloku	Any
y1	Druhý analogový výstup bloku	Any
y2	Třetí analogový výstup bloku	Any
y3	Čtvrtý analogový výstup bloku	Any
y4	Pátý analogový výstup bloku	Any
y5	Šestý analogový výstup bloku	Any
y6	Sedmý analogový výstup bloku	Any
y7	Osmý analogový výstup bloku	Any

## SHLD – Vzorkovač (sample and hold)

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok SHLD je určen pro podržení hodnoty vstupního signálu  $u$ , přičemž jeho funkce je dána parametrem `mode`.

V případě *vynuceného vzorkování* se nastaví výstup  $y$  na hodnotu vstupu  $u$  v okamžiku náběžné hrany (`off`→`on`) řídicího vstupu `SETH` a zůstává konstantní až do příchodu nové náběžné hrany.

Pokud je zvoleno *držení předchozí hodnoty*, na výstup  $y$  se nastaví poslední hodnota vstupu  $u$  před příchodem vzestupné hrany na vstupu `SETH`. Tato hodnota je držena po celou dobu, kdy platí `SETH = on`. Pokud je na vstupu `SETH = off`, je vstup  $u$  jednoduše kopírován na výstup  $y$ .

Při režimu *držení aktuální hodnoty* se na výstup  $y$  nastaví hodnota vstupu  $u$  v okamžiku náběžné hrany. Tato hodnota je držena po celou dobu, kdy platí `SETH = on`. Pokud je na vstupu `SETH = off`, je vstup  $u$  jednoduše kopírován na výstup  $y$ .

Vstup `R1` slouží k resetování bloku, inicializuje výstup  $y$  na hodnotu  $y_0$  a má prioritu před vstupem `SETH`.

Zvažte též použití bloku [PARR](#), který může být rovněž použit pro uložení číselné hodnoty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>u</code>	Analogový vstupní signál	Double (F64)
<code>SETH</code>	Vstup pro nastavení a podržení výstupní hodnoty	Bool
<code>R1</code>	Reset bloku	Bool

### Parametr

<code>y0</code>	Počáteční hodnota výstupu	Double (F64)
<code>mode</code>	Režim vzorkování	⊙3 Long (I32)
	1 . . . . . Vynucené vzorkování	
	2 . . . . . Držení předchozí hodnoty	
	3 . . . . . Držení aktuální hodnoty	

Výstup

y

Analogový výstupní signál

Double (F64)

## SINT – Jednoduchý integrátor

Symbol bloku

Licence: **STANDARD**



Popis funkce

Blok SINT realizuje diskretní integrátor popsany diferencní rovnicí

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde  $T_S$  je perioda spouštění bloku a  $t_i$  je integrační konstanta. Pokud je  $T_i = 0$ , místo integrace se provádí sumace podle vztahu

$$y_k = y_{k-1} + u_k.$$

Pro  $T_i < 0$  je chování nedefinované.

Je-li  $y_k$  mimo saturační meze  $y_{\min}$ ,  $y_{\max}$ , potom je výstup i stav integrátoru příslušně omezen.

Pro složitější případy je možné použít blok **INTE**, který disponuje rozšířenou funkcionalitou.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

u	Analogový vstupní signál	Double (F64)
---	--------------------------	--------------

### Parametr

t <sub>i</sub>	Integrační časová konstanta	⊙1.0	Double (F64)
y <sub>0</sub>	Počáteční hodnota výstupu		Double (F64)
y <sub>max</sub>	Horní saturační mez	⊙1.0	Double (F64)
y <sub>min</sub>	Dolní saturační mez	⊙-1.0	Double (F64)

### Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

## SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok **SPIKE** realizuje nelineární filtr odstraňující ze vstupního signálu  $u$  izolované úzké špičky (pulzy). Jeden krok **SPIKE** filtru provádí následující transformaci  $(u, y) \rightarrow y$ :

```
delta := y - u;
if abs(delta) < gap
  then
    begin
      y := u;
      gap := gap/q;
      ifgap < mingap then gap:= mingap;
    end
  else
    begin
      if delta < 0
        then y := y + gap
        else y := y - gap;
      gap := gap * q;
    end
```

kde `mingap` a `q` jsou parametry bloku.

Zvolíme-li parametr `mingap` dostatečně velký, potom signál prochází filtrem beze změny. Zmenšováním tohoto parametru je možné docílit stav, kdy dojde k odfiltrování nežádoucích špiček, ale jinak zůstává signál nezkreslen. Doporučená volba je 1 % z celkového rozsahu vstupního signálu  $u$ . Parametr `q` určuje rychlost adaptace tolerančního okénka filtru.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>u</code>	Vstupní signál filtru	Double (F64)
----------------	-----------------------	--------------

### Parametr

<code>mingap</code>	Minimální velikost tolerančního okénka	⊙0.01 Double (F64)
---------------------	--	--------------------

*q* Rychlost adaptace tolerančního okénka  $\downarrow 1.0 \oplus 2.0$  Double (F64)

Výstup

*y* Filtrovaný výstupní signál Double (F64)

## SSW – Jednoduchý přepínač

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SSW** provádí výběr jednoho ze dvou vstupních signálů **u1**, **u2** podle logického vstupu **SW** a získanou hodnotu kopíruje na výstup **y**. Je-li **SW = off** (**SW = on**), potom je vybraný vstup **u1** (**u2**).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u1</b>	První analogový vstup bloku	Any
<b>u2</b>	Druhý analogový vstup bloku	Any
<b>SW</b>	Přepínací signál	Bool
	<b>off</b> ... je zvolen vstupní signál <b>u1</b>	
	<b>on</b> .... je zvolen vstupní signál <b>u2</b>	

### Výstup

<b>y</b>	Analogový výstupní signál	Any
----------	---------------------------	-----

## SWR – Přepínač s rampovou funkcí

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SWR** provádí výběr jednoho ze dvou vstupních signálů **u1**, **u2** podle logického vstupu **SW** a podle něho nastavuje výstup **y**. Je-li **SW = off** (**SW = on**), potom je vybrán vstup **u1** (**u2**). Při přepnutí vstupu se výstup nezmění okamžitě, ale vysleduje vybraný vstup po rampě s definovanou strmostí. Tato strmost může být různá pro oba vstupy **u1**, **u2** a je určena po řadě časovými konstantami **t1** a **t2**. Po vysledování vstupu se funkce omezení strmosti vypne až do následujícího přepnutí.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>u1</b>	První analogový vstup bloku	Double (F64)
<b>u2</b>	Druhý analogový vstup bloku	Double (F64)
<b>SW</b>	Přepínací signál	Bool
	<b>off</b> ... je zvolen vstupní signál <b>u1</b>	
	<b>on</b> .... je zvolen vstupní signál <b>u2</b>	

### Parametr

<b>t1</b>	Časová konstanta omezovače strmosti, <b>u2</b> -> <b>u1</b>	⊙1.0	Double (F64)
<b>t2</b>	Časová konstanta omezovače strmosti, <b>u1</b> -> <b>u2</b>	⊙1.0	Double (F64)
<b>y0</b>	Počáteční hodnota výstupu		Double (F64)

### Výstup

<b>y</b>	Analogový výstupní signál	Double (F64)
----------	---------------------------	--------------



## VDEL – Dopravní zpoždění s proměnnou délkou

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok VDEL realizuje proměnné časové zpoždění vstupního signálu  $u$  o čas přivedený na vstup  $d$ . Přesněji, výstupní signál  $y$  je zpožděn o čas, který vznikne zaokrouhlením vstupu  $d$  na nejbližší celočíselný násobek  $n$  periody  $T_S$  spuštění bloku. Jestliže po spuštění bloku není dosud k dispozici  $n$  posledních vzorků, potom je výstup  $y$  nastaven na inicializační hodnotu  $y_0$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

$u$	Analogový vstupní signál	Double (F64)
$d$	Časové zpoždění [s]	Double (F64)

### Parametr

$y_0$	Počáteční hodnota výstupu	Double (F64)
$n_{max}$	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊖1000 Long (I32)

### Výstup

$y$	Zpožděný vstupní signál	Double (F64)
-----	-------------------------	--------------

## ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací

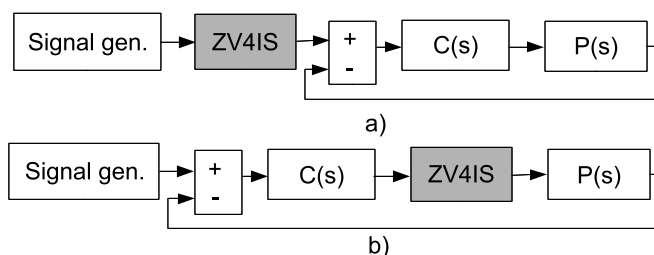
Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok ZV4IS realizuje funkci frekvenčního filtru typu pásmová zadrž. Hlavní oblastí použití je řízení pohybu mechanických systémů s pružnými částmi, kde vlivem nedostatečné tuhosti konstrukce hrozí nebezpečí vzniku reziduálních vibrací. Ty se projevují jako mechanické chvění vybuzené v důsledku momentu nebo síly, kterou aktuátory působí na pracovní mechanismus stroje. Tyto vibrace mohou mít negativní vliv na přesnost regulace, vedou ke zvýšenému opotřebením mechanických částí stroje a v krajním případě mohou způsobit nestabilitu regulačních smyček. Obecně lze tvarovací filtr využít v libovolné aplikaci pro řízení kmitavých systémů nebo pro potlačení konkrétní frekvence ve spojitém signálu.



Tvarovací filtr je možné použít dvěma způsoby. Zapojením v *otevřené smyčce* (viz obrázek výše nahoře) je modifikován referenční signál přicházející od obsluhy nebo od generátoru trajektorie z vyšší úrovně řízení. Výhodou tohoto uspořádání je, že dynamika vlastního filtru neovlivňuje chování podřízené zpětnovazební smyčky. Podmínkou správné funkce je korektní nastavení regulátoru  $C(s)$  ve zpětné vazbě, který musí pracovat v lineárním režimu. V opačném případě může dojít ke zkreslení frekvenčního spektra akční veličiny a tím k vybuzení nežádoucích kmitů na rezonančních frekvencích stroje (ve schématu  $P(s)$ ). Záporům přímovazebního zapojení je absence tlumení při působení vnějších poruch. Tuto nevýhodu odstraňuje *zapojení v uzavřené smyčce* (na obrázku dole), kdy filtr je umístěn za zpětnovazební regulátor a tvaruje přímo akční veličinu. V této variantě jsou kompenzovány vibrace vybuzené jak v důsledku změny referenčního signálu tak vlivem působících vnějších poruch. Nevýhodou tohoto zapojení je zanesení dynamického zpoždění do zpětné vazby a tím nutnost přeladit vnitřní regulátor.

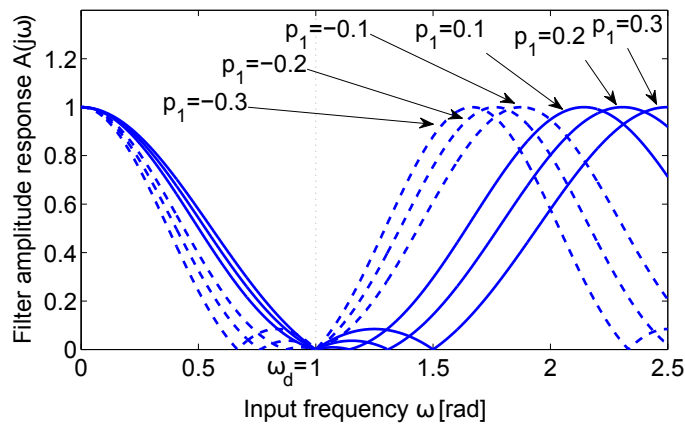
Vlastní algoritmus filtrace lze popsat v časové oblasti vztahem

$$y(t) = A_1 u(t - t_1) + A_2 u(t - t_2) + A_3 u(t - t_3) + A_4 u(t - t_4)$$

Filtr má tedy strukturu sumy vážených dopravních zpoždění kde zesílení  $A_1..A_4$  a hodnoty zpoždění  $t_1..t_4$  závisí na volbě typu filtru, frekvenci a tlumení kmitavého módu systému. Výhodou uvedené struktury oproti klasickým dynamickým notch filtrům užívaným v regulační technice je konečná impulzní odezva, která je důležitá zejména v aplikacích řízení pohybu, zaručená stabilita a monotónní přechodová charakteristika filtru a obecně menší zpoždění zaváděné do cesty signálu.

Pro správnou funkci filtru je třeba zadat vlastní frekvenci  $\omega$  a tlumení  $\xi$  kmitavého módu, který má být potlačen. Parametr  $i\text{par}$  pak udává typ tvarovacího filtru, pro  $i\text{par} = 1$  je použit jeden z deseti základních filtrů, které se volí parametrem  $i\text{stype}$ . Jednotlivé typy se liší tvarem frekvenční charakteristiky a šířkou nepropustného pásma. Při přesné znalosti  $\omega$  a  $\xi$  je vhodný filtr typu ZV nebo ZVD (Zero Vibration), které dosahují nejrychlejší odezvy na vstupní signál. Při velké neurčitosti v modelu systému/signálu lze použít robustní filtry UEI (Extra Insensitive) nebo UTHEI, které dosahují velké šířky nepropustného pásma za cenu delší odezvy filtru. Číslo na konci názvu filtru odpovídá maximální přípustné procentuální úrovni vybuzených vibrací pro dané  $\omega$  a  $\xi$  (jedno, dvě nebo pět procent).

Pro jemné ladění filtru lze použít kompletní parametrizaci volbou  $i\text{par} = 2$ , pro kterou se zpřístupní volné parametry  $p\_alpha, p\_a2$  a  $p\_a3$ . Ty určují tvar frekvenční charakteristiky filtru a lze je použít pro nalezení optimálního kompromisu mezi robustností a zavedeným zpožděním.



Parametr asymetrie  $p\_alpha$  určuje relativní polohu sedla nepropustné oblasti frekvenční charakteristiky filtru vzhledem k nastavené frekvenci  $\omega$ . Kladná hodnota znamená posun vpravo do oblasti vyšších frekvencí, záporná do opačného směru, nulová hodnota vede na symetrickou charakteristiku (viz obrázek výše). S parametrem  $p\_alpha$  souvisí také délka filtru, tedy celkové zpoždění zavedené do cesty vstupního signálu, obecně menší hodnota znamená pomalejší filtr s větším zpožděním. Asymetrické filtry jsou vhodné v případech, kdy frekvence, která má být tlumena je proměnná a pohybuje se v určitém intervalu nad nebo pod nominální známou hodnotou, přičemž vyšší pravděpodobnost se předpokládá na jednom z okrajů intervalu (asymetrická hustota pravděpodobnosti).



<code>xi</code>	Součinitel relativního tlumení		Double (F64)
<code>ipar</code>	Specifikace	⊙1	Long (I32)
	1 ..... základní typy tvarovačů signálu		
	2 ..... kompletní parametrizace		
<code>istype</code>	Typ	⊙2	Long (I32)
	1 ..... ZV		
	2 ..... ZVD		
	3 ..... ZVDD		
	4 ..... MISZV		
	5 ..... UEI1		
	6 ..... UEI2		
	7 ..... UEI5		
	8 ..... UTHEI1		
	9 ..... UTHEI2		
	10 ..... UTHEI5		
<code>p_alpha</code>	Asymetrie tvarovače	⊙0.2	Double (F64)
<code>p_a2</code>	Necitlivost	⊙0.5	Double (F64)
<code>p_a3</code>	Přídavný parametr (pouze pro <code>p_alpha=0</code> )	⊙0.5	Double (F64)
<code>nmax</code>	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙1000	Long (I32)

## Výstup

<code>y</code>	Filtrovaný výstupní signál		Double (F64)
<code>E</code>	Příznak chyby		Bool
	off ... bez chyby		
	on ... nastala chyba		



## Kapitola 6

# GEN – Generátory signálů

### Obsah

---

ANLS – Řízený generátor po částech lineární funkce . . . . .	184
BINS – Řízený generátor binární posloupnosti . . . . .	186
BIS – Generátor binární posloupnosti . . . . .	188
BISR – * Generátor binární posloupnosti s možností pozastavení	189
MP – Ručně generovaný pulz . . . . .	190
PRBS – Pseudonáhodná binární posloupnost . . . . .	191
SG, SGI – Řízený generátor signálu . . . . .	193

---

Knihovna GEN je specializována na generování signálů. Obsahuje bloky jako [ANLS](#) pro generování po částech lineární funkce času nebo generátory binárních sekvencí [BINS](#), [BIS](#), [BISR](#). Knihovna také nabízí [MP](#) pro manuální generování pulsních signálů, [PRBS](#) pro generování pseudonáhodných binárních sekvencí a [SG](#) pro generování periodických signálů. Tato knihovna poskytuje zásadní nástroje pro vytváření a manipulaci s různými typy signálů.

## ANLS – Řízený generátor po částech lineární funkce

Symbol bloku

Licence: **STANDARD**

## Popis funkce

Blok ANLS generuje na výstupu  $y$  po částech lineární funkci zadanou uzlovými body  $t_1, y_1; t_2, y_2; t_3, y_3; t_4, y_4$ . Počáteční hodnota  $y$  je definována parametrem  $y_0$ . Start generování funkce (časový okamžik 0) je určen náběžnou hranou vstupu RUN. V intervalu  $\langle t_i, t_{i+1} \rangle, i = 0, \dots, 3, t_0 = 0$  je výstup  $y$  definován vztahem

$$y = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i} (t - t_i).$$

Je-li  $t_i = t_{i+1}$ , potom se výstup  $y$  mění v čase  $t_i$  skokem z hodnoty  $y_i$  na hodnotu  $y_{i+1}$ . Generování funkce je předčasně ukončeno v případě, že  $RUN = \text{off}$  (výstup je resetován na  $y_0$  a  $is$  na 0), nebo jestliže  $t > t^*$ , kde  $t^*$  je rovno času  $t_i$ , kde index  $i \leq 4$  je největší možné celé číslo takové, že  $t_1 < \dots < t_i$ . Po tomto tzv. normálním ukončení si výstup podrží svoji předcházející hodnotu. Má-li parametr RPT hodnotu **on**, potom se po normálním ukončení spustí opětovné generování funkce podle stejného algoritmu atd. Takto lze například generovat obdélníkový, pilovitý nebo lichoběžníkový signál.

## Vstup

RUN	Povolení generování posloupnosti	Bool
-----	----------------------------------	------

## Výstupy

$y$	Analogový výstupní signál	Double (F64)
$is$	Index aktivního časového úseku	Long (I32)

## Parametry

$y_0$	Počáteční hodnota výstupu		Double (F64)
$t_1$	Čas uzlového bodu 1	⊙1.0	Double (F64)
$y_1$	Hodnota uzlového bodu 1		Double (F64)
$t_2$	Čas uzlového bodu 2	⊙1.0	Double (F64)
$y_2$	Hodnota uzlového bodu 2	⊙1.0	Double (F64)
$t_3$	Čas uzlového bodu 3	⊙2.0	Double (F64)
$y_3$	Hodnota uzlového bodu 3	⊙1.0	Double (F64)
$t_4$	Čas uzlového bodu 4	⊙2.0	Double (F64)



y4	Hodnota uzlového bodu 4	Double (F64)
RPT	Opakování sekvence	Bool
	off ... zakázáno	on .... povoleno

## BINS – Řízený generátor binární posloupnosti

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok BINS generuje na výstupu Y zadanou binární posloupnost. Počáteční hodnota Y je definována parametrem Y0. Start generování posloupnosti (časový okamžik 0) je určen náběžnou hranou vstupu START. V časech  $t_1, t_2, \dots, t_8$  se mění hodnota výstupu Y na hodnotu opačnou ( $\text{off} \rightarrow \text{on}$ ,  $\text{on} \rightarrow \text{off}$ ). V případě, že je parametr RPT nastaven na  $\text{off}$ , nastane poslední přepnutí výstupu v čase  $t_i$ , jestliže  $t_{i+1} < t_i$ . Výstup si poté podrží svoji poslední hodnotu. Má-li však parametr RPT hodnotu  $\text{on}$ , potom se místo posledního přepnutí vrátí blok do svého původního stavu Y0, interní čas bloku je nastaven na 0 a generování posloupnosti se periodicky opakuje. Dojde-li ke změně parametrů bloku při běhu, potom se nové parametry uplatní až při následném spuštění generování posloupnosti. Poznamenejme, že k opětovnému spuštění generování posloupnosti může dojít i za běhu generátoru.

Časové okamžiky přepnutí jsou interně zaokrouhlovány na nejbližší celý násobek periody spouštění bloku, což může vést např. k vymizení pulzů, které jsou užší než  $T_S/2$  nebo spojení více po sobě jdoucích úzkých pulzů do jednoho širokého pulzu. Je proto důrazně doporučováno zadávat okamžiky přepnutí jako celé násobky periody spouštění bloku.

### Vstup

START	Spouštěcí signál (náběžná hrana)	Bool
-------	----------------------------------	------

### Výstupy

Y	Logický výstupní signál	Bool
is	Index aktivního časového úseku	Long (I32)

### Parametry

Y0	Počáteční hodnota výstupu off ... vypnuto/nepřavda on .... zapnuto/pravda	Bool
t1	Okamžik přepnutí 1 [s]	↓0.0 ⊙1.0 Double (F64)
t2	Okamžik přepnutí 2 [s]	↓0.0 ⊙2.0 Double (F64)
t3	Okamžik přepnutí 3 [s]	↓0.0 ⊙3.0 Double (F64)
t4	Okamžik přepnutí 4 [s]	↓0.0 ⊙4.0 Double (F64)

t5	Okamžik přepnutí 5 [s]	↓0.0 ⊕5.0	Double (F64)
t6	Okamžik přepnutí 6 [s]	↓0.0 ⊕6.0	Double (F64)
t7	Okamžik přepnutí 7 [s]	↓0.0 ⊕7.0	Double (F64)
t8	Okamžik přepnutí 8 [s]	↓0.0 ⊕8.0	Double (F64)
RPT	Opakování sekvence		Bool
	off ... zakázáno		
	on .... povoleno		

## BIS – Generátor binární posloupnosti

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok BIS generuje na výstupu Y zadanou binární posloupnost. Okamžiku spuštění exekuce bloku je přiřazen čas 0. Počáteční hodnota výstupu Y je definována parametrem Y0. V časech  $t_1, t_2, \dots, t_8$  se mění hodnota výstupu Y na hodnotu opačnou ( $\text{off} \rightarrow \text{on}$ ,  $\text{on} \rightarrow \text{off}$ ). V případě, že je parametr RPT nastaven na **off**, nastane poslední přepnutí výstupu v čase  $t_i$ , jestliže  $t_{i+1} < t_i$ . Výstup si poté podrží svoji poslední hodnotu. Má-li však parametr RPT hodnotu **on**, potom se místo posledního přepnutí vrátí blok do svého původního stavu Y0, interní čas bloku je nastaven na 0 a generování posloupnosti se periodicky opakuje.

Časové okamžiky přepnutí jsou interně zaokrouhlovány na nejbližší celý násobek periody spouštění bloku, což může vést např. k vymizení pulzů, které jsou užší než  $T_S/2$  nebo spojení více po sobě jdoucích úzkých pulzů do jednoho širokého pulzu. Je proto důrazně doporučováno zadávat okamžiky přepnutí jako celé násobky periody spouštění bloku.

### Výstupy

Y	Logický výstupní signál	Bool
is	Index aktivního časového úseku	Long (I32)

### Parametry

Y0	Počáteční hodnota výstupu <b>off</b> ... vypnuto/nepravda <b>on</b> ... zapnuto/pravda	Bool
t1	Okamžik přepnutí 1 [s]	↓0.0 ⊕1.0 Double (F64)
t2	Okamžik přepnutí 2 [s]	↓0.0 ⊕2.0 Double (F64)
t3	Okamžik přepnutí 3 [s]	↓0.0 ⊕3.0 Double (F64)
t4	Okamžik přepnutí 4 [s]	↓0.0 ⊕4.0 Double (F64)
t5	Okamžik přepnutí 5 [s]	↓0.0 ⊕5.0 Double (F64)
t6	Okamžik přepnutí 6 [s]	↓0.0 ⊕6.0 Double (F64)
t7	Okamžik přepnutí 7 [s]	↓0.0 ⊕7.0 Double (F64)
t8	Okamžik přepnutí 8 [s]	↓0.0 ⊕8.0 Double (F64)
RPT	Opakování sekvence <b>off</b> ... zakázáno <b>on</b> ... povoleno	Bool

## BISR – \* Generátor binární posloupnosti s možností pozastavení

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

RUN	Povolení běhu algoritmu	Bool
R1	Reset bloku	Bool

### Parametr

Y0	Počáteční hodnota výstupu off ... vypnuto/nepřavda on .... zapnuto/pravda	Bool
ADDT	Postupné časování off ... Absolutní časování (sekvence jako celek) on .... Postupné časování (segment po segmentu)	Bool
RPT	Opakování sekvence off ... zakázáno on .... povoleno	Bool
t1	Okamžik přepnutí 1 [s]	↓0.0 ⊕1.0 Double (F64)
t2	Okamžik přepnutí 2 [s]	↓0.0 ⊕2.0 Double (F64)
t3	Okamžik přepnutí 3 [s]	↓0.0 ⊕3.0 Double (F64)
t4	Okamžik přepnutí 4 [s]	↓0.0 ⊕4.0 Double (F64)
t5	Okamžik přepnutí 5 [s]	↓0.0 ⊕5.0 Double (F64)
t6	Okamžik přepnutí 6 [s]	↓0.0 ⊕6.0 Double (F64)
t7	Okamžik přepnutí 7 [s]	↓0.0 ⊕7.0 Double (F64)
t8	Okamžik přepnutí 8 [s]	↓0.0 ⊕8.0 Double (F64)

### Výstup

Y	Logický výstupní signál	Bool
is	Index aktivního časového úseku	Long (I32)

## MP – Ručně generovaný pulz

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok MP generuje na výstupu Y pulz délky `pwidth` při náběžné hraně parametru `BSTATE` (`off`→`on`). Hodnota parametru `BSTATE` je algoritmem bloku okamžitě shozena na hodnotu `off` (`BSTATE` představuje signál z krátce stisknutého tlačítka). Je-li `RPTF` = `on`, potom je aktivní opětovné nahození `BSTATE` během generování pulsu a výsledkem je prodloužení generovaného pulsu. Je-li `RPTF` = `off`, je nahození `BSTATE` během generování výstupního pulsu neúčinné.

Blok MP reaguje pouze na náběžnou hranu parametru `BSTATE`, nelze ho tedy použít k vygenerování pulsu ihned při startu exekutivy. K tomuto účelu použijte blok [BIS](#).

### Výstup

Y	Logický výstupní signál	Bool
---	-------------------------	------

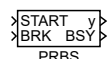
### Parametry

<code>pwidth</code>	Šířka pulzu [s] (0 pro jeden tick)	⊙1.0	Double (F64)
<code>BSTATE</code>	Aktivace výstupního pulzu <code>off</code> ... žádná činnost <code>on</code> ... vygenerování výstupního pulzu		Bool
<code>RPTF</code>	Povolení prodloužení pulzu <code>off</code> ... zakázáno <code>on</code> ... povoleno		Bool

## PRBS – Pseudonáhodná binární posloupnost

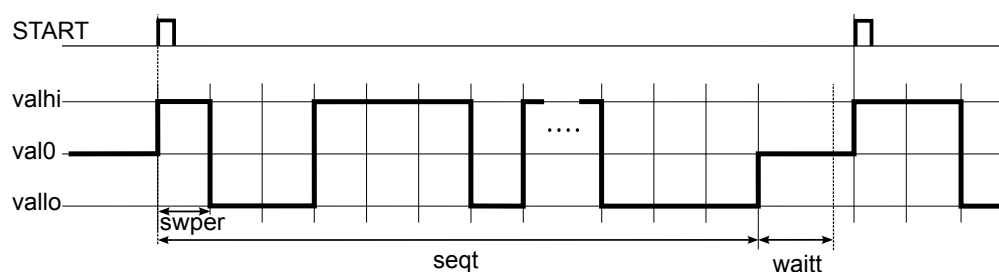
Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok PRBS generuje pseudonáhodnou binární posloupnost. Způsob generování je zřejmý z níže uvedeného obrázku.



Počáteční a konečná hodnota posloupnosti je `val0`. Z této hodnoty je generování startováno náběžnou hranou na vstupu `START` (`off`→`on`). V tom okamžiku se výstup `y` přepne z hodnoty `val0` na hodnotu `valhi` a dále se přepíná na druhou možnou hladinu s časovou periodou `swper` a pravděpodobností `swprob`. Tak se postupuje až do uběhnutí času `seqt`. Posloupnost je ukončena opět hodnotou `val0`. Následuje prodleva `waitt` sloužící pro ustálení odezvy řízené soustavy. Teprve poté je možné odstartovat generování nové posloupnosti. V případě potřeby je možné generování posloupnosti přerušit vstupem `BRK = on`.

### Vstupy

<code>START</code>	Spouštěcí signál (náběžná hrana)	<code>Bool</code>
<code>BRK</code>	Signál pro přerušení	<code>Bool</code>

### Výstupy

<code>y</code>	Vygenerovaná pseudonáhodná binární posloupnost	<code>Double (F64)</code>
<code>BSY</code>	Příznak probíhající operace	<code>Bool</code>

### Parametry

<code>val0</code>	Počáteční a koncová hodnota	<code>Double (F64)</code>
<code>valhi</code>	Horní hladina výstupu <code>y</code>	⊕1.0 <code>Double (F64)</code>
<code>vallo</code>	Dolní hladina výstupu <code>y</code>	⊖1.0 <code>Double (F64)</code>

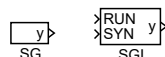
<code>swper</code>	Perioda náhodného přepínání výstupu $y$ mezi hladinami [s]	$\odot 1.0$	Double (F64)
<code>swprob</code>	Pravděpodobnost přepnutí	$\downarrow 0.0 \uparrow 1.0 \odot 0.2$	Double (F64)
<code>seqt</code>	Délka posloupnosti [s]	$\odot 10.0$	Double (F64)
<code>waitt</code>	Doba ustálení [s]	$\odot 2.0$	Double (F64)



## SG, SGI – Řízený generátor signálu

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **SG** a **SGI** mohou generovat podle zvoleného typu signálu **isig** periodické funkce: sinus, obdélník (se střídou 1), pilovitý signál a bílý šum s rovnoměrným rozdělením. Amplitudu a frekvenci výstupního signálu **y** určují po řadě parametry **amp** a **freq**. Výstup **y** v případech  $isig \in \{1, 2, 3\}$  může být navíc fázově posunut podle parametru **phase**  $\in (0, 2\pi)$ .

V případě bloku **SGI** je možné navíc synchronizovat počátek generování výstupů u více generátorů **SGI** pomocí vstupů **RUN** a **SYN**. Vstup **SYN** je možné používat za běhu po změně parametrů bloku.

### Vstupy

<b>RUN</b>	Povolení běhu algoritmu, spuštění generátoru	Bool
<b>SYN</b>	Synchronizační signál	Bool

### Výstup

<b>y</b>	Analogový výstupní signál	Double (F64)
----------	---------------------------	--------------

### Parametry

<b>isig</b>	Typ generovaného signálu	⊙1	Long (I32)
	1 ..... sinusový signál		
	2 ..... obdélníkový signál se střídou 1		
	3 ..... pilovitý signál		
	4 ..... bílý šum s rovnoměrným rozdělením		
	5 ..... trojúhelníkový signál		
<b>amp</b>	Amplituda generovaného signálu	⊙1.0	Double (F64)
<b>freq</b>	Frekvence generovaného signálu	⊙1.0	Double (F64)
<b>phase</b>	Fázový posun generovaného signálu		Double (F64)
<b>offset</b>	Hodnota přičítaná k výstupu	⊙1.0	Double (F64)
<b>ifrunit</b>	Jednotky pro frekvenci	⊙1	Long (I32)
	1 ..... Hz		
	2 ..... rad/s		

<code>iphunit</code>	Jednotky pro fázový posun	⊙1 Long (I32)
	1 ..... stupně	
	2 ..... radiány	

## Kapitola 7

# REG – Bloky pro regulaci

### Obsah

---

ARLY – Relé s předstihem . . . . .	197
FLCU – Fuzzy regulátor . . . . .	198
FRID – * Identifikace frekvenční charakteristiky . . . . .	200
I3PM – Identifikace modelu se třemi parametry . . . . .	202
LC – Derivační kompenzátor . . . . .	204
LLC – Integračně-derivační kompenzátor . . . . .	205
LPI – Hodnocení kvality regulační smyčky . . . . .	206
MCU – Jednotka pro ruční zadávání . . . . .	208
PIDAT – PID regulátor s reléovým autotunerem . . . . .	210
PIDE – PID regulátor se statikou . . . . .	213
PIDGS – PID regulátor s přepínáním sad parametrů . . . . .	215
PIDMA – PID regulátor s momentovým autotunerem . . . . .	217
PIDU – PID regulátor . . . . .	223
PIDUI – PID regulátor s parametry na vstupech . . . . .	226
POUT – Pulzní výstup . . . . .	228
PRGM – Programátor . . . . .	229
PSMPC – Prediktivní „pulse-step“ regulátor . . . . .	231
PWM – Blok šířkové modulace . . . . .	235
RLY – Relé s hysterezí . . . . .	237
SAT – Saturace výstupu s proměnnými mezemi . . . . .	238
SC2FA – Stavový regulátor systému 2. řádu s autotunerem . . . . .	240
SCU – Krokový regulátor s polohovou zpětnou vazbou . . . . .	247
SCUV – Krokový regulátor s rychlostním výstupem . . . . .	250
SELU – Selektor aktivního regulátoru . . . . .	254
SMHCC – Regulátor pro procesy s topením a chlazením . . . . .	255
SMHCCA – * Regulátor pro procesy s topením a chlazením s autotunerem . . . . .	258

SWU – Přepínač vstupu pro vysledování . . . . .	260
TSE – Třístavový prvek . . . . .	261

---

Bloky pro regulaci tvoří nejrozsáhlejší podknihovnu knihovny RexLib a zahrnují bloky od jednoduchých dynamických kompenzátorů, přes bloky pro přepínání regulačních struktur, bloky pro přizpůsobení výstupů akčním členům (krokové regulátory, šířková modulace) až po několik verzí PID (P, I, PI, PD a PID) regulátorů. Mezi regulátory jsou například blok **PIDGS**, umožňující za běhu přepínat několik sad parametrů (tzv. *gain scheduling*), **PIDMA** s vestavěným *momentovým autotunerem*, blok **PIDAT** s vestavěným reléovým autotunerem nebo blok fuzzy regulátoru **FLCU**, a další.

## ARLY – Relé s předstihem

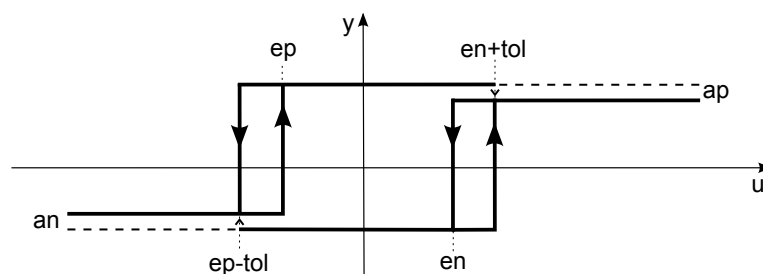
Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok ARLY transformuje vstupní analogový signál  $u$  na výstupní analogový signál  $y$  podle níže uvedeného obrázku.



### Vstup

$u$  Analogový vstupní signál Double (F64)

### Výstup

$y$  Analogový výstupní signál Double (F64)

### Parametry

$ep$	Mez pro přepnutí do stavu „Zapnuto“	$\ominus -1.0$	Double (F64)
$en$	Mez pro přepnutí do stavu „Vypnuto“	$\ominus 1.0$	Double (F64)
$tol$	Toleranční mez pro amplitudu superponovaného šumu vstupu $u$	$\downarrow 0.0 \ominus 0.5$	Double (F64)
$ap$	Hodnota výstupu $y$ ve stavu „Zapnuto“	$\ominus 1.0$	Double (F64)
$an$	Hodnota výstupu $y$ ve stavu „Vypnuto“	$\ominus -1.0$	Double (F64)
$y0$	Počáteční hodnota výstupu $y$ po spuštění		Double (F64)

## FLCU – Fuzzy regulátor

Symbol bloku

Licence: [ADVANCED](#)



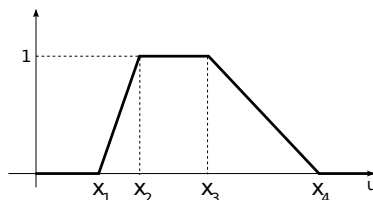
### Popis funkce

Blok FLCU realizuje jednoduchý fuzzy regulátor se dvěma vstupy a jedním výstupem. Dostatečný úvod do problematiky fuzzy řízení je uveden v textu [2].

Funkce bloku je jednoznačně určena lichoběžníkovými funkcemi příslušnosti jazykových výrazů vstupů  $u$  a  $v$ , dále impulsními funkcemi příslušnosti jazykových výrazů výstupu  $y$  a konečně expertními pravidly. Pravidla mají následující tvar:

Jestliže ( $u$  je  $U_i$ ) AND ( $v$  je  $V_j$ ), potom ( $y$  je  $Y_k$ ),

kde  $U_i, i = 1, \dots, nu$  jsou jazykové výrazy příslušné ke vstupu  $u$ ;  $V_j, j = 1, \dots, nv$  jsou jazykové výrazy příslušné ke vstupu  $v$  a  $Y_k, k = 1, \dots, ny$  jsou jazykové výrazy příslušné k výstupu  $y$ . Lichoběžníkové (trojúhelníkové) funkce příslušnosti odpovídající vstupům  $u$  a  $v$  jsou definovány čtyřmi čísly podle následujícího obrázku



U trojúhelníkových funkcí nejsou všechna čísla  $x_1, \dots, x_4$  vesměs různá. Matice funkcí příslušnosti vstupů  $u$  a  $v$  se potom skládají z řádků  $[x_1, x_2, x_3, x_4]$ . Matice  $mf_u$  a  $mf_v$  jsou tedy po řadě typu  $(nu \times 4)$  a  $(nv \times 4)$ .

Impulsní funkce příslušnosti prvního řádu odpovídající výstupu  $y$  se zapisují jako trojice

$$y_k, a_k, b_k,$$

kde  $y_k$  je hodnota výstupu přiřazená jazykovému výrazu  $Y_k, k = 1, \dots, ny$  v případě  $a_k = b_k = 0$ . Je-li  $a_k \neq 0$  a  $b_k \neq 0$ , potom je výrazu  $Y_k$  přiřazena hodnota  $y_k + a_k u + b_k v$ . Matice funkcí příslušnosti výstupu  $sty$  je typu  $(ny \times 3)$  a skládá se po řadě z řádků  $[y_k, a_k, b_k], k = 1, \dots, ny$ .

Soubor pravidel se skládá též jako matice a její řádky jsou  $[i_l, j_l, k_l, w_l], l = 1, \dots, nr$ , kde  $i_l, j_l$  a  $k_l$  označuje jistý jazykový výraz příslušný po řadě vstupu  $u, v$  a výstupu  $y$ . Číslo  $w_l$  udává váhu pravidla v procentech  $w_l \in \{0, 1, \dots, 100\}$ . Tímto způsobem lze jednoduše některé pravidlo zdůraznit, popřípadě vypustit.

## Vstupy

u	První analogový vstup bloku	Double (F64)
v	Druhý analogový vstup bloku	Double (F64)

## Parametry

umax	Horní omezení vstupu u	⊙1.0	Double (F64)
umin	Dolní omezení vstupu u	⊙-1.0	Double (F64)
vmax	Horní omezení vstupu v	⊙1.0	Double (F64)
vmin	Dolní omezení vstupu v	⊙-1.0	Double (F64)
nmax	Počet rezervovaných funkcí příslušnosti (pro každý vstup a výstup)	↓4 ↑10000 ⊙10	Long (I32)
mfu	Matice funkcí příslušnosti – vstup u	⊙[-1 -1 -1 0; -1 0 0 1; 0 1 1 1]	Double (F64)
mfv	Matice funkcí příslušnosti – vstup v	⊙[-1 -1 -1 0; -1 0 0 1; 0 1 1 1]	Double (F64)
sty	Matice funkcí příslušnosti – výstup y	⊙[-1 0 0; 0 0 0; 1 0 0]	Double (F64)
rls	Matice pravidel	⊙[1 2 3 100; 1 1 1 100; 1 0 3 100]	Byte (U8)

## Výstupy

y	Analogový výstupní signál	Double (F64)
ir	Dominantní pravidlo	Long (I32)
wr	Stupeň pravdivosti dominantního pravidla	Double (F64)

## FRID – \* Identifikace frekvenční charakteristiky

Symbol bloku

Licence: [ADVANCED](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

dv	Proměnná dopředné vazby	Double (F64)
pv	Řízená veličina	Double (F64)
ID	Zahájení ladicího experimentu	Bool
HLD	Pozastavení	Bool
BRK	Ukončení ladicího experimentu	Bool

## Parametry

ubias	Stejnoseměrná složka budicího signálu	Double (F64)
uamp	Amplituda budicího signálu	⊙1.0 Double (F64)
wb	Počáteční frekvence [rad/s]	⊙1.0 Double (F64)
wf	Koncová frekvence [rad/s]	⊙10.0 Double (F64)
isweep	Režim rozmítání 1 ..... logaritmické 2 ..... lineární	⊙1 Long (I32)
cp	Rychlost rozmítání	⊙0.995 Double (F64)
iavg	Počet hodnot pro průměrování	⊙10 Long (I32)
obw	Šířka pásma 1 ..... Malá 2 ..... Střední 3 ..... Velká	⊙2 Long (I32)
stime	Doba ustálení [s]	⊙10.0 Double (F64)
umax	Maximální amplituda generátoru	⊙1.0 Double (F64)



thdmin	Minimální požadované zkreslení	⊙0.1	Double (F64)
adapt_rc	Rychlost změny amplitudy rozmítání	⊙0.001	Double (F64)
pv_max	Maximální požadovaná amplituda PV	⊙1.0	Double (F64)
pv_sat	Maximální dovolená amplituda PV	⊙2.0	Double (F64)
ADAPT_EN	Zapnutí adaptace amplitudy generátoru	⊙on	Bool
immode	Režim měření	⊙1	Long (I32)
	1 . . . . . Ruční volba frekvencí		
	2 . . . . . Lineárně nmw frekvencí v intervalu <wb,wf>		
	3 . . . . . Logaritmičticky nmw frekvencí v intervalu <wb,wf>		
	4 . . . . . Automatická detekce důležitých frekvencí (N/A)		
nwm	Počet frekvencí v automatickém režimu		Long (I32)
wm	Pole frekvencí pro ruční režim [array of rad/s]		Double (F64)
		⊙[2.0 4.0 6.0 8.0]	

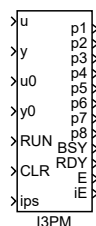
## Výstupy

mv	Akční zásah regulátoru (manipulated variable)		Double (F64)
SAT	Saturace		Bool
IDBSY	Příznak probíhajícího ladicího experimentu		Bool
w	Actuální frekvence [rad/s]		Double (F64)
xres	Reálná složka přenosu (rozmítání)		Double (F64)
xims	Imaginární složka přenosu (rozmítání)		Double (F64)
xrem	Reálná složka přenosu (změřená)		Double (F64)
ximm	Imaginární složka přenosu (změřená)		Double (F64)
epv	Odhad PV		Double (F64)
IDE	Příznak chyby		Bool
iIDE	Kód chyby		Long (I32)
A0	Odhad stejnosměrné složky přenosu		Double (F64)
A1	Odhad amplitudy 1. harmonické		Double (F64)
A2	Odhad amplitudy 2. harmonické		Double (F64)
A3	Odhad amplitudy 3. harmonické		Double (F64)
A4	Odhad amplitudy 4. harmonické		Double (F64)
A5	Odhad amplitudy 5. harmonické		Double (F64)
THD	Celkové harmonické zkreslení		Double (F64)
DAV	Platná data		Bool

## I3PM – Identifikace modelu se třemi parametry

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok I3PM identifikuje tříparametrový model soustavy metodou zobecněných momentů.

### Vstupy

u	Vstup identifikované soustavy	Double (F64)
y	Výstup identifikované soustavy	Double (F64)
u0	Ustálená hodnota vstupu	Double (F64)
y0	Ustálená hodnota výstupu	Double (F64)
RUN	Spuštění identifikace	Bool
CLR	Nulování bloku	Bool
ips	Význam výstupních signálů	Long (I32)
	0 ..... model prvního řádu s dopravním zpožděním	
	p1 ... zesílení	
	p2 ... dopravní zpoždění	
	p3 ... časová konstanta	
	1 ..... momenty vstupu a výstupu	
	p1 ... parametr $mu_0$	
	p2 ... parametr $mu_1$	
	p3 ... parametr $mu_2$	
	p4 ... parametr $my_0$	
	p5 ... parametr $my_1$	
	p6 ... parametr $my_2$	
	2 ..... momenty procesu	
	p1 ... parametr $mp_0$	
	p2 ... parametr $mp_1$	
	p3 ... parametr $mp_2$	
	3 ..... charakteristická čísla procesu	
	p1 ... parametr $\kappa$	
	p2 ... parametr $\mu$	
	p3 ... parametr $\sigma^2$	
	p4 ... parametr $\sigma$	

## Výstupy

$p_i$	Identifikované parametry v závislosti na $ips$ , $i = 1, \dots, 8$	Double (F64)
BSY	Příznak probíhající identifikace	Bool
RDY	Příznak připravenosti	Bool
E	Příznak chyby	Bool
iE	Kód chyby	Long (I32)
	1 ..... předčasné ukončení (RUN = off)	
	2 ..... $\mu_0 = 0$	
	3 ..... $mp_0 = 0$	
	4 ..... $\sigma^2 < 0$	

## Parametry

tident	Délka identifikace [s]	⊙100.0	Double (F64)
irtype	Typ regulátoru	⊙6	Long (I32)
	1 ..... D    3 ..... ID    5 ..... PD    7 ..... PID		
	2 ..... I    4 ..... P    6 ..... PI		
ispeed	Požadovaná rychlost uzavřené smyčky	⊙2	Long (I32)
	1 ..... pomalá uzavřená smyčka		
	2 ..... středně rychlá uzavřená smyčka		
	3 ..... rychlá uzavřená smyčka		

## LC – Derivační kompenzátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LC realizuje diskretní simulátor přenosu derivačního članku

$$C(s) = \frac{td * s}{\frac{td}{nd} * s + 1},$$

kde  $td$  je derivační konstanta a  $nd$  je parametr určující vliv parazitního filtru prvního řádu. Doporučená hodnota  $nd$  je  $2 \leq nd \leq 10$ . Je-li  $ISSF = on$ , potom je stav parazitního filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu  $u$ .

Pro diskretizaci přenosu  $C(s)$  je použita přesná diskretizace v okamžicích vzorkování.

Vstup

$u$	Analogový vstupní signál	Double (F64)
$R1$	Reset (stav jako po initu)	Bool
$HLD$	Pozastavení vykonávání bloku	Bool

Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------

Parametry

$td$	Derivační časová konstanta	⊙1.0	Double (F64)
$nd$	Parametr filtru derivační složky	⊙10.0	Double (F64)
$ISSF$	Ustálený stav při spuštění		Bool
	off ... nulový počáteční stav		
	on .... ustálený počáteční stav		

## LLC – Integračně-derivační kompenzátor

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok LLC realizuje diskretní simulátor přenosu integračně-derivačního članku

$$C(s) = \frac{a * \text{tau} * s + 1}{\text{tau} * s + 1},$$

kde  $\text{tau}$  je časová konstanta jmenovatele a její  $a$ -násobek ( $a * \text{tau}$ ) je časová konstanta čitatele. Je-li  $\text{ISSF} = \text{on}$ , potom je stav integračního članku nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu  $u$ .

Pro diskretizaci přenosu  $C(s)$  je použita přesná diskretizace v okamžicích vzorkování.

### Vstup

$u$	Analogový vstupní signál	Double (F64)
$R1$	Reset (stav jako po initu)	Bool
$HLD$	Pozastavení vykonávání bloku	Bool

### Parametry

$\text{tau}$	Časová konstanta	⊙1.0	Double (F64)
$a$	Koeficient pro výpočet časové konstanty čitatele		Double (F64)
$\text{ISSF}$	Ustálený stav při spuštění off ... nulový počáteční stav on ... ustálený počáteční stav		Bool

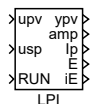
### Výstup

$y$	Analogový výstupní signál	Double (F64)
-----	---------------------------	--------------

## LPI – Hodnocení kvality regulační smyčky

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok LPI (*Loop Performance Index*) je navržen pro ohodnocení kvality zpětné regulace tím, že ovlivňuje hodnotu signálu před jeho přivedením do regulátoru a měří odezvu systému. Tento blok je užitečný pro analýzu a identifikaci chování regulační smyčky v reálném čase.

Procesní veličina **pv** regulační smyčky je připojena na vstup **upv** a z výstupu **ypv** do regulátoru. Požadovaná veličina **sp** regulační smyčky je přivedena na vstup **usp**. Blok se aktivuje signálem **RUN** pouze v automatickém režimu regulátoru, když je žádoucí provést identifikaci regulační smyčky.

Při aktivaci (**RUN**=1) blok LPI injektuje do procesní veličiny sinusový signál o definované amplitudě **ad** a frekvenci **fd**, což umožňuje měření odezvy systému. Výstupní signál je dále zpracován BandPass filtrem a Fourierovou transformací pro určení průměrné amplitudy signálu. Výsledný index výkonosti **Ip** je vypočítán na základě poměru mezi nastavenými parametry a naměřenou amplitudou, což poskytuje kvantitativní hodnocení potlačení výstupních poruch regulačním systémem.

Výstup **Ip** reflektuje, jak efektivně regulační systém potlačuje poruchy v definovaném frekvenčním pásmu **fa**. Hodnota **Ip**=1 naznačuje, že systém potlačuje poruchy v souladu s očekáváním; hodnoty vyšší než 1 ukazují na lepší výkon; nižší hodnoty indikují horší nastavení regulační smyčky.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>upv</b>	Vstup řízené veličiny	Double (F64)
<b>usp</b>	Vstup požadované hodnoty	Double (F64)
<b>RUN</b>	Povolení běhu algoritmu	Bool

### Parametr

<b>ms</b>	Horní omezení citlivostní funkce	↓1.00001 ↑1000.0 ⊙2.0	Double (F64)
<b>fa</b>	Přípustná šířka pásma	↓1e-10 ↑1e+10 ⊙10.0	Double (F64)
<b>fd</b>	Excitovaná/měřená frekvence	↓1e-10 ↑1e+10 ⊙1.0	Double (F64)
<b>ad</b>	Amplituda excitace	⊙0.01	Double (F64)
<b>nper</b>	Velikost okna (počet period fd)	↓1 ⊙4	Long (I32)

<code>ifrunit</code>	Jednotky pro frekvenci 1 ..... Hz 2 ..... rad/s	⊙1	Long (I32)
<code>xi</code>	Koeficient tlumení filtru	↓0.001 ↑100.0 ⊙1.0	Double (F64)
<code>nmax</code>	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙256	Long (I32)

## Výstup

<code>ypv</code>	Výstup řízené veličiny	Double (F64)
<code>amp</code>	Amplituda signálu po filtraci	Double (F64)
<code>Ip</code>	Výkonnost regulační smyčky	Double (F64)
<code>E</code>	Příznak chyby	Bool
<code>iE</code>	Kód chyby	Error

## MCU – Jednotka pro ruční zadávání

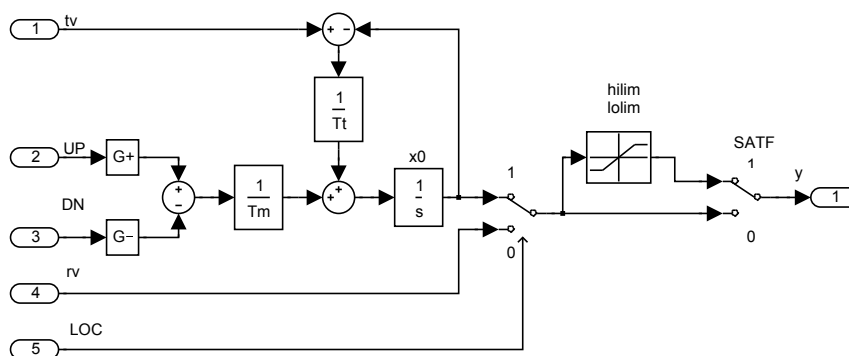
Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

V lokálním režimu ( $LOC = on$ ) je blok MCU určen k ručnímu zadávání výstupu  $y$  pomocí tlačítek „více“ (vstup UP) a „méně“ (vstup DN). Strmost najíždění z počáteční hodnoty  $y_0$  na žádanou hodnotu je určena integrační konstantou  $t_m$  a dobou stlačení ovládacích tlačítek. Po uplynutí každých  $t_a$  sekund je strmost vždy násobena faktorem  $q$ , až do vypršení doby  $t_f$ . Rozsah výstupu  $y$  může být omezen ( $SATF = on$ ) saturačními mezemi  $lolim$  a  $hilim$ . V případě, že žádné z tlačítek není stlačeno ( $UP = off$  a  $DN = off$ ), vysleduje výstup  $y$  vstupní hodnotu  $tv$ . Rychlost vysledování je dána integrační časovou konstantou  $tt$ . V případě  $LOC = off$  je vstup  $rv$  s případnými omezeními ( $SATF = on$ ) kopírován na výstup  $y$ . Podrobná funkce bloku je přímo patrná z obrázku s vnitřním schématem bloku.



### Vstupy

tv	Veličina pro vysledování	Double (F64)
UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool
rv	Hodnota pro externí zadávání výstupu v režimu $LOC = off$	Double (F64)
LOC	Lokální nebo vzdálený režim	Bool



## Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

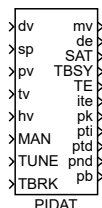
## Parametry

tt	Časová konstanta výsledování vstupní hodnoty tv	⊙1.0	Double (F64)
tm	Počáteční časová konstanta strmosti najíždění	⊙100.0	Double (F64)
y0	Počáteční hodnota výstupu		Double (F64)
q	Faktor určující velikost změny strmosti najíždění	⊙5.0	Double (F64)
ta	Interval, po kterém dochází ke zvýšení strmosti [s]	⊙4.0	Double (F64)
tf	Interval, po kterém se strmost již dále nemění [s]	⊙8.0	Double (F64)
SATF	Saturace		Bool
	off ... signál není omezen      aktivní		
	on .... saturační    meze    jsou		
hilim	Horní saturační mez	⊙1.0	Double (F64)
lolim	Dolní saturační mez	⊙-1.0	Double (F64)

## PIDAT – PID regulátor s reléovým autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



### Popis funkce

Blok PIDAT má zcela stejné regulační funkce jako blok [PIDU](#). Navíc je vybaven funkcí automatického nastavování parametrů regulátoru. Pro využití této funkce je nutné převést řízený systém do přibližně ustáleného stavu (ve vhodném pracovním bodě), zvolit požadovaný typ regulátoru (PI nebo PID) a aktivovat vstup TUNE hodnotou `on` (start identifikačního experimentu). V následném identifikačním experimentu je řízený proces regulován pomocí speciálního adaptivního reléového regulátoru a ze získaného záznamu vstupu a výstupu procesu je odhadnut vhodný bod jeho frekvenční charakteristiky. Na základě toho jsou poté určeny parametry regulátoru. Amplitudu reléového regulátoru (úroveň vybuzení systému) je možné nastavit parametrem `amp` a jeho hysterezi parametrem `hys`. Zvolíme-li `hys = 0`, potom se hystereze relé určí automaticky na základě odhadu úrovně šumu měření regulované veličiny. Během identifikačního experimentu je `TBSY = on`. Po řádném skončení experimentu je `TE = off` a vypočítané parametry se objeví na výstupech `pk`, `pti`, `ptd`, `pnd`, `pb`. Váhový koeficient `c` je uvažován `c = 0`. Skončil-li experiment s chybou, je `TE = on` a `ite` blíže specifikuje důvod chyby. Při výskytu chyby se doporučuje zvětšit parametr `amp`. Jeho volbu usnadňuje zabudovaná funkce, která parametr `amp` automaticky zmenšuje při hrozbě překročení maximální dovolené odchylky `maxdev` regulované veličiny od jejího počátečního ustáleného stavu. Identifikační experiment je možné předčasně ukončit aktivací vstupu TBRK.

### Vstupy

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on ... manuální režim	Bool
TUNE	Zahájení ladicího experimentu	Bool

TBRK	Ukončení ladicího experimentu	Bool
------	-------------------------------	------

## Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on ... výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladicího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... Ladění proběhlo bez chyby	
	on ... Během ladění se vyskytla chyba	
ite	Kód chyby (během probíhajícího ladicího experimentu očekávaný čas v sekundách do jeho konce)	Long (I32)
	1000 .. příliš nízký poměr užitečného signálu k šumu měření	
	1001 .. příliš velká hystereze reléového regulátoru	
	1002 .. příliš přísné pravidlo pro ukončení	
	1003 .. příliš velká chyba při určování fáze identifikovaného bodu	
pk	Navržené zesílení regulátoru	Double (F64)
pti	Navržená integrační časová konstanta regulátoru	Double (F64)
ptd	Navržená derivační časová konstanta regulátoru	Double (F64)
pnd	Navržený parametr filtru derivační složky	Double (F64)
pb	Navržený váhový faktor pro proporcionální složku	Double (F64)

## Parametry

irtype	Typ regulátoru	⊙6 Long (I32)
	1 ..... D      4 ..... P      7 ..... PID	
	2 ..... I      5 ..... PD	
	3 ..... ID      6 ..... PI	
RACT	Převrácené působení výstupu regulátoru	Bool
	off ... vyšší mv → vyšší pv	
	on ... vyšší mv → nižší pv	
k	Zesílení regulátoru $K$ . Hodnota 0 (dle definice) vypne regulátor, záporné hodnoty nejsou dovoleny (k tomu slouží parametr RACT). ↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta $T_i$ . Hodnota 0 znamená vypnutí integrační složky regulátoru (stejný efekt jako vypnutí parametrem irtype). ↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta $T_d$ . Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem irtype). ↓0.0 ⊙1.0	Double (F64)
nd	Parametr $N$ filtru derivační složky. Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem irtype). ↓0.0 ⊙10.0	Double (F64)

<b>b</b>	Váhový faktor pro proporcionální složku	↓0.0 ⊙1.0	Double (F64)
<b>c</b>	Váhový faktor pro derivační složku	↓0.0	Double (F64)
<b>tt</b>	Časová konstanta výsledování.	↓0.0 ⊙1.0	Double (F64)
<b>hilim</b>	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
<b>lolim</b>	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
<b>iainf</b>	Druh apriorní informace	⊙1	Long (I32)
	1 ..... žádná apriorní informace		
	2 ..... astatický proces		
	3 ..... proces nízkého řádu		
	4 ..... statický proces + požadavek na aperiodickou odezvu uzavřené smyčky		
	5 ..... statický proces + požadavek na středně rychlou odezvu uzavřené smyčky		
	6 ..... statický proces + požadavek na rychlou odezvu uzavřené smyčky		
<b>k0</b>	Statické zesílení procesu (musí být zadáno v případě <b>iainf</b> = 3, 4, 5)	⊙1.0	Double (F64)
<b>n1</b>	Maximální počet půlperiod pro nalezení bodu frekvenční charakteristiky	⊙20	Long (I32)
<b>mm</b>	Maximální počet půlperiod pro průměrování	⊙4	Long (I32)
<b>amp</b>	Amplituda reléového regulátoru	⊙0.1	Double (F64)
<b>uhys</b>	Hystereze reléového regulátoru		Double (F64)
<b>ntime</b>	Čas vymezený pro odhad amplitudy šumu na počátku experimentu [s]	⊙5.0	Double (F64)
<b>rerrap</b>	Ukončovací hodnota relativní chyby amplitudy kmitů	⊙0.1	Double (F64)
<b>aerrph</b>	Ukončovací hodnota absolutní chyby fáze odhadovaného bodu	⊙10.0	Double (F64)
<b>maxdev</b>	Maximální přípustná odchylka regulované veličiny od ustáleného stavu	⊙1.0	Double (F64)

Parametry **n1**, **mm**, **ntime**, **rerrap** a **aerrph** se nedoporučuje měnit.

## PIDE – PID regulátor se statikou

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok PIDE je základní blok pro vytvoření úplného modifikovaného regulátoru PI(D), který se liší od standardního PI(D) regulátoru (blok [PIDU](#)) tím, že má zadané konečné statické zesílení (ve skutečnosti se zadává velikost odchylky  $\varepsilon$ , která způsobí saturaci výstupu). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci modifikovaného PID regulátoru s dvěma stupni volnosti v automatickém (MAN = off) nebo manuálním režimu (MAN = on).

V automatickém režimu v lineární oblasti realizuje zákon řízení daný vztahem

$$U(s) = \pm K \left[ bW(s) - Y(s) + \frac{1}{T_i s + \beta} E(s) + \frac{T_d s}{\frac{T_d s}{N} + 1} (cW(s) - Y(s)) \right] + Z(s),$$

kde

$$\beta = \frac{K\varepsilon}{1 - K\varepsilon}$$

a  $U(s)$  je obraz akční veličiny  $mv$ ,  $W(s)$  je obraz požadované hodnoty  $sp$ ,  $Y(s)$  je obraz regulované veličiny  $pv$ ,  $E(s)$  je Laplaceova transformace regulační odchylky,  $Z(s)$  je obraz dopředné vazby  $dv$  a  $K, T_i, T_d, N, \varepsilon$  ( $= b_p/100$ ),  $b, c$  jsou parametry regulátoru. Znaménko pravé strany závisí na parametru RACT. Rozsah řídicí veličiny  $mv$  je omezen saturačními mezemi  $lolim$  a  $hilim$ . Propojením výstupu  $mv$  se vstupem  $tv$  a vhodnou volbou parametru  $tt$  dosáhneme žádaného chování regulátoru při dosažení saturačních hodnot  $mv$ . Odstraníme tak nežádoucí unášení integrační složky (wind up effect) a současně s tím zajistíme bezrázové přepínání (bumpless transfer) automatického a manuálního režimu.

V manuálním režimu je vstup  $hv$  (po případném omezení) kopírován na výstup  $mv$ . Signál připojený na vstup  $tv$  zajišťuje v tomto režimu příslušné vysledování vnitřního stavu regulátoru pro následné bezrázové přepnutí do automatického režimu (pro  $\varepsilon > 0$  však vysledování není zcela přesné).

### Vstupy

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)

hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim	Bool

### Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace off ... lineární zákon řízení on .... výstup regulátoru je saturován	Bool

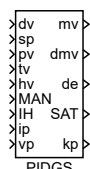
### Parametry

irtype	Typ regulátoru 1 ..... D      4 ..... P      7 ..... PID 2 ..... I      5 ..... PD 3 ..... ID      6 ..... PI	⊙6	Long (I32)
RACT	Převrácené působení výstupu regulátoru off ... vyšší mv → vyšší pv on .... vyšší mv → nižší pv		Bool
k	Zesílení regulátoru $K$	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta $T_i$	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta $T_d$	↓0.0 ⊙1.0	Double (F64)
nd	Parametr $N$ filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0	Double (F64)
tt	Časová konstanta výsledování. Pro regulátory bez integrační složky nemá žádný význam.	↓0.0 ⊙1.0	Double (F64)
bp	Hodnota regulační odchylky, která způsobí, že výstup regulátoru mv v ustáleném stavu je roven hodnotě 100		Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)

## PIDGS – PID regulátor s přepínáním sad parametrů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Regulační funkce bloku PIDGS je přesně shodná s blokem PIDU. Blok PIDGS má však až šest sad základních parametrů, které je možné bezrázově přepínat pomocí vstupu ip (index sady parametrů) nebo vstupu vp (přepínací analogová veličina). V případě použití přepínací analogové veličiny je třeba zadat  $GSCF = on$  a vektor příslušných přepínacích mezí  $thrsha$ . Sady parametrů jsou poté přepínány takto: sada 0 je pro  $vp < thrsha(0)$ , sada 1 pro  $thrsha(0) < vp < thrsha(1)$  atd. až sada 5 pro  $thrsha(4) < vp$ . Index aktuální sady je k dispozici na výstupu kp.

### Vstupy

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim	Bool
IH	Zastavení integrace off ... integrování povoleno on .... integrování pozastaveno	Bool
ip	Index sady parametrů	↓0 ↑5 Long (I32)
vp	Přepínací veličina	Double (F64)

### Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (diference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace off ... lineární zákon řízení on .... výstup regulátoru je saturován	Bool

kp Index aktuální sady parametrů Long (I32)

## Parametry

hilim Horní mez akčního zásahu regulátoru  $\odot 1.0$  Double (F64)  
 lolim Dolní mez akčního zásahu regulátoru  $\odot -1.0$  Double (F64)  
 dz Pásmo necitlivosti Double (F64)  
 icotype Typ výstupu regulátoru  $\odot 1$  Long (I32)  
     1 ..... analogový výstup  
     2 ..... šířkově modulovaný výstup (PWM)  
     3 ..... krokový regulátor s polohovou zpětnou vazbou (SCU)  
     4 ..... krokový regulátor bez polohové zpětné vazby (SCUV)

nmax Alokovaný počet sad parametrů  $\downarrow 4 \uparrow 10000 \odot 10$  Long (I32)  
 GSCF Přepínání parametrů podle vstupu vp Bool  
     off ... přepínání indexem sady parametrů  
     on .... přepínání analogovým signálem

hys Hystereze pro přepínání podle vstupu vp Double (F64)  
 irtypea Vektor typů regulátoru  $\odot [6 \ 6 \ 6 \ 6 \ 6 \ 6]$  Byte (U8)  
     1 ..... D      4 ..... P      7 ..... PID  
     2 ..... I      5 ..... PD  
     3 ..... ID     6 ..... PI

RACTA Vektor příznaků obráceného působení výstupu regulátoru Bool  
      $\odot [0 \ 0 \ 0 \ 0 \ 0 \ 0]$   
     0 ..... vyšší mv  $\rightarrow$  vyšší pv  
     1 ..... vyšší mv  $\rightarrow$  nižší pv

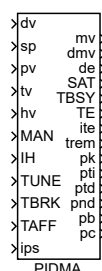
ka Vektor zesílení regulátoru  $K \odot [1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0]$  Double (F64)  
 tia Vektor integračních časových konstant  $T_i \odot [4.0 \ 4.0 \ 4.0 \ 4.0 \ 4.0 \ 4.0]$  Double (F64)  
 tda Vektor derivačních časových konstant  $T_d \odot [1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0]$  Double (F64)  
 nda Vektor parametrů filtru derivační složky  $N \odot [10.0 \ 10.0 \ 10.0 \ 10.0 \ 10.0 \ 10.0]$  Double (F64)  
 ba Váhové faktory pro proporcionální složku  $\odot [1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0]$  Double (F64)  
 ca Váhové faktory pro derivační složku  $\odot [0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$  Double (F64)  
 tta Vektor časových konstant vysledování  $\odot [1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0 \ 1.0]$  Double (F64)  
 thrsha Vektor mezí přepínací veličiny  $\odot [0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0]$  Double (F64)



## PIDMA – PID regulátor s momentovým autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



### Popis funkce

V automatickém režimu ( $\text{MAN} = \text{off}$ ) realizuje blok PIDMA řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N} s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

kde  $U(s)$  je Laplaceova transformace řídicí veličiny  $mv$ ,  $W(s)$  je Laplaceova transformace požadované hodnoty  $sp$ ,  $Y(s)$  je Laplaceova transformace regulované veličiny  $pv$ ,  $Z(s)$  je Laplaceova transformace dopředné vazby  $dv$  a  $K$ ,  $T_i$ ,  $T_d$ ,  $N$ ,  $b$ ,  $c$  jsou parametry regulátoru. Znaménko pravé strany závisí na parametru  $\text{RACT}$ . Rozsah řídicí veličiny  $mv$  (polohového výstupu regulátoru) je omezen parametry  $\text{hilim}$ ,  $\text{lolim}$ . Parametr  $\text{dz}$  udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem  $\text{IH} = \text{on}$ . Pro správnou funkci regulátoru je nutné propojit výstup regulátoru  $mv$  se vstupem  $tv$  a správně nastavit časovou konstantu vysledování  $tt$

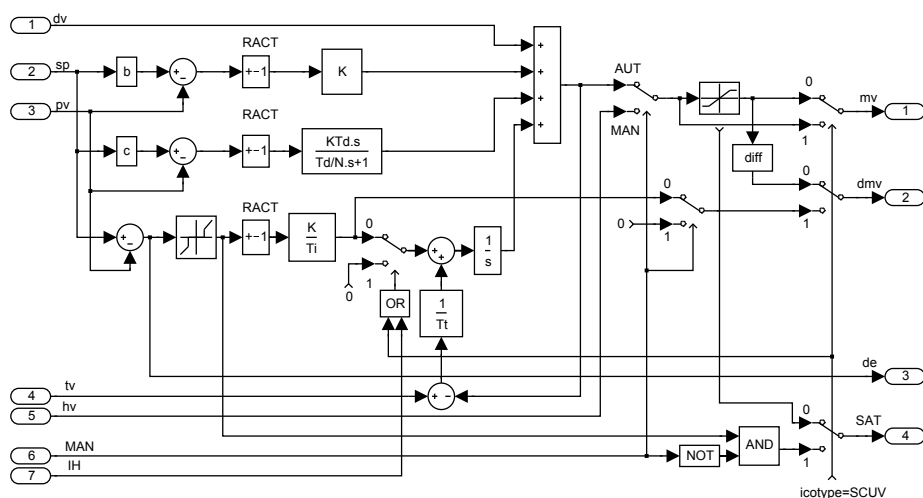
Doporučená výchozí hodnota pro PID regulátor je  $tt \approx \sqrt{T_i T_d}$ , pro PI regulátor pak  $tt \approx T_i/2$ . Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu  $mv$  (tzv. antiwindup). Úpravou parametru  $tt$  je v případě potřeby možné nastavit přesné chování v saturaci (tzv. odskakování od saturace vlivem šumu) a při přepínání více regulátorů (velikost skoku při přepnutí, pokud není nulová regulační odchylka).

Přídavné výstupy  $dmv$ ,  $de$  a  $\text{SAT}$  poskytují po řadě rychlostní výstup regulátoru (diference  $mv$ ), regulační odchylku a příznak saturace výstupu regulátoru  $mv$ .

Jestliže je blok PIDMA propojen s blokem  $\text{SCUV}$  (za účelem realizace krokového regulátoru bez polohové zpětné vazby), potom parametr  $\text{icotype}$  musí být nastaven na hodnotu 4 a význam výstupů  $mv$ ,  $dmv$  a  $\text{SAT}$  je v tomto případě pozměněn: výstup  $mv$  je roven součtu P a D složky regulátoru, zatímco výstup  $dmv$  poskytuje diferenci jeho I

složky a výstup **SAT** nese informaci pro blok **SCUV**, zda je regulační odchylka **de** v automatickém režimu menší než pásmo necitlivosti **dz**. Pro případ propojení bloků **PIDMA** a **SCUV** se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku **c** rovný nule.

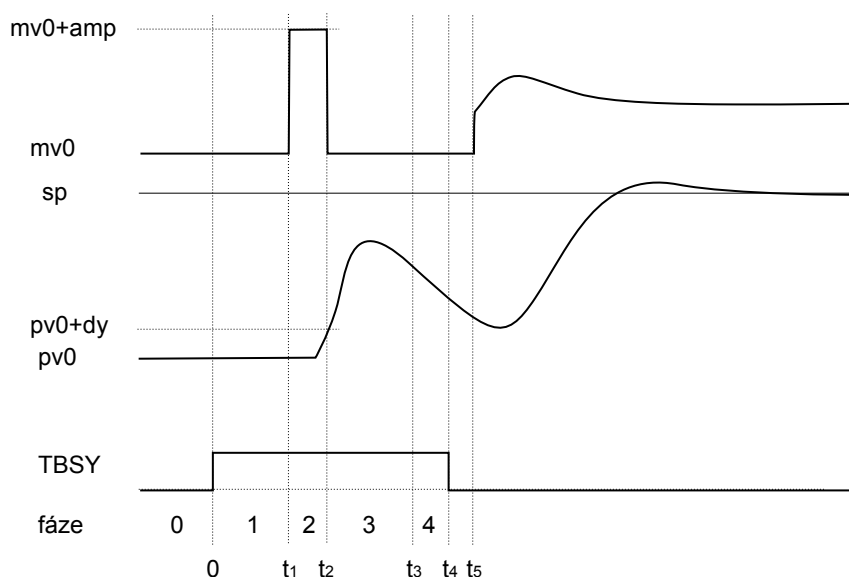
V manuálním režimu (**MAN = on**) je vstup **hv** kopírován na výstup **mv**. Celková regulační funkce bloku **PIDMA** je zřejmá z následujícího obrázku.



Blok **PIDMA** rozšiřuje řídicí funkci standardního PID regulátoru o vestavěné automatické nastavování parametrů (PID autotuner). Před spuštěním autotuneru musí operátor ve vhodném pracovním bodě dosáhnout ustáleného stavu a zvolit požadovaný typ regulátoru **ittype** (PI nebo PID) a nastavit další parametry autotuneru (**iainf**, **DGC**, **tdg**, **tn**, **amp**, **dy** a **ispeed**). Identifikační experiment se startuje vstupem **TUNE** (vstupem **TBRK** jej lze předčasně ukončit). V tomto módu (**TBSY = on**) je nejprve odhadnut drift a šum regulované veličiny (ve specifikovaném čase **tdg+tn**) a poté je na vstup procesu aplikován pravoúhlý puls. Z odezvy procesu jsou odhadnuty první tři momenty jeho impulsní odezvy. Amplituda pulsu se nastavuje parametrem **amp**. Puls je ukončen poté, co se hodnota regulované veličiny **pv** změní o více, než určuje tolerance (práh) **dy** (zadáva se vždy jako kladné číslo). Pokud je nastaven příznak **DGC**, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajcího do konce procesu ladění je přiveden na výstup **trem**.

Pokud experiment skončí úspěšně (**TE = off**) a vstup **ips = 0**, objeví se optimální parametry na výstupech **pk**, **pti**, **ptd**, **pnd**, **pb**, **pc**. V opačném případě (**TE = on**) určuje výstup **ite** kód chyby experimentu. Další hodnoty vstupu **ips** jsou rezervovány pro speciální účely.

Funkce autotuneru je demonstrována na následujícím obrázku.



Během identifikačního experimentu výstup `ite` indikuje jednotlivé fáze činnosti autotuneru. Ve fázi odhadu strmosti odezvy (odezvy) (`ite = -4`) může být proces ladění předčasně manuálně ukončen. V tomto případě jsou parametry regulátoru řádně navrženy, avšak jejich možná nepřesnost je indikována varovným kódem `ite = 100`.

Po ukončení experimentu (`TBSY on` → `off`) je funkce regulátoru závislá na nastaveném režimu (manuální, automatický). Jestliže `TAFF = on`, potom jsou navržené parametry okamžitě použity.

## Vstupy

<code>dv</code>	Proměnná dopředné vazby	Double (F64)
<code>sp</code>	Požadovaná hodnota (setpoint)	Double (F64)
<code>pv</code>	Řízená veličina	Double (F64)
<code>tv</code>	Velichina pro vysledování	Double (F64)
<code>hv</code>	Hodnota výstupu v manuálním režimu	Double (F64)
<code>MAN</code>	Manuální nebo automatický režim <code>off</code> ... automatický režim <code>on</code> ... manuální režim	Bool
<code>IH</code>	Zastavení integrace <code>off</code> ... integrování povoleno <code>on</code> ... integrování pozastaveno	Bool
<code>TUNE</code>	Zahájení ladicího experimentu nebo vynucení přechodu do další fáze experimentu	Bool
<code>TBRK</code>	Ukončení ladicího experimentu	Bool
<code>TAFF</code>	Přijetí výsledků ladicího experimentu <code>off</code> ... parametry jsou pouze vypočítány <code>on</code> ... parametry jsou dosazeny do řídicího algoritmu	Bool

ips	Význam výstupních signálů $pk$ , $pti$ , $ptd$ , $pnd$ , $pb$ a $pc$	Long (I32)
	0 ..... navržené parametry $k$ , $ti$ , $td$ , $nd$ , $b$ a $c$ PID regulátoru	
	1 ..... momenty procesu: zesílení ( $pk$ ), míra zpoždění soustavy ( $pti$ ), míra délky odezvy soustavy ( $ptd$ )	
	2 ..... tříparametrový model procesu prvního řádu s dopravním zpožděním: zesílení ( $pk$ ), dopravní zpoždění ( $pti$ ), časová konstanta ( $ptd$ )	
	3 ..... tříparametrový model procesu druhého řádu s násobnou časovou konstantou a dopravním zpožděním: zesílení ( $pk$ ), dopravní zpoždění ( $pti$ ), časová konstanta ( $ptd$ )	
	4 ..... odhad mezí intervalu pro manuální doladění zesílení k PID regulátoru ( $irtype = 7$ ): horní mez $k_{hi}$ ( $pk$ ), dolní mez $k_{lo}$ ( $pti$ )	
	>99 ... slouží pro diagnostické účely	

## Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (diference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on ... výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladicího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... Ladění proběhlo bez chyby	
	on ... Během ladění se vyskytla chyba	
ite	Kód chyby	Long (I32)
	<i>Kódy chyb ladění (po experimentu):</i>	
	0 ..... bez chyby	
	1 ..... příliš malá hodnota prahu pro ukončení pulzu	
	2 ..... příliš velká amplituda pulzu	
	3 ..... nebylo dosaženo ustáleného stavu	
	4 ..... příliš malá amplituda pulzu	
	5 ..... nebylo dosaženo ustáleného stavu	
	6 ..... při experimentu došlo k saturaci výstupu regulátoru	
	7 ..... pro vybraný typ regulátoru není podporováno automatické nastavování	
	8 ..... nedodržena podmínka monotónnosti procesu	
	9 ..... selhání extrapolace	
	10 ... neočekávané hodnoty momentů (fatální)	
	11 ... ruční přerušování experimentu uživatelem	
	12 ... nesprávný směr řídicí veličiny (změňte parametr RACT)	
	100 ... ruční ukončení ladění (varování)	

*Kódy fází ladění (během experimentu):*

- 0 . . . . čekání na ustálený stav před začátkem experimentu
- 1 . . . . odhad driftu a šumu (parametry *tdg* a *tn*)
- 2 . . . . generování obdélníkového pulzu (pulz končí při změně *pv* o hodnotu větší než *dy*)
- 3 . . . . hledání vrcholu odezvy
- 4 . . . . odhad rychlosti ustalování odezvy

*Poznámka k ukončování fází ladění:*

TUNE .. Náběžná hrana vstupu TUNE během fází -2, -3 and -4 způsobuje předčasné ukončení dané fáze a přechod do fáze následující (nebo ukončení experimentu ve fázi -4).

<i>trem</i>	Odhad času do ukončení experimentu [s]	Double (F64)
<i>pk</i>	Navržené zesílení regulátoru $K$ ( <i>ips</i> = 0)	Double (F64)
<i>pti</i>	Navržená integrační časová konstanta regulátoru $T_i$ ( <i>ips</i> = 0)	Double (F64)
<i>ptd</i>	Navržená derivační časová konstanta regulátoru $T_d$ ( <i>ips</i> = 0)	Double (F64)
<i>pnd</i>	Navržený parametr filtru derivační složky $N$ ( <i>ips</i> = 0)	Double (F64)
<i>pb</i>	Navržený váhový faktor pro proporcionální složku ( <i>ips</i> = 0)	Double (F64)
<i>pc</i>	Navržený váhový faktor pro derivační složku ( <i>ips</i> = 0)	Double (F64)

## Parametry

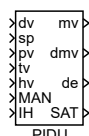
<i>irtype</i>	Typ regulátoru 1 . . . . D 3 . . . . ID 5 . . . . PD 7 . . . . PID 2 . . . . I 4 . . . . P 6 . . . . PI	⊙6 Long (I32)
<i>RACT</i>	Převrácené působení výstupu regulátoru <i>off</i> . . . vyšší <i>mv</i> → vyšší <i>pv</i> <i>on</i> . . . . vyšší <i>mv</i> → nižší <i>pv</i>	Bool
<i>k</i>	Zesílení regulátoru $K$ . Hodnota 0 (dle definice) vypne regulátor, záporné hodnoty nejsou dovoleny (k tomu slouží parametr <i>RACT</i> ). ↓0.0 ⊙1.0	Double (F64)
<i>ti</i>	Integrační časová konstanta $T_i$ . Hodnota 0 znamená vypnutí integrační složky regulátoru (stejný efekt jako vypnutí parametrem <i>irtype</i> ). ↓0.0 ⊙4.0	Double (F64)
<i>td</i>	Derivační časová konstanta $T_d$ . Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem <i>irtype</i> ). ↓0.0 ⊙1.0	Double (F64)
<i>nd</i>	Parametr $N$ filtru derivační složky. Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem <i>irtype</i> ). ↓0.0 ⊙10.0	Double (F64)
<i>b</i>	Váhový faktor pro proporcionální složku ↓0.0 ↑2.0 ⊙1.0	Double (F64)
<i>c</i>	Váhový faktor pro derivační složku ↓0.0 ↑2.0	Double (F64)

tt	Časová konstanta vysledování; hodnota 0 znamená implicitní hodnotu, což je $T_i/2$ pro regulátor s integrační složkou a vypnutí vysledování pro regulátor bez integrační složky. Pokud pro P nebo PD regulátor potřebujeme vysledování (tzv. regulace kolem rovnovážného bodu), zapneme vysledování nastavením kladné hodnoty (větší než perioda vzorkování). Vypnout vysledování pro regulátor s integrační složkou není možné (kvůli windup efektu). ↓0.0 ⊕1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊕1.0 Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊖-1.0 Double (F64)
dz	Pásmo necitlivosti	Double (F64)
icotype	Typ výstupu regulátoru 1 ..... analogový výstup 2 ..... šířkově modulovaný výstup (PWM) 3 ..... krokový regulátor s polohovou zpětnou vazbou (SCU) 4 ..... krokový regulátor bez polohové zpětné vazby (SCUV)	⊕1 Long (I32)
itttype	Požadovaný typ regulátoru pro návrh 6 ..... PI regulátor 7 ..... PID regulátor	⊕6 Long (I32)
iainf	Druh apriorní informace 1 ..... proces bez integrátoru 2 ..... proces s integračním chováním	⊕1 Long (I32)
DGC	Kompenzace gradientu trendu off ... zakázáno      on .... povoleno	⊕on Bool
tdg	Doba odhadu gradientu trendu [s]	⊕60.0 Double (F64)
tn	Doba odhadování šumu [s]	⊕5.0 Double (F64)
amp	Amplituda pulzu	⊕0.5 Double (F64)
dy	Práh pro ukončení pulsu (absolutní změna od ustálené hodnoty pv) ↓0.0 ⊕0.1	Double (F64)
ispeed	Požadovaná rychlost uzavřené smyčky 1 ..... požadována pomalá uzavřená smyčka 2 ..... požadována středně rychlá uzavřená smyčka 3 ..... požadována rychlá uzavřená smyčka	⊕2 Long (I32)
ipid	Forma PID regulátoru 1 ..... paralelní realizace 2 ..... sériová realizace	⊕1 Long (I32)

## PIDU – PID regulátor

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **PIDU** je základní blok pro vytvoření úplného regulátoru PID (P, I, PI, PD, PID, PI+S). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci PID regulátoru se dvěma stupni volnosti v automatickém (**MAN = off**) nebo manuálním režimu (**MAN = on**).

V automatickém režimu (**MAN = off**) realizuje blok **PIDU** řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N} s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

kde  $U(s)$  je Laplaceova transformace řídicí veličiny  $mv$ ,  $W(s)$  je Laplaceova transformace požadované hodnoty  $sp$ ,  $Y(s)$  je Laplaceova transformace regulované veličiny  $pv$ ,  $Z(s)$  je Laplaceova transformace dopředné vazby  $dv$  a  $K$ ,  $T_i$ ,  $T_d$ ,  $N$ ,  $b$ ,  $c$  jsou parametry regulátoru. Znaménko pravé strany závisí na parametru **RACT**. Rozsah řídicí veličiny  $mv$  (polohového výstupu regulátoru) je omezen parametry **hilim**, **lolim**. Parametr **dz** udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem **IH** (**IH = on**). Pro správnou funkci regulátoru je nutné propojit výstup regulátoru  $mv$  se vstupem **tv** a správně nastavit časovou konstantu vysledování **tt**.

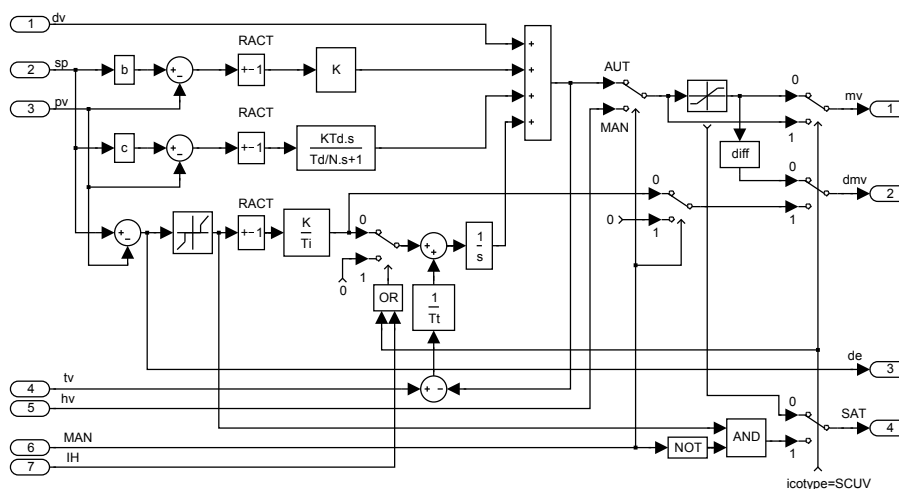
Doporučená výchozí hodnota pro PID regulátor je  $tt \approx \sqrt{T_i T_d}$ , pro PI regulátor pak  $tt \approx T_i/2$ . Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu  $mv$  (tzv. antiwindup). Úpravou parametru **tt** je v případě potřeby možné nastavit přesné chování v saturaci (tzv. odskakování od saturace vlivem šumu) a při přepínání více regulátorů (velikost skoku při přepnutí, pokud není nulová regulační odchylka).

Přídavné výstupy **dmv**, **de** a **SAT** poskytují po řadě rychlostní výstup regulátoru (diference  $mv$ ), regulační odchylku a příznak saturace výstupu regulátoru  $mv$ .

Jestliže je blok **PIDU** propojen s blokem **SCUV** (za účelem realizace krokového regulátoru bez polohové zpětné vazby), potom parametr **icotype** musí být nastaven na hodnotu **4** a význam výstupů  $mv$ , **dmv** a **SAT** je v tomto případě pozměněn: výstup  $mv$

je roven součtu P a D složky regulátoru, zatímco výstup  $dmv$  poskytuje diferenci jeho I složky a výstup  $SAT$  nese informaci pro blok **SCUV**, zda je regulační odchylka  $de$  v automatickém režimu menší než pásmo necitlivosti  $dz$ . Pro případ propojení bloků PIDU a **SCUV** se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku  $c$  rovný nule.

V manuálním režimu ( $MAN = on$ ) je vstup  $hv$  kopírován na výstup  $mv$ , pokud nenarazí na horní či dolní omezení výstupu regulátoru. Celková regulační funkce bloku PIDU je zřejmá z následujícího obrázku.



## Vstupy

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro výsledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on ... manuální režim	Bool
IH	Zastavení integrace off ... integrování povoleno on ... integrování pozastaveno	Bool

## Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
de	Regulační odchylka	Double (F64)



SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on .... výstup regulátoru je saturován	

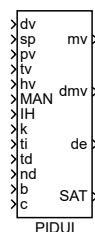
## Parametry

irtype	Typ regulátoru	⊙6	Long (I32)
	1 ..... D      4 ..... P      7 ..... PID		
	2 ..... I      5 ..... PD		
	3 ..... ID      6 ..... PI		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv → vyšší pv		
	on .... vyšší mv → nižší pv		
k	Zesílení regulátoru $K$ . Hodnota 0 (dle definice) vypne regulátor, záporné hodnoty nejsou dovoleny (k tomu slouží parametr RACT).		Double (F64)
		↓0.0 ⊙1.0	
ti	Integrační časová konstanta $T_i$ . Hodnota 0 znamená vypnutí integrační složky regulátoru (stejný efekt jako vypnutí parametrem irtype).		Double (F64)
		↓0.0 ⊙4.0	
td	Derivační časová konstanta $T_d$ . Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem irtype).		Double (F64)
		↓0.0 ⊙1.0	
nd	Parametr $N$ filtru derivační složky. Hodnota 0 znamená vypnutí derivační složky regulátoru (stejný efekt jako vypnutí parametrem irtype).		Double (F64)
		↓0.0 ⊙10.0	
b	Váhový faktor pro proporcionální složku	↓0.0 ↑2.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0 ↑2.0	Double (F64)
tt	Časová konstanta výsledování; hodnota 0 znamená implicitní hodnotu, což je $T_i/2$ pro regulátor s integrační složkou a vypnutí výsledování pro regulátor bez integrační složky. Pokud pro P nebo PD regulátor potřebujeme výsledování (tzv. regulace kolem rovnovážného bodu), zapneme výsledování nastavením kladné hodnoty (větší než perioda vzorkování). Vypnout výsledování pro regulátor s integrační složkou není možné (kvůli windup efektu).	↓0.0 ⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
dz	Pásmo necitlivosti		Double (F64)
icotype	Typ výstupu regulátoru	⊙1	Long (I32)
	1 ..... analogový výstup		
	2 ..... šířkově modulovaný výstup (PWM)		
	3 ..... krokový regulátor s polohovou zpětnou vazbou (SCU)		
	4 ..... krokový regulátor bez polohové zpětné vazby (SCUV)		

## PIDUI – PID regulátor s parametry na vstupech

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Regulační funkce bloku PIDUI je přesně shodná s blokem PIDU. Jediný rozdíl spočívá v tom, že základní parametry PID algoritmu jsou vyvedeny na vstupy. V důsledku toho je lze pohodlně měnit v závislosti na výstupech jiných bloků. Tímto způsobem lze realizovat speciální adaptivní PID regulátory.

### Vstupy

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Velichina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim	Bool
IH	Zastavení integrace off ... integrování povoleno on .... integrování pozastaveno	Bool
k	Zesílení regulátoru $K$	Double (F64)
ti	Integrační časová konstanta $T_i$	Double (F64)
td	Derivační časová konstanta $T_d$	Double (F64)
nd	Parametr $N$ filtru derivační složky	Double (F64)
b	Váhový faktor pro proporcionální složku	Double (F64)
c	Váhový faktor pro derivační složku	Double (F64)

### Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
de	Regulační odchylka	Double (F64)

**SAT** Saturace Bool  
 off ... lineární zákon řízení  
 on .... výstup regulátoru je saturován

## Parametry

**irtype** Typ regulátoru ⊙6 Long (I32)  
 1 ..... D      4 ..... P      7 ..... PID  
 2 ..... I      5 ..... PD  
 3 ..... ID      6 ..... PI

**RACT** Převrácené působení výstupu regulátoru Bool  
 off ... vyšší mv → vyšší pv  
 on .... vyšší mv → nižší pv

**tt** Časová konstanta vysledování ⊙1.0 Double (F64)

**hilim** Horní mez akčního zásahu regulátoru ⊙1.0 Double (F64)

**lolim** Dolní mez akčního zásahu regulátoru ⊙-1.0 Double (F64)

**dz** Pásmo necitlivosti Double (F64)

**icotype** Typ výstupu regulátoru ⊙1 Long (I32)  
 1 ..... analogový výstup  
 2 ..... šířkově modulovaný výstup ([PWM](#))  
 3 ..... krokový regulátor s polohovou zpětnou vazbou ([SCU](#))  
 4 ..... krokový regulátor bez polohové zpětné vazby ([SCUV](#))

## POUT – Pulzní výstup

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok POUT tvaruje vstupní pulzy  $U$  takovým způsobem, že délka výstupního pulzu  $Y$  je alespoň  $dtime$  sekund a prodleva mezi dvěma sousedními výstupními pulzy je minimálně  $btime$  sekund. Vstupní pulz, který přijde po sestupné hraně výstupního signálu dříve, než uplyne čas  $btime$ , nezpůsobí žádnou odezvu na výstupu  $Y$ .

### Vstup

$U$	Logický vstupní signál	Bool
-----	------------------------	------

### Výstup

$Y$	Logický výstupní signál	Bool
-----	-------------------------	------

### Parametry

$dtime$	Minimální trvání výstupního pulzu [s]	⊙1.0	Double (F64)
$btime$	Minimální prodleva mezi sousedními výstupními pulzy [s]	⊙1.0	Double (F64)

## PRGM – Programátor

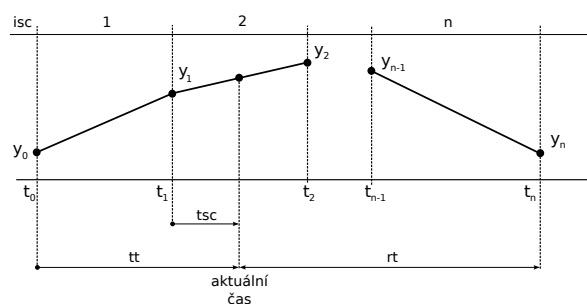
Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok PRGM je určen pro generování časových funkcí (programů) složených z  $n$  lineárních částí definovaných  $(n + 1)$  rozměrnými vektory  $\mathbf{tm} = [t_0, \dots, t_n]$  času a požadovaných hodnot  $\mathbf{y} = [y_0, \dots, y_n]$  (generovaná křivka je spojitá po částech lineární, viz. obrázek). Nejčastěji je používán pro generování požadované hodnoty regulátoru. Generování programu je spuštěno vstupem **RUN = on**; přechod zpět na **RUN = off** vrací stav programátoru do základního stavu. Vstup **DEF** nastaví **sp** na hodnotu **spv** a po vymizení hodnoty **DEF = on** se pokračuje přejetím po rampě na nejbližší následující uzel, čas přitom není narušen. Vstup **HLD = on** zmrazí výstupní hodnotu **sp** a všechny výstupní časy (**tsc**, **tt**, **rt**), po vymizení hodnoty **HLD = on** se pokračuje z okamžiku zmrznutí dále podle programu. Je-li při přechodu **HLD on**→**off** nastaven vstup **CON = on**, nepokračuje se od okamžiku zmrazení, ale najede se do uzlového bodu s indexem **ind** po rampě za čas **trt**. Index uzlového bodu **ind** musí být rovný nebo větší než aktuálně prováděný sektor (v okamžiku **HLD on**→**off**). Je-li **RPT = on**, potom se program generuje opakovaně.



### Vstupy

RUN	Povolení generování časové funkce programu	Bool
DEF	Inicializace <b>sp</b> na hodnotu <b>spv</b>	Bool
spv	Inicializační hodnota	Double (F64)
HLD	Zmrazení výstupu a výstupních časů	Bool
CON	Pokračování od uzlového bodu <b>ind</b>	Bool

<code>ind</code>	Index uzlového bodu pro pokračování	Long (I32)
<code>trt</code>	Čas pro dosažení požadovaného uzlu <code>ind</code>	Double (F64)
<code>RPT</code>	Příznak opakování generování časové funkce	Bool

### Výstupy

<code>sp</code>	Požadovaná hodnota (hodnota časové funkce v daném čase)	Double (F64)
<code>isc</code>	Aktuální sektor funkce	Long (I32)
<code>tsc</code>	Čas od začátku sektoru	Double (F64)
<code>tt</code>	Čas od startu generování časové funkce	Double (F64)
<code>rt</code>	Čas do konce programu	Double (F64)
<code>CNF</code>	Příznak sledování nakonfigurované křivky	Bool
<code>E</code>	Chyba, časy uzlů nejsou seřazeny vzestupně	Bool

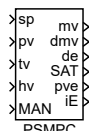
### Parametry

<code>nmax</code>	Maximální (alokovaná) délka pole	$\downarrow 4 \uparrow 10000000 \odot 10$	Long (I32)
<code>tmunits</code>	Jednotky pro zadávání časů	$\odot 1$	Long (I32)
	1 . . . . . sekundy		
	2 . . . . . minuty		
	3 . . . . . hodiny		
<code>tm</code>	$(n + 1)$ -rozměrný vektor vzestupně uspořádaných časů	$\odot [0 \ 1 \ 2]$	Double (F64)
<code>y</code>	$(n + 1)$ -rozměrný vektor hodnot časové funkce	$\odot [0 \ 1 \ 0]$	Double (F64)

## PSMPC – Prediktivní „pulse-step“ regulátor

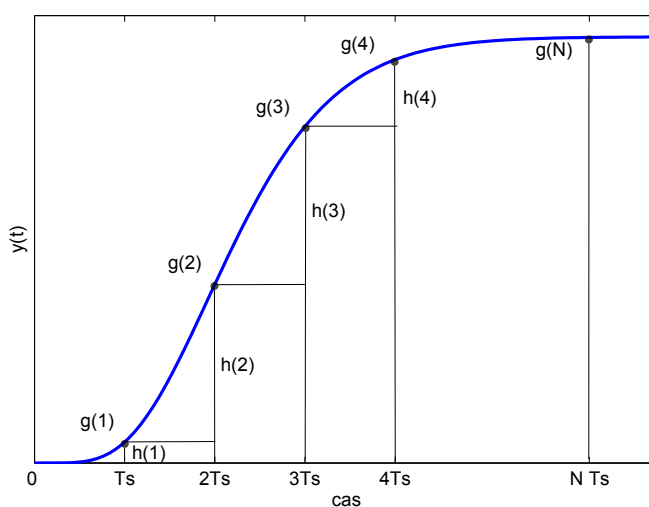
Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok PSMPC (Pulse Step Model Predictive Control) je určen pro realizaci vysoce kvalitních regulátorů pro obtížně regulovatelné lineární časově invariantní soustavy s omezením akční veličiny (např. soustavy s dopravním zpožděním nebo s neminimální fází). Zvláště výhodný je pro případy, kdy je požadován velmi rychlý přechod z jedné hodnoty regulované veličiny na druhou bez překmitu. Regulátor PSMPC však může být obecně použit všude tam, kde je běžně nasazován standardní PID regulátor a kde žádáme vysokou kvalitu regulace.



PSMPC je prediktivní regulátor s explicitně zadaným intervalovým omezením akční veličiny. Pro účely predikce je použit model ve tvaru diskrétní přechodové charakteristiky  $g(j)$ ,  $j = 1, \dots, N$ . Na obrázku výše je naznačen způsob, jakým lze tuto posloupnost získat ze spojité přechodové charakteristiky. Poznamenejme, že  $N$  musí být zvoleno dostatečně velké, aby přechodová charakteristika byla popsána až do ustáleného stavu ( $NT_S > t_{95}$ , kde  $T_S$  je perioda vzorkování regulátoru a  $t_{95}$  je doba ustálení na 95 % konečné hodnoty). Pro systémy s monotónní přechodovou charakteristikou je alternativně možné použít momentový množinový model [3] a popsat systém pouze třemi charakteristickými čísly

$\kappa, \mu$  a  $\sigma^2$ , které je možno určit z jednoduchého pulzního experimentu. Řízený systém pak aproximujeme buď přenosem prvního řádu s dopravním zpožděním

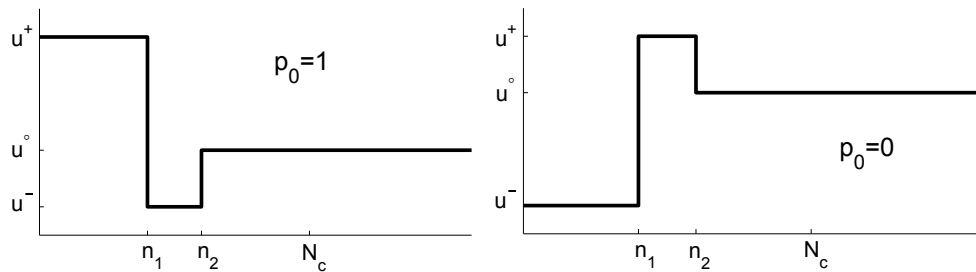
$$F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = \tau + D, \quad \sigma^2 = \tau^2 \quad (7.1)$$

nebo přenosem druhého řádu s dopravním zpožděním

$$F_{SOPDT}(s) = \frac{K}{(\tau s + 1)^2} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = 2\tau + D, \quad \sigma^2 = 2\tau^2 \quad (7.2)$$

se stejnými charakteristickými čísly. Typ aproximace se zadává parametrem `imtype`.

Pro zjednodušení on-line optimalizace v otevřené smyčce je množina přípustných posloupností řízení omezena pouze na posloupnosti ve tvaru "pulz-skok" zobrazené na obrázku níže.



Poznamenejme, že každá taková posloupnost je jednoznačně určena jen třemi čísly  $n_1, n_2 \in \{0, \dots, N_C\}$  a  $u^\infty \in \langle u^-, u^+ \rangle$ , kde  $N_C \in \{0, 1, \dots\}$  je horizont řízení a  $u^-, u^+$  označují po řadě zadanou dolní a horní mez akční veličiny regulátoru. On-line optimalizace (vzhledem k  $n_1, n_2$  a  $u^\infty$ ) spočívá v minimalizaci kritéria

$$I = \sum_{i=N_1}^{N_2} \hat{e}(k+i|k)^2 + \lambda \sum_{i=0}^{N_C} \Delta \hat{u}(k+i|k)^2 \rightarrow \min, \quad (7.3)$$

kde  $\hat{e}(k+i|k)$  je v kroku  $k$  predikovaná regulační odchylka na intervalu predikce  $i \in \{N_1, N_2\}$ ,  $\Delta \hat{u}(k+i|k)$  jsou difference řídicího signálu na intervalu  $i \in \{0, N_C\}$  a  $\lambda$  je koeficient penalizace změn akční veličiny. Pro nalezení optima úlohy (7.3) je použita kombinace metody nejmenších čtverců a hrubé síly. Hodnota  $u^\infty$  je určena pro všechny přípustné kombinace  $p_0, n_1$  a  $n_2$  a následně je z nich vybrána optimální řídicí sekvence pro řízení v otevřené smyčce. Ve skutečnosti je však vždy aplikován pouze první krok této řídicí sekvence a v další vzorkovací periodě je celý optimalizační proces zopakován. Tím se řídicí strategie mění na zpětnovazební řízení.

Parametry prediktivního regulátoru, kromě modelu řízené soustavy a omezení jejího vstupu, jsou horizont řízení  $N_C$ , horizont predikce  $N_1, N_2$  a koeficient  $\lambda$ . Pouze poslední uvedený parametr je určen pro ruční doladění kvality regulace při rutinním uvádění do provozu. V případě použití modelu soustavy ve tvaru přenosu (7.1) nebo (7.2) jsou parametry  $N_1, N_2$  zvoleny automaticky na základě charakteristických čísel  $\mu, \sigma^2$ . Regulátor potom může být efektivně laděn „ručně“ pouze seřizováním charakteristických čísel procesu  $\kappa, \mu, \sigma^2$ .



## Varování

Při použití bloku PSMPC pro simulaci v systému Matlab/Simulink je třeba zajistit, aby pole **sr** byl dostatečně velké, tak aby jím definovaný buffer pojmul interně vygenerovanou přechodovou charakteristiku určenou z FOPDT nebo SOPDT modelu. V opačném případě blok hlásí chybu.

## Vstupy

<b>sp</b>	Požadovaná hodnota (setpoint)	Double (F64)
<b>pv</b>	Řízená veličina	Double (F64)
<b>tv</b>	Velichina pro vysledování (použitý řídicí signál)	Double (F64)
<b>hv</b>	Hodnota výstupu v manuálním režimu	Double (F64)
<b>MAN</b>	Manuální nebo automatický režim	Bool
	<b>off</b> ... automatický režim	
	<b>on</b> .... manuální režim	

## Výstupy

<b>mv</b>	Akční zásah regulátoru (manipulated variable)	Double (F64)
<b>dmv</b>	Rychlostní výstup regulátoru (diference)	Double (F64)
<b>de</b>	Regulační odchylka	Double (F64)
<b>SAT</b>	Saturace	Bool
	<b>off</b> ... lineární zákon řízení	
	<b>on</b> .... výstup regulátoru je saturován	
<b>pve</b>	Predikovaná hodnota regulované veličiny na základě zadaného modelu	Double (F64)
<b>iE</b>	Kód chyby	Long (I32)
	0 ..... bez chyby	
	1 ..... nesprávný FOPDT model	
	2 ..... nesprávný SOPDT model	
	3 ..... chyba v zadání přechodové charakteristiky	

## Parametry

<b>nc</b>	Délka horizontu řízení ( $N_C$ )	⊙5	Long (I32)
<b>np1</b>	Začátek koincidenčního intervalu ( $N_1$ )	⊙1	Long (I32)
<b>np2</b>	Konec koincidenčního intervalu ( $N_2$ )	⊙10	Long (I32)
<b>lambda</b>	Koeficient penalizace změn řízení ( $\lambda$ )	⊙0.05	Double (F64)
<b>umax</b>	Horní mez akčního zásahu regulátoru ( $u^+$ )	⊙1.0	Double (F64)
<b>umin</b>	Dolní mez akčního zásahu regulátoru ( $u^-$ )	⊙-1.0	Double (F64)
<b>imtype</b>	Typ modelu řízené soustavy	⊙3	Long (I32)
	1 ..... model prvního řádu		
	2 ..... model druhého řádu		
	3 ..... přechodová charakteristika		
<b>kappa</b>	Statické zesílení ( $\kappa$ )	⊙1.0	Double (F64)
<b>mu</b>	Míra zpoždění soustavy ( $\mu$ )	⊙20.0	Double (F64)

<code>sigma</code>	Míra délky odezvy soustavy ( $\sqrt{\sigma^2}$ )	$\odot 10.0$	Double (F64)
<code>nmax</code>	Reservovaná velikost pole $sr$	$\downarrow 10 \uparrow 10000$	$\odot 32$ Long (I32)
<code>sr</code>	Diskrétní přechodová charakteristika ( $[g(1), \dots, g(N)]$ )		Double (F64)
	$\odot [0 \ 0.2642 \ 0.5940 \ 0.8009 \ 0.9084 \ 0.9596 \ 0.9826 \ 0.9927 \ 0.9970 \ 0.9988 \ 0.9995]$		

## PWM – Blok šířkové modulace

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok PWM provádí pulzně šířkovou modulaci vstupního signálu z intervalu od  $-1$  do  $+1$ . Užitím tohoto bloku je možné realizovat proporcionální akční veličinu  $i$  u akčních členů s jedním (např. topení zapnuto/vypnuto) nebo dvěma (např. topení zapnuto/vypnuto a chlazení zap./vyp.) binárními vstupy. Šířka  $L$  výstupního pulzu je určena vztahem:

$$L = \text{pertm} * |u|,$$

kde  $\text{pertm}$  je perioda modulace. Je-li  $u > 0$  ( $u < 0$ ), pulz je generován na výstupu UP (DN). Z praktických důvodů je však délka generovaného pulzu dále upravována podle zadaných parametrů bloku. Faktor asymetrie  $\text{asyfac}$  definuje poměr mezi délkou negativního pulzu DN a délkou pozitivního pulzu UP. Modifikované délky se počítají podle vztahů:

$$\begin{aligned} \text{jestliže } u > 0 \text{ potom } L(\text{UP}) &:= \begin{cases} L & \text{pro } \text{asyfac} \leq 1.0 \\ L/\text{asyfac} & \text{pro } \text{asyfac} > 1.0 \end{cases} \\ \text{jestliže } u < 0 \text{ potom } L(\text{DN}) &:= \begin{cases} L * \text{asyfac} & \text{pro } \text{asyfac} \leq 1.0 \\ L & \text{pro } \text{asyfac} > 1.0 \end{cases} \end{aligned}$$

kteří pro libovolnou hodnotu  $\text{asyfac} > 0$  zajišťují, že maximální délka generovaných pulzů je rovna  $\text{pertm}$ . Dále, jestliže vypočtená délka pulzu je menší než  $\text{dtime}$ , potom je výsledná délka nastavena na nulu. Jestliže se vypočtená délka pulzu liší od  $\text{pertm}$  méně než  $\text{btime}$ , potom je výsledná délka nastavena na  $\text{pertm}$ . Jestliže kladný pulz UP je následovaný záporným pulzem DN nebo obráceně, potom pozdější pulz je v případě potřeby posunut tak, že vzdálenost mezi těmito dvěma pulzy je alespoň  $\text{offtime}$ . Jestliže  $\text{SYNCH} = \text{on}$ , potom změna vstupu  $u$  způsobí okamžitý přepočtení délky výstupního pulzu za předpokladu, že není splněna synchronizační podmínka mezi začátkem periody modulace a okamžikem změny vstupu  $u$ .

### Vstup

$u$  Analogový vstupní signál Double (F64)

### Výstupy

UP Signál UP (nahoru, více) Bool  
DN Signál DN (dolů, méně) Bool

## Parametry

<code>pertm</code>	Perioda šířkové modulace [s]	⊙10.0	Double (F64)
<code>dtime</code>	Minimální trvání výstupního pulzu [s]	⊙0.1	Double (F64)
<code>btime</code>	Minimální prodleva mezi pulzy [s]	⊙0.1	Double (F64)
<code>offtime</code>	Minimální prodleva mezi pulzy opačné polarity [s]	⊙1.0	Double (F64)
<code>asyfac</code>	Faktor asymetrie	⊙1.0	Double (F64)
<code>SYNCH</code>	Synchronizační příznak pro začátek periody		Bool
	<code>off</code> ... synchronizace vypnuta		
	<code>on</code> ... synchronizace zapnuta		

## RLY – Relé s hysterezí

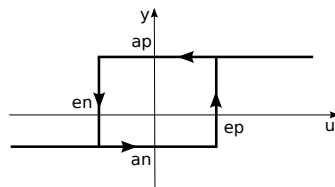
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok RLY transformuje vstupní analogový signál  $u$  na výstupní analogový signál  $y$  podle níže uvedeného obrázku.



Vstup

$u$  Analogový vstupní signál Double (F64)

Výstup

$y$  Analogový výstupní signál Double (F64)

Parametry

$ep$	Hodnota $u > ep$ způsobí $y = ap$ („Zapnuto“)	$\odot 1.0$	Double (F64)
$en$	Hodnota $u < en$ způsobí $y = an$ („Vypnuto“)	$\odot -1.0$	Double (F64)
$ap$	Výstup ve stavu „Zapnuto“	$\odot 1.0$	Double (F64)
$an$	Výstup ve stavu „Vypnuto“	$\odot -1.0$	Double (F64)
$y_0$	Počáteční hodnota výstupu $y$ po spuštění		Double (F64)

## SAT – Saturace výstupu s proměnnými mezemi

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok SAT kopíruje vstup  $u$  na výstupu  $y$ , pokud pro vstupní veličinu platí  $lolim \leq u$  a  $u \leq hilim$ , kde  $lolim$  a  $hilim$  jsou stavové proměnné bloku. Je-li  $u < lolim$  (resp.  $u > hilim$ ), potom  $y = lolim$  ( $y = hilim$ ). Horní a dolní limit jsou buď pevné hodnoty dané po řadě parametry bloku  $hilim0$  a  $lolim0$  (případ  $HLD = on$ ) nebo jsou řízeny vstupy  $hi$  a  $lo$  ( $HLD = off$ ). Maximální rychlost změny aktivních mezí  $hilim$  a  $lolim$  je dána časovými konstantami  $tp$  a  $tn$ . Parametr  $tp$  určuje maximální kladnou strmost a  $tn$  maximální zápornou strmost změn  $hilim$  a  $lolim$ . Omezení strmosti změn mezí je aktivní i v případě, že hodnoty mezí měníme ručně ( $HLD = on$ ) pomocí parametrů  $hilim0$  a  $lolim0$ . Pro možnost okamžitých změn saturačních mezí je potřeba nastavit  $tp = 0$  a  $tn = 0$ . Výstupy  $HL$  a  $LL$  signalizují po řadě horní a dolní saturaci.

Pokud je to potřeba, parametry  $hilim0$  a  $lolim0$  jsou použity jako počáteční hodnoty pro saturační meze řízené vstupními signály.

### Vstupy

$u$	Analogový vstupní signál	Double (F64)
$hi$	Horní saturační mez pro případ $HLD = off$	Double (F64)
$lo$	Dolní saturační mez pro případ $HLD = off$	Double (F64)

### Výstupy

$y$	Analogový výstupní signál	Double (F64)
$HL$	Příznak saturace na horní mezi	Bool
$LL$	Příznak saturace na dolní mezi	Bool

### Parametry

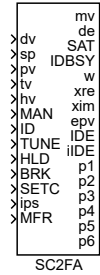
$tp$	Časová konstanta rychlosti změn aktivních hodnot mezí v kladném směru	$\odot 1.0$	Double (F64)
$tn$	Časová konstanta rychlosti změn aktivních hodnot mezí v záporném směru	$\odot 1.0$	Double (F64)
$hilim0$	Horní omezení výstupu (platné pro $HLD = on$ )	$\odot 1.0$	Double (F64)
$lolim0$	Dolní omezení výstupu (platné pro $HLD = on$ )	$\odot -1.0$	Double (F64)

HLD      Pevné saturační meze       $\odot$ on    Bool  
          off ... proměnné meze    on .... pevné meze

## SC2FA – Stavový regulátor systému 2. řádu s autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



### Popis funkce

Funkční blok SC2FA realizuje stavový regulátor pro systém druhého řádu (7.4) s frekvenčním autotunerem. Je vhodný především pro aktivní řízení (zatlumení) kmitavých systémů s velmi slabým tlumením ( $\xi < 0,1$ ). Může však být použit též jako samonastavující se regulátor pro libovolný systém, který lze s dostatečnou přesností popsat přenosem ve tvaru

$$F(s) = \frac{b_1s + b_0}{s^2 + 2\xi\Omega s + \Omega^2}, \quad (7.4)$$

kde  $\Omega > 0$  je přirozená (netlumená) frekvence,  $\xi$ ,  $0 < \xi < 1$ , je koeficient tlumení a  $b_1$ ,  $b_0$  jsou libovolná reálná čísla. Blok pracuje ve dvou režimech, v režimu Identifikace a návrhu a v režimu Regulace.

Režim „Identifikace a návrhu“ ze zapíná nastavením binárního vstupu ID = on. Vlastní proces identifikace a návrhu se spouští náběžnou hranou vstupu RUN. Na výstupu bloku mv se poté objeví budící harmonický signál se stejnoměrnou složkou ubias, amplitudou uamp a frekvencí  $\omega$  postupně probíhající interval  $\langle \mathbf{wb}, \mathbf{wf} \rangle$ . Aktuální frekvence  $\omega$  je přitom kopírována na výstup w. Rychlost změny (rozmitání) frekvence je dána parametrem cp, který udává relativní zmenšení počáteční periody  $T_b = \frac{2\pi}{\mathbf{wb}}$  budící sinusovky za čas  $T_b$ , tedy

$$c_p = \frac{\mathbf{wb}}{\omega(T_b)} = \frac{\mathbf{wb}}{\mathbf{wb}e^{\gamma T_b}} = e^{-\gamma T_b}. \quad (7.5)$$

Hodnota parametru cp se obvykle pohybuje v intervalu  $cp \in \langle 0,95; 1 \rangle$ . Čím menší je koeficient tlumení řízeného systému, tím více se musí cp blížit k jedné.

Identifikace systému se spouští náběžnou hranou vstupu RUN zároveň s generátorem budícího signálu se startovací frekvencí  $\omega = \mathbf{wb}$ . Po uplynutí stime se startuje výpočet odhadu aktuálního bodu frekvenční charakteristiky. Jeho reálná a imaginární část se průběžně kopíruje po řadě na výstupy xre a xim. Je-li parametr bloku MANF nastaven na 0,



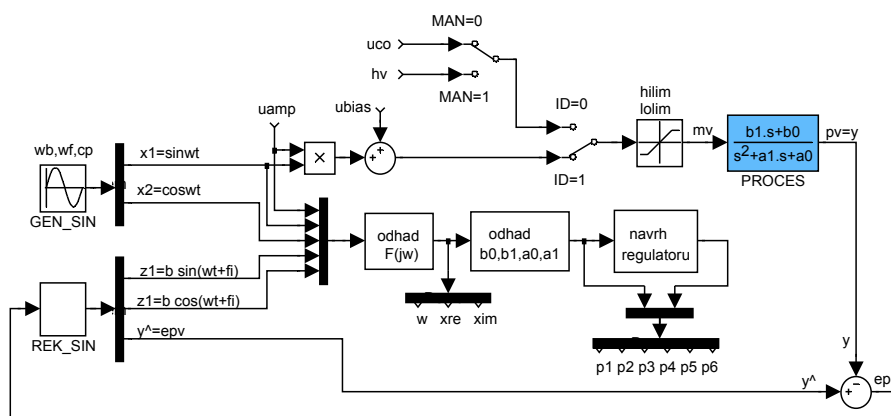
potom se v procesu identifikace dvakrát zastaví rozmítání frekvence na dobu `stime` a to v okamžicích, kdy jsou poprvé dosaženy body s fázovým zpožděním `ph1` a `ph2`. Přednastavené hodnoty parametrů `ph1` a `ph2` jsou po řadě  $-60^\circ$  a  $-120^\circ$  a mohou být změněny na libovolné hodnoty v intervalu  $(-360^\circ, 0^\circ)$ , přičemž `ph1` > `ph2`. Po uplynutí `stime` sekund při zastavení ve fázi `ph1`, resp. `ph2` se spočítá průměr posledních `iavg` naměřených bodů (průměrováním tedy získáme odhad příslušného bodu frekvenční charakteristiky) pro následný výpočet parametrického modelu ve tvaru (7.4). Je-li `MANF = on`, potom je možné provést „navzorkování“ dvou bodů frekvenční charakteristiky ručně pomocí vstupu `HLD`. Vstup `HLD = on` zastaví rozmítání frekvence a opětovné nastavení `HLD = off` vede k jeho pokračování. Ostatní funkce jsou identické.

V případě potřeby je možné proces identifikace přerušit vstupem `BRK = on`. Jsou-li již v tomto okamžiku oba dva body pro parametrickou identifikaci určeny, pokračuje se v návrhu regulátoru normálním způsobem. V opačném případě je proces ukončen bez návrhu regulátoru a výstup `IDE = on` signalizuje chybu.

Během vlastní „identifikace a návrhu“ je výstup `IDBSY` nastaven na 1. Po skončení je shozen na 0. Při bezchybném návrhu regulátoru je výstup `IDE = off` a výstup `iIDE` signalizuje jednotlivé fáze identifikačního experimentu. Přibližování k prvnímu bodu je `iIDE = -1`, zastavení v prvním bodě `iIDE = 1`, přibližování k druhému bodu je `iIDE = -2`, zastavení v druhém bodě `iIDE = 2` a poslední fáze po zastavení v druhém bodě je `iIDE = -3`. Jestliže identifikace skončí s chybou, pak je `IDE = on` a číslo na výstupu `iIDE` specifikuje příslušnou chybu.

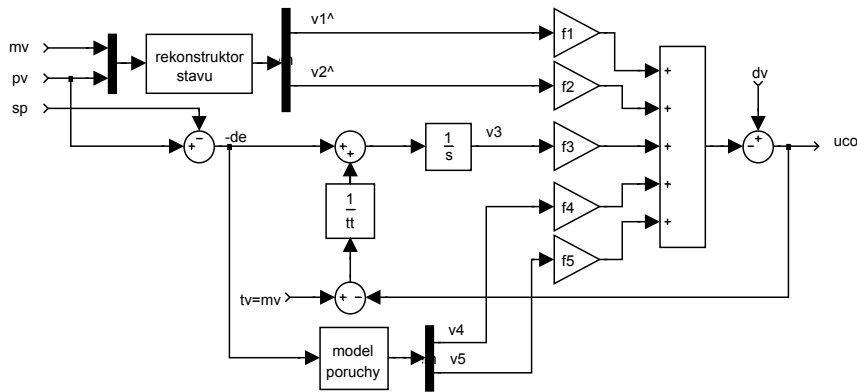
Vypočtené parametry stavového regulátoru jsou instalovány okamžitě do algoritmu řízení, jestliže vstup `SETC` je trvale nastaven na `on`. V opačném případě se provede nastavení parametrů až po ukončení návrhu na náběžnou hranu vstupu `SETC`. Výsledky parametrické identifikace a návrhu stavového regulátoru je možné získat na výstupech `p1`, `p2`, ..., `p6` vhodným nastavením vstupu `ips`. Náběžná hrana na vstupu `MFR` po skončení identifikace (`IDBSY = off`) odstartuje generování frekvenční charakteristiky získaného parametrického modelu na výstupech `w`, `xre`, `xim`. Takto je možno porovnat její průběh s „přímo odměřenou“ frekvenční charakteristikou systému.

V režimu „regulace“ (binární vstup `ID = off`) může regulátor pracovat v manuálním módu (`MAN = on`) nebo v automatickém módu. Jestliže je blok regulátoru spuštěn (při studeném startu) s hodnotou vstupu `ID = off`, potom se předpokládá, že zadané parametry bloku `mb0`, `mb1`, `ma0` a `ma1` odpovídají dříve určeným koeficientům  $b_0$ ,  $b_1$ ,  $a_0$  a  $a_1$  přenosu řízeného systému a automaticky proběhne návrh stavového regulátoru. Je-li regulátor navíc v automatickém módu a `SETC = on`, potom zákon řízení od počátku využívá nově navržené parametry. Takto lze vypustit identifikaci při opakovaném spuštění bloku.



Na výše uvedeném obrázku je zjednodušené vnitřní schéma samonastavujícího se stavového regulátoru, část frekvenční identifikace. Na spodním obrázku je stavová zpětná vazba s rekonstruktorem stavu a ošetřením unášení integrační složky. Na obrázku není znázorněna skutečnost, že blok návrhu regulátoru v části frekvenční identifikace automaticky nastaví parametry rekonstruktora stavu a koeficienty  $f_1, f_2, \dots, f_5$  stavové zpětné vazby.

ı



Model řízeného systému je brán jako systém 2. řádu s přenosem ve tvaru (7.4). Jednoduchými úpravami lze dojít k přenosům

$$F(s) = \frac{(b_1 s + b_0)}{s^2 + a_1 s + a_0} \quad (7.6)$$

a

$$F(s) = \frac{K_0 \Omega^2 (\tau s + 1)}{s^2 + 2\xi \Omega s + \Omega^2}. \quad (7.7)$$

Parametry těchto přenosů je možné po skončení identifikace přečíst z výstupů p1, . . . , p6. Význam těchto výstupů se mění při změně vstupu `ips`, avšak pouze pokud neběží identifikace (tedy `IDBSY = off`).

## Vstupy

<code>dv</code>	Proměnná dopředné vazby	Double (F64)
<code>sp</code>	Požadovaná hodnota (setpoint)	Double (F64)
<code>pv</code>	Řízená veličina	Double (F64)
<code>tv</code>	Velichina pro vysledování	Double (F64)

hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on ... manuální režim	Bool
ID	Režim identifikace nebo regulace off ... režim regulátoru návrhu on ... režim identifikace a	Bool
TUNE	Zahájení ladicího experimentu, start generátor harmonického budicího signálu (off→on)	Bool
HLD	Zastavení rozmítání frekvence	Bool
BRK	Signál pro přerušení identifikačního experimentu	Bool
SETC	Přijmutí a nastavení parametrů regulátoru off ... parametry jsou pouze vypočítány on ... parametry jsou přijaty ihned po vypočtení off→on jednorázové přijetí vypočtených parametrů	Bool
ips	Význam výstupních signálů 0 ..... Dva body frekvenční charakteristiky v komplexní rovině p1 ... frekvence 1. změřeného bodu v rad/s p2 ... reálná část 1. bodu p3 ... imaginární část 1. bodu p4 ... frekvence 2. změřeného bodu v rad/s p5 ... reálná část 2. bodu p6 ... imaginární část 2. bodu 1 ..... Model druhého řádu ve tvaru (7.6) p1 ... parametr $b_1$ p2 ... parametr $b_0$ p3 ... parametr $a_1$ p4 ... parametr $a_0$ 2 ..... Model druhého řádu ve tvaru (7.7) p1 ... parametr $K_0$ p2 ... parametr $\tau$ p3 ... parametr $\Omega$ v rad/s p4 ... parametr $\xi$ p5 ... parametr $\Omega$ v Hz p6 ... rezonanční frekvence modelu v Hz 3 ..... Parametry stavové zpětné vazby p1 ... parametr $f_1$ p2 ... parametr $f_2$ p3 ... parametr $f_3$ p4 ... parametr $f_4$ p5 ... parametr $f_5$	Long (I32)
MFR	Generování frekvenční charakteristiky modelu na výstupy w, xre a xim (off→on spouští generování)	Bool

### Výstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)

SAT	Saturace off ... lineární zákon řízení on ... výstup regulátoru je saturován	Bool
IDBSY	Příznak probíhající identifikace off ... identifikace neprobíhá on ... běží identifikační experiment	Bool
w	Odhad bodu frekvenční charakteristiky - frekvence v rad/s	Double (F64)
xre	Odhad bodu frekvenční charakteristiky - reálná část	Double (F64)
xim	Odhad bodu frekvenční charakteristiky - imaginární část	Double (F64)
epv	Rekonstruovaný signál pv (pro účely ručního ladění rekonstruktoru)	Double (F64)
IDE	Příznak chyby identifikace off ... identifikace proběhla v pořádku on ... identifikace skončila s chybou	Bool
iIDE	Kód chyby 101 ... vzorkovací frekvence je příliš nízká 102 ... chyba identifikace jednoho nebo dvou bodů frekvenční charakteristiky 103 ... saturace výstupu během identifikace 104 ... je zadán nebo spočten nesprávný model procesu	Long (I32)
p1..p6	Výsledky identifikace a návrhu regulátoru	Double (F64)

## Parametry

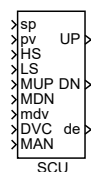
ubias	Stejnoseměrná složka budicího harmonického signálu	Double (F64)
uamp	Amplituda budicího harmonického signálu	⊙1.0 Double (F64)
wb	Počátek frekvenčního intervalu [rad/s]	⊙1.0 Double (F64)
wf	Konec frekvenčního intervalu [rad/s]	⊙10.0 Double (F64)
isweep	Způsob rozmítání frekvence 1 ..... logaritmické 2 ..... lineární (zatím není implementováno)	⊙1 Long (I32)
cp	Rychlost rozmítání	↓0.5 ↑1.0 ⊙0.995 Double (F64)
iavg	Počet vzorků pro průměrování	⊙10 Long (I32)
alpha	Relativní poloha pólů rekonstruktoru (ve fázi identifikace)	Double (F64) ⊙2.0
xi	Koeficient tlumení rekonstruktoru (ve fázi identifikace)	Double (F64) ⊙0.707
MANF	Ruční výběr bodů frekvenční charakteristiky off ... zakázáno on ... povoleno	Bool
ph1	Fázové zpoždění prvního bodu ve stupních	⊙-60.0 Double (F64)
ph2	Fázové zpoždění druhého bodu ve stupních	⊙-120.0 Double (F64)
stime	Doba ustálení [s]	⊙10.0 Double (F64)
ralpha	Relativní poloha pólů rekonstruktoru	⊙4.0 Double (F64)
rx1	Koeficient tlumení rekonstruktoru	⊙0.707 Double (F64)
ac11	Relativní poloha 1. dvojice pólů uzavřené smyčky	⊙1.0 Double (F64)

xicl1	Tlumení 1. dvojice pólů uzavřené smyčky	⊙0.707	Double (F64)
INTGF	Příznak rozšíření o integrátor	⊙on	Bool
	off ... stavový model neobsahuje integrátor		
	on .... integrátor je zahrnut ve stavovém modelu		
apcl	Relativní poloha reálného pólu	⊙1.0	Double (F64)
DISF	Příznak rozšíření o model poruchy		Bool
	off ... stavový model neobsahuje model poruchy		
	on .... model poruchy je zahrnut ve stavovém modelu		
dom	Přirozená frekvence modelu poruchy	⊙1.0	Double (F64)
dxi	Koeficient tlumení modelu poruchy		Double (F64)
ac12	Relativní poloha 2. dvojice	⊙2.0	Double (F64)
xicl2	Tlumení 2. dvojice pólů uzavřené smyčky	⊙0.707	Double (F64)
tt	Časová konstanta vysledování	⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
mb1p	Koeficient přenosu řízeného systému ( $b_1$ )		Double (F64)
mb0p	Koeficient přenosu řízeného systému ( $b_2$ )	⊙1.0	Double (F64)
ma1p	Koeficient přenosu řízeného systému ( $a_1$ )	⊙0.2	Double (F64)
ma0p	Koeficient přenosu řízeného systému ( $a_0$ )	⊙1.0	Double (F64)

## SCU – Krokový regulátor s polohovou zpětnou vazbou

Symbol bloku

Licence: [STANDARD](#)



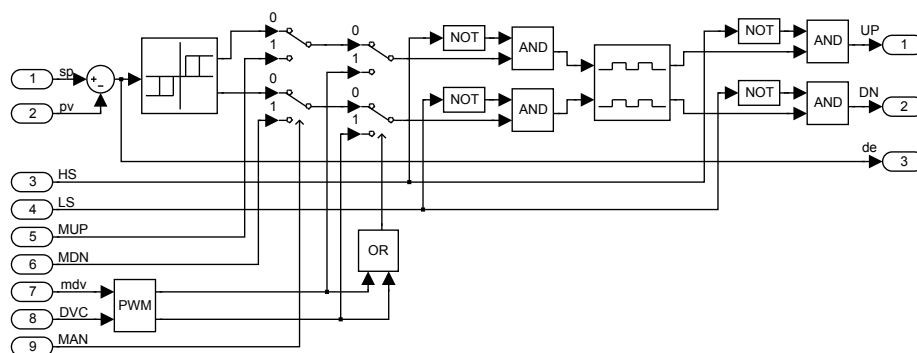
### Popis funkce

Blok SCU je polohový regulátor servoventilu s třístavovým výstupem. Ve spojení s nadřazeným blokem PIDU nebo od něho odvozeným (PIDMA, atd.) je určen k realizaci třístavového krokového regulátoru s polohovou zpětnou vazbou.

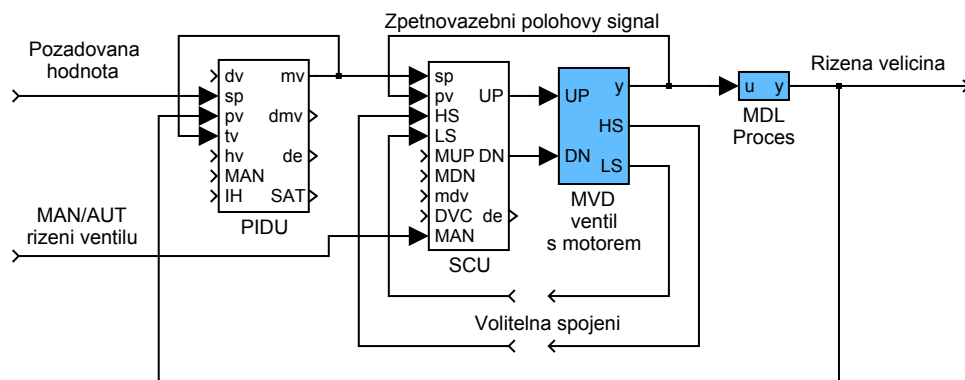
Blok SCU nejprve zpracovává regulační odchylku  $sp - pv$  na třístavový výstup symetrickým třístavovým algoritmem s parametry (práhy)  $thron$  a  $throff$  (viz blok TSE, uvažujte parametry  $ep = thron$ ,  $epoff = throff$ ,  $en = -thron$  a  $enoff = -throff$ ). Parametr RACT určuje, pro kterou polaritu odchylky jsou generovány pulsy UP (více) nebo DN (méně). Výstupy symetrického třístavového algoritmu jsou dále zpracovávány tak, že délka libovolného generovaného pulsu (UP, DN) na výstupu bloku je alespoň  $dtime$  a prodleva mezi dvěma po sobě jdoucími pulsy opačné polarity je alespoň  $btime$ . Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu  $MAN = on$ , pak lze s motorem pohybovat tam a zpět pomocí signálů MUP a MDN, eventuálně lze pomocí vstupu  $mdv$  nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou ( $off \rightarrow on$ ) na vstupu DVC.

Celková funkce bloku SCU je dostatečně zřejmá z následujícího diagramu.



Úplný třístavový krokový regulátor s polohovou zpětnou vazbou je zobrazen na následujícím obrázku.



## Vstupy

sp	Požadovaná hodnota (výstup primárního regulátoru)	Double (F64)
pv	Řízená veličina (poloha servopohonu ventilu)	Double (F64)
HS	Horní koncový spínač (příznak, že poloha ventilu je na horní mezi)	Bool
LS	Dolní koncový spínač (příznak, že poloha ventilu je na spodní mezi)	Bool
MUP	Manuální povel UP (nahoru, přidej)	Bool
MDN	Manuální povel DN (dolů, uber)	Bool
mdv	Ruční diferenční hodnota (požadovaný přírůstek/úbytek polohy, mající vyšší prioritu než přímé signály MUP/MDN)	Double (F64)
DVC	Přijetí ruční diferenční hodnoty (off→on)	Bool
MAN	Manuální nebo automatický režim off ... automatický režim on ... manuální režim	Bool

## Výstupy

UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool
de	Regulační odchylka	Double (F64)

## Parametry

thron	Mez pro zapnutí	↓0.0 ⊙0.02	Double (F64)
throff	Mez pro vypnutí	↓0.0 ⊙0.01	Double (F64)
dtime	Minimální trvání výstupního pulzu [s]	↓0.0 ⊙0.1	Double (F64)
btime	Minimální prodleva mezi pulzy [s]	↓0.0 ⊙0.1	Double (F64)

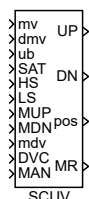


<b>RACT</b>	Převrácené působení výstupu regulátoru <b>off</b> ... vyšší mv → vyšší pv <b>on</b> ... vyšší mv → nižší pv	<b>Bool</b>
<b>trun</b>	Časová konstanta motoru	↓0.0 ⊕10.0 <b>Double (F64)</b>

## SCUV – Krokový regulátor s rychlostním výstupem

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok SCUV nahrazuje polohový regulátor SCU v úplné regulační smyčce s třístavovým krokovým regulátorem, jestliže polohový signál servoventilu není dostupný anebo dostatečně spolehlivý. Nadřazený regulátor PIDU (nebo odvozený) je propojen s blokem SCUV signály mv, dmv a SAT (výstupy bloku PIDU a vstupy bloku SCUV).

Jestliže je nadřazený regulátor typu PI nebo PID ( $CW_{OI} = \text{off}$ ), potom jsou všechny tři vstupy mv, dmv a SAT bloku SCUV zpracovávány speciálním integračním algoritmem a symetrickým třístavovým algoritmem s parametry (práhy)  $\text{thron}$  a  $\text{throff}$  (viz blok TSE, uvažujte parametry  $\text{ep} = \text{thron}$ ,  $\text{epoff} = \text{throff}$ ,  $\text{en} = -\text{thron}$  a  $\text{enoff} = -\text{throff}$ ). Vzniklé pulsy (více, méně) jsou dále upravovány tak, že délka libovolného generovaného pulsu (UP, DN) na výstupu bloku je alespoň  $\text{dtime}$  a prodleva mezi dvěma po sobě jdoucími pulsy opačné polaroty je alespoň  $\text{btime}$ . Parametr RACT určuje směr otáčení motoru. Poznamenejme, že nadřazený regulátor PIDU musí mít nastavení  $\text{icotype} = 4$ . Blok SCUV rekonstruuje rychlostní výstup nadřazeného regulátoru ze vstupů mv a dmv. Navíc, jestliže regulační odchylka nadřazeného regulátoru je menší než pásmo necitlivosti ( $\text{SAT} = \text{on}$ ), potom je výstup vnitřního integrátoru bloku SCUV nulován. Takto je dosaženo klidu servoventilu při dostatečně malé regulační odchylce nadřazeného regulátoru ( $|\text{de}| < \text{dz}$  – viz popis bloku PIDU).

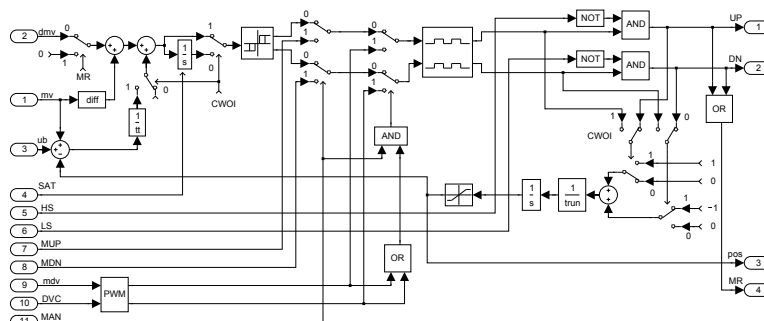
Poloha servoventilu pos je odhadována dalším vnitřním integrátorem s časovou konstantou  $\text{trun}$ . Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

Jestliže je nadřazený regulátor typu P nebo PD ( $CW_{OI} = \text{on}$ ), potom může být regulační odchylka nadřazeného regulátoru manuálně odstraněna vhodným nastavením vstupu ub. V tomto případě je řídicí algoritmus bloku SCUV lehce modifikován. Je užita rekonstruovaná hodnota polohy servoventilu pos a parametry  $\text{thron}$ ,  $\text{throff}$  a  $\text{tt}$  musí být pečlivě nastaveny pro potlačení střídání pulsů více a méně v ustáleném stavu.

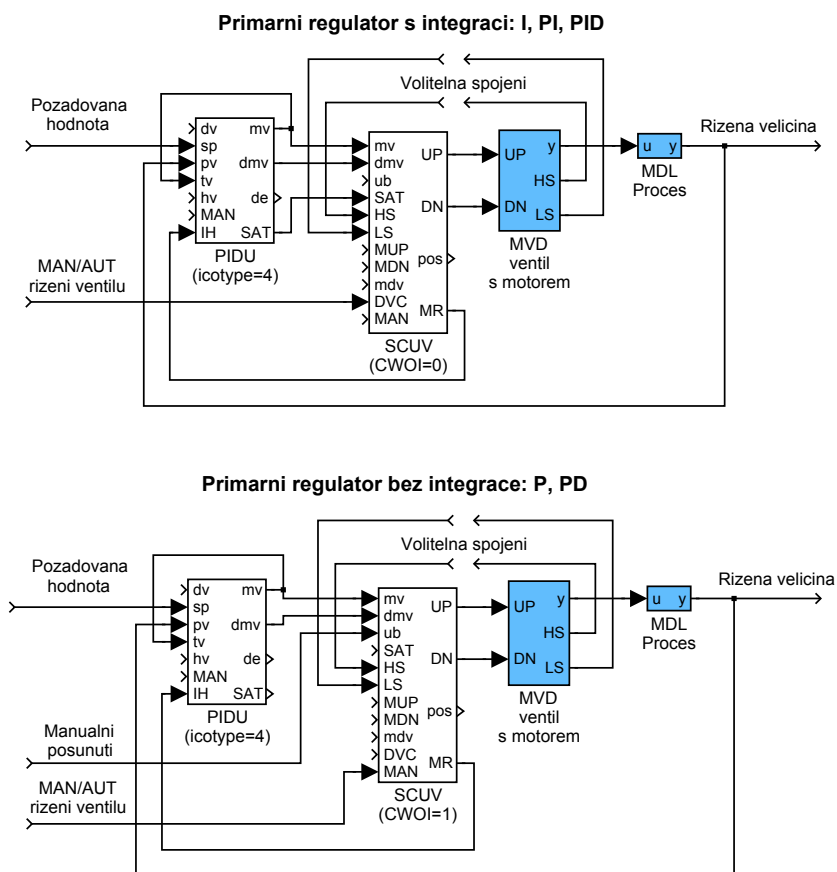
K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu  $\text{MAN} = \text{on}$ , pak lze s motorem pohybovat tam a zpět pomocí signálů MUP a MDN, eventuálně lze pomocí vstupu mdv nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou ( $\text{off} \rightarrow \text{on}$ ) na vstupu

DVC.

Celková funkce bloku SCUV je zřejmá z následujícího diagramu:



Úplně třístavové krokové regulátory bez polohové zpětné vazby jsou zobrazeny na následujících obrázcích:



## Vstupy

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (diference)	Double (F64)

ub	Posunutí (jen pokud je primární regulátor typu P nebo PD)	Double (F64)
SAT	Nulování interního integrátoru (propojen s výstupem SAT primárního regulátoru)	Bool
HS	Horní koncový spínač (příznak, že poloha ventilu je na horní mezi)	Bool
LS	Dolní koncový spínač (příznak, že poloha ventilu je na spodní mezi)	Bool
MUP	Manuální povel UP (nahoru, přidej)	Bool
MDN	Manuální povel DN (dolů, uber)	Bool
mdv	Ruční diferenční hodnota (požadovaný přírůstek/úbytek polohy, mající vyšší prioritu než přímé signály MUP/MDN)	Double (F64)
DVC	Přijetí ruční diferenční hodnoty <code>off</code> → <code>on</code>	Bool
MAN	Manuální nebo automatický režim <code>off</code> ... Automatický režim <code>on</code> ... Manuální režim	Bool

## Výstupy

UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool
pos	Simulovaná poloha motoru	Double (F64)
MR	Požadavek na běh motoru <code>off</code> ... motor neběží (UP = <code>off</code> a DN = <code>off</code> ) <code>on</code> ... motor se má pohybovat (UP = <code>on</code> nebo DN = <code>on</code> )	Bool

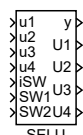
## Parametry

thron	Mez pro zapnutí	↓0.0 ⊙0.02	Double (F64)
throff	Mez pro vypnutí	↓0.0 ⊙0.01	Double (F64)
dtime	Minimální trvání výstupního pulzu [s]	↓0.0 ⊙0.1	Double (F64)
btime	Minimální prodleva mezi dvěma následujícími pulzy [s]	↓0.0 ⊙0.1	Double (F64)
RACT	Převrácené působení výstupu regulátoru <code>off</code> ... vyšší mv → vyšší pv <code>on</code> ... vyšší mv → nižší pv		Bool
trun	Časová konstanta motoru (určuje dobu, za kterou se motor posune o hodnotu jedna)	↓0.0 ⊙10.0	Double (F64)
CWOI	Regulátor bez integrační složky <code>off</code> ... primární regulátor s integrátorem (I, PI, PID) <code>on</code> ... primární regulátor bez integrátoru (P, PD)		Bool
tt	Časová konstanta vysledování	↓0.0 ⊙1.0	Double (F64)

## SELU – Selektor aktivního regulátoru

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok SELU je určen pro přepínání aktivního regulátoru v případě selektorové regulace. Provádí výběr jednoho ze vstupních signálů  $u_1$ ,  $u_2$ ,  $u_3$ ,  $u_4$  a kopíruje ho na výstup  $y$  buď podle celočíselného vstupu  $iSW$  (je-li parametr bloku  $BINF = off$ ) nebo podle binárních vstupů  $SW1$  a  $SW2$  ( $BINF = on$ ) dle následující tabulky.

$iSW$	$SW1$	$SW2$	$y$	$U1$	$U2$	$U3$	$U4$
0	off	off	$u_1$	off	on	on	on
1	off	on	$u_2$	on	off	on	on
2	on	off	$u_3$	on	on	off	on
3	on	on	$u_4$	on	on	on	off

Z této tabulky je patrný též význam logických výstupů  $U1$ ,  $U2$ ,  $U3$ ,  $U4$ , které se používají jako vstupy bloků SWU pro realizaci funkce vysledování neaktivních regulátorů v selektorové regulaci.

### Vstupy

$u_1..u_4$	Signály, ze kterých bude jeden vybrán	Double (F64)
$iSW$	Selektor aktivního signálu, použit pokud $BINF = off$	Long (I32)
$SW1$	Binární vstup pro výběr, použit pokud $BINF = on$	Bool
$SW2$	Binární vstup pro výběr, použit pokud $BINF = on$	Bool

### Výstupy

$y$	Analogový výstupní signál	Double (F64)
$U1..U4$	Binární signály pro selektorovou regulaci	Bool

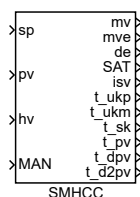
### Parametr

$BINF$	Výběr pomocí binárních vstupů	Bool
	off ... zakázáno (výběr přes $iSW$ )	
	on .... povoleno (výběr přes $SW1$ a $SW2$ )	

## SMHCC – Regulátor pro procesy s topením a chlazením

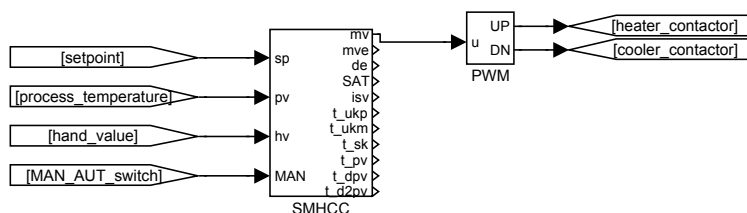
Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Regulátor SMHCC (Sliding Mode Heating/Cooling Controller) je snadno nastavitelný regulátor pro kvalitní regulaci teplotních soustav s dvoustavovým (ON-OFF) topením a dvoustavovým (ON-OFF) chlazením. Klasickým příkladem takových soustav je plastikařský lis. SMHCC může být samozřejmě nasazen i na jiné soustavy, kde se dosud běžně používají konvenční termostaty. Pro zajištění správné funkce je nutné blok SMHCC doplnit blokem PWM (Pulse Width Modulation), jak je patrné z následujícího obrázku.



Blok SMHCC pracuje se dvěma časovými periodami. První perioda  $T_S$  je vzorkovací perioda měřené teploty a je rovněž rovna periodě, se kterou se blok regulátoru SMHCC spouští. Druhá perioda  $T_C = i_{pwm}T_S$  je perioda řízení, se kterou blok SMHCC generuje akční zásahy. Tato perioda  $T_C$  je totožná s periodou cyklu bloku PWM. V každém okamžiku, když se změní akční zásah  $mv$  bloku SMHCC, algoritmus bloku PWM přepočte šířku pulsu a spustí nový PWM cyklus. Třetí perioda, kterou je třeba stanovit, je perioda spouštění  $T_R$  bloku PWM. Obecně může být  $T_R \neq T_S$ . Pro dosažení co nejlepší kvality řízení je doporučeno nastavit periodu  $T_S$  na minimální možnou hodnotu ( $i_{pwm}$  na maximální možnou hodnotu), poměr  $T_C/T_S$  maximální, ale  $T_C$  by měla být dostatečně malá vzhledem k dynamice procesu. Pro aplikace v plastikařském průmyslu jsou doporučeny následující hodnoty:

$$T_S = 0.1, i_{pwm} = 100, T_C = 10s, T_R = 0.01s.$$

Zákon řízení bloku regulátoru SMHCC v automatickém režimu (MAN=off) je založen na diskretní technice dynamického řízení s klouzavým režimem a dále je použit speciální filtr třetího řádu pro odhad první a druhé derivace regulační odchylky.

Po změně požadované hodnoty **sp** (setpoint) se regulátor dostane do první fáze, tzv. přibližovací, kdy diskrétní proměnná klouzavého režimu

$$s_k \triangleq \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

je stlačena do nuly. Ve výše uvedené definici neznámé  $e_k, \dot{e}_k, \ddot{e}_k$  po řadě označují filtrovanou regulační odchylku (**pv-sp**), první a druhou derivaci  $e_k$  v čase  $k$ . Parametry  $\xi$  a  $\Omega$  jsou popsány níže. V druhé fázi (kvazi klouzavý režim) je proměnná  $s_k$  držena v okolí nulové hodnoty pomocí patřičných zásahů řízení, režim topení se střídá s režimem chlazení. Amplitudy topení a chlazení se adaptují tak, aby se dosáhlo přibližně  $s_k = 0$ . V důsledku toho je hypotetická spojitá proměnná klouzavého režimu

$$s \triangleq \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

stále přibližně nulová. Jinak řečeno regulační odchylka  $e$  je popsána diferenciální rovnicí druhého řádu

$$s \triangleq \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e = 0.$$

Z toho plyne, že vývoj  $e$  může být ovlivněn volbou parametrů  $\xi$  a  $\Omega$ . Poznamenejme, že pro stabilní chování musí být splněno  $\xi > 0$  a  $\Omega > 0$ . Typická optimální hodnota  $\xi$  leží v intervalu  $[0.1, 8]$ . Optimální hodnota  $\Omega$  je silně závislá na řízeném procesu, pomalejší procesy mají menší hodnotu a rychlejší větší. Doporučená hodnota  $\Omega$  pro začátek ladění parametrů je  $\pi/(5T_C)$ .

Řídicí veličena  $mv$  je obvykle v intervalu  $[-1, 1]$ . Kladná hodnota odpovídá topení, záporná chlazení, např.  $mv = 1$  znamená plné topení. Omezení na  $mv$  může být zadáno parametry **hilim\_p** a **hilim\_m**. Omezení může být potřebné, když existuje velká nesymetrie mezi topením a chlazením. Jestliže je například chlazení mnohem agresivnější než topení, je vhodné nastavit **hilim\_p** = 1 and **hilim\_m** < 1. Pokud chceme omezení aplikovat pouze v intervalu po změně požadované hodnoty **sp**, volíme **u0\_p** a **u0\_m** tak, že platí **u0\_p** ≤ **hilim\_p** a **u0\_m** ≤ **hilim\_m**.

Hodnoty amplitud proměnných pro topení a chlazení **t\_ukp**, **t\_ukm** se automaticky adaptují speciálním algoritmem tak, aby byl dosažen kvazi klouzavý režim, ve kterém se střídají znaménka **sk** po každém kroku. V tomto případě se výstup **isv** přepíná mezi 1 a -1. Rychlost adaptace amplitud topení a chlazení je dána časovými konstantami **taup** a **taum**. Obě tyto časové konstanty musí být dostatečně velké, aby zajistily správnou funkci adaptace, ale jejich jemné doladění není nezbytné pro výslednou kvalitu regulace. Pro úplnost dodejme, že  $mv$  je určena na základě amplitud **t\_ukp** a **t\_ukm** podle následujícího výrazu

$$if \ (sk < 0.0) \ then \ mv = t\_ukp \ else \ mv = -t\_ukm .$$

Dále je třeba říci, že dosažení kvazi klouzavého režimu nastává velmi zřídka, protože řízené procesy obsahují dopravní zpoždění a působí na ně poruchy. Vhodným indikátorem kvality "klouzání" je opět výstup **isv**. Pro jemné doladění je možno v mimořádných případech použít parametr **beta** definující šířku pásma derivačního filtru. Ve většině případů však vyhovuje přednastavená hodnota **beta** = 0.1.



V manuálním režimu ( $MAN = on$ ) je vstup regulátoru  $hv$  kopírován po případném omezení saturačními mezemi na výstup  $mv$ .

## Vstupy

$sp$	požadovaná hodnota (setpoint)	Double (F64)
$pv$	regulovaná veličina (process variable)	Double (F64)
$hv$	výstup regulátoru v manuálním režimu (hand value)	Double (F64)
$MAN$	režim činnosti regulátoru	Bool
	0 ..... automatický režim1 ..... manuální režim	

## Výstupy

$mv$	řídící veličina (manipulated variable)	Double (F64)
$de$	regulační odchylka (deviation error) $de = sp - pv$	Double (F64)
$SAT$	příznak saturace	Bool
	0 ..... regulátor pracuje v lineární oblasti	
	1 ..... výstup regulátoru je satureován, $mv \geq hilim\_p$ nebo $mv \leq -hilim\_m$	
$isv$	počet kladných nebo záporných kroků přepínací proměnné $sk$	Long (I32)
$t\_ukp$	aktuální hodnota amplitudy topení	Double (F64)
$t\_ukm$	aktuální hodnota amplitudy chlazení	Double (F64)
$t\_sk$	přepínací proměnná $sk$	Double (F64)
$t\_ek$	filtrovaná regulační odchylka $-de$	Double (F64)
$t\_dek$	filtrovaná první derivace regulační odchylky $t\_ek$	Double (F64)
$t\_2dek$	filtrovaná druhá derivace regulační odchylky $t\_ek$	Double (F64)

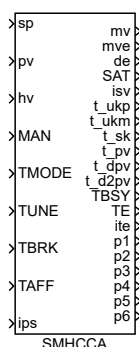
## Parametry

$ipwmc$	počet PWM cyklů během jedné periody spouštění bloku SMHCC ( $T_C/T_S$ )	Long (I32)
$xi$	relativní tlumení $xi > 0$	Double (F64)
$om$	přirozená frekvence $om > 0$	Double (F64)
$taup$	časová konstanta adaptace amplitudy topení v sekundách	Double (F64)
$taum$	časová konstanta adaptace amplitudy chlazení v sekundách	Double (F64)
$beta$	šířka pásma derivačního filtru; $beta > 0$ ; doporučená hodnota 0.1	Double (F64)
$hilim\_p$	horní saturační mez amplitudy topení ( $0 < hilim\_p \leq 1$ )	Double (F64)
$hilim\_m$	horní saturační mez amplitudy chlazení ( $0 < hilim\_m \leq 1$ )	Double (F64)
$u0\_p$	počáteční hodnota amplitudy topení po změně požadované hodnoty nebo startu bloku	Double (F64)
$u0\_m$	počáteční hodnota amplitudy chlazení po změně požadované hodnoty nebo startu bloku	Double (F64)
$sp\_dif$	Práh pro detekci změny setpointu	$\odot 10.0$ Double (F64)
$tauf$	Časová konstanta filtru ekvivalentní akční veličiny	$\odot 400.0$ Double (F64)

## SMHCCA – \* Regulátor pro procesy s topením a chlazením s autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim	Bool
TMODE	Režim ladění	Bool
TUNE	Zahájení ladicího experimentu	Bool
TBRK	Ukončení ladicího experimentu	Bool
TAFF	Přijetí výsledků ladicího experimentu off ... parametry jsou pouze vypočítány on .... parametry jsou dosazeny do řídicího algoritmu	Bool
ips	Význam výstupních signálů 0 ..... parametry regulátoru 1 ..... pomocné parametry	Long (I32)

### Parametry

ipwmc	Délka PWM cyklu (počet vzorkovacích period bloku)	⊙100	Long (I32)
xi	Relativní tlumení	↓0.5 ↑8.0 ⊙1.0	Double (F64)
om	Přirozená frekvence	↓0.0 ⊙0.01	Double (F64)

<code>taup</code>	Časová konstanta adaptace amplitudy topení [s]	⊖700.0	Double (F64)
<code>taum</code>	Časová konstanta adaptace amplitudy chlazení [s]	⊖400.0	Double (F64)
<code>beta</code>	Šířka pásma derivačního filtru	⊖0.01	Double (F64)
<code>hilim_p</code>	Horní saturační mez amplitudy topení	↓0.0 ↑1.0 ⊖1.0	Double (F64)
<code>hilim_m</code>	Horní saturační mez amplitudy chlazení	↓0.0 ↑1.0 ⊖1.0	Double (F64)
<code>u0_p</code>	Počáteční hodnota amplitudy topení	⊖1.0	Double (F64)
<code>u0_m</code>	Počáteční hodnota amplitudy chlazení	⊖1.0	Double (F64)
<code>sp_dif</code>	Práh pro detekci změny setpointu	⊖10.0	Double (F64)
<code>tauf</code>	Časová konstanta filtru ekvivalentní akční veličiny	⊖400.0	Double (F64)
<code>itm</code>	Metoda ladění regulátoru	⊖1	Long (I32)
	1 . . . . . omezeno na symetrické procesy		
	2 . . . . . asymetrické procesy (zatím není implementováno)		
<code>ut_p</code>	Amplituda topení pro ladicí experiment	↓0.0 ↑1.0 ⊖1.0	Double (F64)
<code>ut_m</code>	Amplituda chlazení pro ladicí experiment	↓0.0 ↑1.0 ⊖1.0	Double (F64)

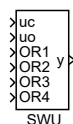
## Výstupy

<code>mv</code>	Akční zásah regulátoru (manipulated variable)	Double (F64)
<code>mve</code>	Ekvivalentní akční veličina	Double (F64)
<code>de</code>	Regulační odchylka	Double (F64)
<code>SAT</code>	Saturace	Bool
	<code>off</code> . . . lineární zákon řízení	
	<code>on</code> . . . výstup regulátoru je saturován	
<code>isv</code>	Počet kroků přepínací proměnné	Long (I32)
<code>t_ukp</code>	Aktuální amplituda topení	Double (F64)
<code>t_ukm</code>	Aktuální amplituda chlazení	Double (F64)
<code>t_sk</code>	Přepínací proměnná	Double (F64)
<code>t_pv</code>	Filtrovaná řízená veličina	Double (F64)
<code>t_dpv</code>	Filtrovaná první derivace řízené veličiny	Double (F64)
<code>t_d2pv</code>	Filtrovaná druhá derivace řízené veličiny	Double (F64)
<code>TBSY</code>	Příznak probíhajícího ladicího experimentu	Bool
<code>TE</code>	Příznak chyby během ladění	Bool
	<code>off</code> . . . ladění proběhlo bez chyby	
	<code>on</code> . . . během ladění se vyskytla chyba	
<code>ite</code>	Kód chyby	Long (I32)
	0 . . . . . bez chyby	
	1 . . . . . příliš zašuměné pv, zkontroluj teplotní vstup 2	
	2 . . . . . nesprávný parametr <code>ut_p</code>	
	3 . . . . . setpoint je příliš nízký	
	4 . . . . . vzorkovací perioda je příliš nízká nebo je druhá derivace příliš zašuměná	
	5 . . . . . předčasné ukončení ladicího experimentu	
<code>p1..p6</code>	Výsledky identifikace a návrhu regulátoru	Double (F64)

## SWU – Přepínač vstupu pro vysledování

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok SWU je určen pro přepínání vhodného signálu na vstup pro vysledování bloků [PIDU](#) a [MCU](#). V případě, že všechny vstupy OR1, ..., OR4 jsou logické nuly (**off**), potom na výstup y je kopírována hodnota vstupu uc, v opačném případě hodnota vstupu uo.

### Vstupy

uc	Tento vstup je kopírován na výstup y, jestliže všechny vstupy OR1, OR2, OR3 a OR4 jsou <b>off</b>	Double (F64)
uo	Tento vstup je kopírován na výstup, jestliže alespoň jeden vstup OR1, OR2, OR3 nebo OR4 je <b>on</b>	Double (F64)
OR1	První logický výstup bloku	Bool
OR2	Druhý logický výstup bloku	Bool
OR3	Třetí logický výstup bloku	Bool
OR4	Čtvrtý logický výstup bloku	Bool

### Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

## TSE – Třístavový prvek

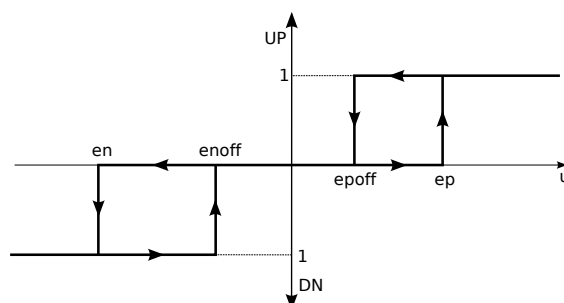
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TSE transformuje analogový vstup  $u$  na třístavový výstup (méně, nečinnost, více) podle níže uvedeného obrázku.



Vstup

$u$	Analogový vstupní signál	Double (F64)
-----	--------------------------	--------------

Výstupy

UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool

Parametry

$ep$	Hodnota $u > ep$ způsobí nastavení výstupů UP = on a DN = off	Double (F64)
	⊙1.0	
$en$	Hodnota $u < en$ způsobí nastavení výstupů UP = off a DN = off	Double (F64)
	⊙-1.0	
$epoff$	Je-li UP = on a $u < epoff$ , potom UP = off	⊙0.5 Double (F64)
$enoff$	Je-li DN = on a $u > enoff$ , potom DN = off	⊙-0.5 Double (F64)



## Kapitola 8

# LOGIC – Logické řízení

### Obsah

---

AND – Logický součin dvou signálů . . . . .	264
ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů . . . . .	265
ATMT – Automat pro sekvenční řízení . . . . .	267
BDOCT, BDHEXD – Bitové demultiplexery . . . . .	270
BITOP – Bitová operace dvou celočíselných signálů . . . . .	272
BMOCT, BMHEXD – Bitové multiplexery . . . . .	273
COUNT – Řízený čítač . . . . .	275
EATMT – Rozšířený automat pro sekvenční řízení . . . . .	277
EDGE – Detekce hrany logického signálu . . . . .	280
EQ – Shodnost dvou signálů . . . . .	281
INTSM – Bitový posun a maska nad celým číslem . . . . .	282
ISSW – Jednoduchý přepínač celočíselných signálů . . . . .	283
ITOI – Transformace celých a binárních čísel . . . . .	284
NOT – Logická negace . . . . .	286
OR – Logický součet dvou signálů . . . . .	287
ORQUAD, OROCT, ORHEXD – Logický součet více signálů . . . . .	288
RS – Klopný obvod . . . . .	290
SR – Klopný obvod . . . . .	291
TIMER – Vícefunkční časovač . . . . .	292

---

Knihovna LOGIC zahrnuje řadu bloků pro provádění logických a sekvenčních operací. Zahrnuje základní booleovské bloky jako **AND**, **OR**, **NOT** pro základní logické operace a pokročilé bloky jako **ATMT** pro konečné automaty. Bloky jako **COUNT** a **TIMER** rozšiřují funkčnost na obousměrné počítání pulsů a operace založené na čase. Další prvky jako **BITOP**, **BMOCT** a **BDOCT** nabízejí bitové operace a možnosti multiplexování/demultiplexování, čímž zvyšují všestrannost knihovny při ovládní kombinační a sekvenční logiky.

## AND – Logický součin dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok AND dělá logický součin dvou vstupních signálů  $Y = U1 \wedge U2$ . Pro logický součin více vstupních signálů použijte jeden z bloků [ANDQUAD](#), [ANDOCT](#) nebo [ANDHEXD](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

Výstup

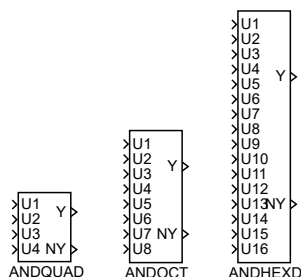
Y	Výstup bloku, logický součin	Bool
NY	Negace výstupního signálu Y	Bool



## ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **ANDQUAD**, **ANDOCT** a **ANDHEXD** vyhodnocují logický součin až 16 vstupních signálů U1 až U16. Signály, jejichž seznam je uveden v parametru **n1** se před provedením logického součinu negují.

Pokud je tedy parametr **n1** prázdný, tak se provádí logický součin  $Y = U1 \wedge U2 \wedge U3 \wedge U4 \wedge U5 \wedge U6 \wedge U7 \wedge U8$ . Pokud bude například **n1=1,3..5**, pak  $Y = \neg U1 \wedge U2 \wedge \neg U3 \wedge \neg U4 \wedge \neg U5 \wedge U6 \wedge \dots \wedge U16$ .

Pokud pracujete s méně než 8 signály, je potřeba ošetřit nepřípojené vstupy bloku pomocí parametru **n1**. Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [AND](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

U1	Logický vstupní signál	Bool
U2	Logický vstupní signál	Bool
U3	Logický vstupní signál	Bool
U4	Logický vstupní signál	Bool
U5	Logický vstupní signál	Bool
U6	Logický vstupní signál	Bool
U7	Logický vstupní signál	Bool
U8	Logický vstupní signál	Bool
U9	Logický vstupní signál	Bool
U10	Logický vstupní signál	Bool
U11	Logický vstupní signál	Bool
U12	Logický vstupní signál	Bool
U13	Logický vstupní signál	Bool
U14	Logický vstupní signál	Bool

U15	Logický vstupní signál	Bool
U16	Logický vstupní signál	Bool

## Parametr

n1	Seznam negovaných signálů	Long (I32)
----	---------------------------	------------

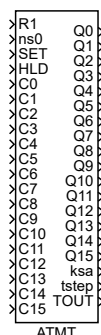
## Výstup

Y	Výstup bloku, logický součin	Bool
NY	Negace výstupního signálu Y	Bool

## ATMT – Automat pro sekvenční řízení

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok ATMT realizuje konečný automat až s 16 stavy a 16 podmínkami přechodů mezi nimi.

Aktuální stav automatu  $i$ ,  $i = 0, 1, \dots, 15$ , je indikován pomocí binárních výstupů  $Q_0, Q_1, \dots, Q_{15}$ . Pokud je automat ve stavu  $i$ , je nastaven příslušný výstup  $Q_i = \text{on}$ . Aktuální stav automatu je též indikován celočíselným výstupem  $ksa \in \{0, 1, \dots, 15\}$ .

Podmínky přechodů  $C_k$ ,  $k = 0, 1, \dots, 15$  jsou aktivovány pomocí binárních vstupů bloku  $C_0, C_1, \dots, C_{15}$ . Pokud je  $C_k = \text{on}$ , je podmínka  $C_k$  splněna, naopak pro  $C_k = \text{off}$  splněna není.

Funkce automatu se zadává pomocí tabulky stavů a přechodů:

$$\begin{array}{l} S1 \quad C1 \quad NS1 \\ S2 \quad C2 \quad NS2 \\ \dots \\ Sn \quad Cn \quad NSn \end{array}$$

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. první řádek

$$S1 \quad C1 \quad NS1$$

má tento význam:

**Jestliže** (aktuální stav je  $S1$  AND podmínka přechodu  $C1$  je splněna),  
potom přejdi do následujícího stavu  $NS1$ .

Výše uvedenou tabulku lze získat ze stavového diagramu automatu nebo z popisu automatu v jazyce SFC (Sequential Function Charts, dříve Grafcet).

Vstup  $R1 = \text{on}$  resetuje stav automatu do počátečního stavu  $S0$ , přičemž vstup  $R1$  má prioritu před vstupem  $SET$ . Náběžná hrana na vstupu  $SET$  způsobí přechod z aktuálního

stavu do stavu `ns0`. Vstup `HLD = on` zablokuje činnost automatu, tzn. automat setrvává v daném stavu i v případě, že je splněna některá podmínka přechodu, rovněž je zastaveno inkrementování času `tstep` a generování výstupu `TOUT`. Výstup `TOUT` indikuje, že automat setrval v daném stavu déle, než je povoleno. Časová omezení `TOi` jednotlivých stavů se definují pomocí vektoru `touts`. Pokud je `TOi = 0`, není pro daný stav nastaveno žádné časové omezení. Výstup `TOUT` je automaticky nastavován na hodnotu `off` při každém přechodu mezi stavy automatu.

Pomocí parametru `moresteps` lze povolit přechod automatu o více kroků v jednom cyklu. Tuto možnost je však vždy potřeba pečlivě zvážit, zejména při použití výstupu `TOUT` v podmínkách pro přechod do dalších stavů. V takovém případě je vhodné konstruovat podmínku přechodu nejen pomocí výstupu `TOUT`, ale zahrnout do ní i informaci o stavu automatu `ksa`.

Součástí systému REXYGEN je také program `SFCeditor`, který umožňuje tvorbu SFC schémat v grafickém návrhovém prostředí. Editor se spouští z programu REXYGEN Studio kliknutím na tlačítko *Configure* na kartě parametrů bloku `ATMT`. Uživatelská příručka editoru je k dispozici jako samostatný dokument.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

R1	Reset bloku	Bool
ns0	Cílový stav pro přechod vynucený vstupem SET	Long (I32)
SET	Vynucení přechodu do stavu ns0	Bool
HLD	Pozastavení	Bool
C0	Podmínka přechodu	Bool
C1	Podmínka přechodu	Bool
C2	Podmínka přechodu	Bool
C3	Podmínka přechodu	Bool
C4	Podmínka přechodu	Bool
C5	Podmínka přechodu	Bool
C6	Podmínka přechodu	Bool
C7	Podmínka přechodu	Bool
C8	Podmínka přechodu	Bool
C9	Podmínka přechodu	Bool
C10	Podmínka přechodu	Bool
C11	Podmínka přechodu	Bool
C12	Podmínka přechodu	Bool
C13	Podmínka přechodu	Bool
C14	Podmínka přechodu	Bool
C15	Podmínka přechodu	Bool

## Parametr

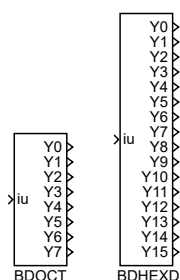
<code>moresteps</code>	Povolit více přechodů mezi stavy v jednom cyklu <code>off ...</code> zakázáno <code>on ....</code> povoleno	Bool
<code>sfcname</code>	Jméno souboru pro ukládání dat speciálního editoru	String
<code>STT</code>	Tabulka přechodů mezi stavy $\odot[0\ 0\ 1; 1\ 1\ 2; 2\ 2\ 3; 3\ 3\ 0]$	Byte (U8)
<code>touts</code>	Časové limity pro jednotlivé stavy $\odot[1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16]$	Double (F64)

## Výstup

<code>Q0</code>	Indikátor aktivního stavu	Bool
<code>Q1</code>	Indikátor aktivního stavu	Bool
<code>Q2</code>	Indikátor aktivního stavu	Bool
<code>Q3</code>	Indikátor aktivního stavu	Bool
<code>Q4</code>	Indikátor aktivního stavu	Bool
<code>Q5</code>	Indikátor aktivního stavu	Bool
<code>Q6</code>	Indikátor aktivního stavu	Bool
<code>Q7</code>	Indikátor aktivního stavu	Bool
<code>Q8</code>	Indikátor aktivního stavu	Bool
<code>Q9</code>	Indikátor aktivního stavu	Bool
<code>Q10</code>	Indikátor aktivního stavu	Bool
<code>Q11</code>	Indikátor aktivního stavu	Bool
<code>Q12</code>	Indikátor aktivního stavu	Bool
<code>Q13</code>	Indikátor aktivního stavu	Bool
<code>Q14</code>	Indikátor aktivního stavu	Bool
<code>Q15</code>	Indikátor aktivního stavu	Bool
<code>ksa</code>	Celočíselná reprezentace stavu	Long (I32)
<code>tstep</code>	Čas uplynulý od posledního přechodu mezi stavy	Double (F64)
<code>TOUT</code>	Příznak překročení časového limitu pro aktuální stav	Bool

## BDOCT, BDHEXD – Bitové demultiplexery

Symboly bloků

Licence: [STANDARD](#)

## Popis funkce

Bloky BDOCT a BDHEXD pracují jako bitové demultiplexery a lze je výhodně použít pro rozebírání celočíselných signálů na jednotlivé bity. Oba bloky se od sebe liší jen počtem výstupů, blok BDOCT má 8 bitových výstupů, blok BDHEXD jich má 16. Výstupní signály  $Y_i$  jsou přímo tvořeny bity signálu, který vznikne bitovým posunem vstupu  $iu$  o  $shift$  bitů vpravo, přičemž v signálu  $Y_0$  je vždy nejnižší bit tohoto čísla.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$iu$             Vstupní signál k dekompozici            Long (I32)

## Parametr

$shift$	Bitový posun vstupního signálu	↓0 ↑31	Long (I32)
$vtype$	Číselný typ	⊙4	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		

## Výstup

$Y_0$	Jednotlivý bit vstupního signálu	Bool
$Y_1$	Jednotlivý bit vstupního signálu	Bool
$Y_2$	Jednotlivý bit vstupního signálu	Bool
$Y_3$	Jednotlivý bit vstupního signálu	Bool

Y4	Jednotlivý bit vstupního signálu	Bool
Y5	Jednotlivý bit vstupního signálu	Bool
Y6	Jednotlivý bit vstupního signálu	Bool
Y7	Jednotlivý bit vstupního signálu	Bool
Y8	Jednotlivý bit vstupního signálu	Bool
Y9	Jednotlivý bit vstupního signálu	Bool
Y10	Jednotlivý bit vstupního signálu	Bool
Y11	Jednotlivý bit vstupního signálu	Bool
Y12	Jednotlivý bit vstupního signálu	Bool
Y13	Jednotlivý bit vstupního signálu	Bool
Y14	Jednotlivý bit vstupního signálu	Bool
Y15	Jednotlivý bit vstupního signálu	Bool

## BITOP – Bitová operace dvou celočíselných signálů

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok BITOP provádí operaci  $i1 \circ i2$  na vstupních signálech po jednotlivých bitech. Výsledkem je celočíselný výstup  $n$ . Kód zvolené bitové operace je uveden v parametru `iop` popsaném níže. V případě bitové negace a dvojkových doplňků se operace provádí pouze se vstupem `i1` (tj. operace je unární).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>i1</code>	První celočíselný vstup bloku	Long (I32)
<code>i2</code>	Druhý celočíselný vstup bloku	Long (I32)

### Parametr

<code>iop</code>	Bitová operace	⊙1 Long (I32)
	1 ..... bitové NOT	
	2 ..... bitové OR	
	3 ..... bitové AND	
	4 ..... bitové XOR	
	5 ..... posun vlevo	
	6 ..... posun vpravo	
	7 ..... dvojkový doplněk signálu - byte	
	8 ..... dvojkový doplněk signálu - word	
	9 ..... dvojkový doplněk signálu - long	
<code>vtype</code>	Číselný typ	⊙4 Long (I32)
	2 ..... Byte (U8)	
	3 ..... Short (I16)	
	4 ..... Long (I32)	
	5 ..... Word (U16)	
	6 ..... DWord (U32)	
	10 .... Large (I64)	

### Výstup

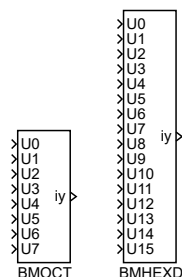
<code>n</code>	Výsledek bitové operace	Long (I32)
----------------	-------------------------	------------



## BMOCT, BMHEXD – Bitové multiplexery

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky BMOCT a BMHEXD pracují jako bitové multiplexery a lze je výhodně využít ke skládání celočíselných signálů z jednotlivých bitů. Oba bloky se od sebe liší jen počtem vstupů, blok BMOCT má 8 bitových vstupů, blok BMHEXD jich má 16. V případě, že parametr `shift = 0`, jsou jednotlivé bity výstupního signálu `iy` přímo tvořeny vstupními signály `Ui`. V nejnižším bitu je vždy signál `U0`.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

U0	Jednotlivý bit výstupního signálu	Bool
U1	Jednotlivý bit výstupního signálu	Bool
U2	Jednotlivý bit výstupního signálu	Bool
U3	Jednotlivý bit výstupního signálu	Bool
U4	Jednotlivý bit výstupního signálu	Bool
U5	Jednotlivý bit výstupního signálu	Bool
U6	Jednotlivý bit výstupního signálu	Bool
U7	Jednotlivý bit výstupního signálu	Bool
U8	Jednotlivý bit výstupního signálu	Bool
U9	Jednotlivý bit výstupního signálu	Bool
U10	Jednotlivý bit výstupního signálu	Bool
U11	Jednotlivý bit výstupního signálu	Bool
U12	Jednotlivý bit výstupního signálu	Bool
U13	Jednotlivý bit výstupního signálu	Bool
U14	Jednotlivý bit výstupního signálu	Bool
U15	Jednotlivý bit výstupního signálu	Bool

## Parametr

shift	Bitový posun výstupního signálu	↓0 ↑31	Long (I32)
vtype	Číselný typ	⊙4	Long (I32)
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	10 .... Large (I64)		

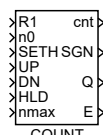
## Výstup

iy	Výsledný výstupní signál		Long (I32)
----	--------------------------	--	------------

## COUNT – Řízený čítač

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **COUNT** je určen pro obousměrné čítání pulsů – přesněji náběžných hran na vstupech **UP** a **DN**. Při výskytu náběžné hrany na vstupu **UP** (**DN**) se výstup **cnt** zvětší o 1 (sníží o 1). Současný výskyt náběžných hran na vstupech **UP** a **DN** indikuje výstup **E** jako chybu čítání. Resetování výstupu **cnt** na hodnotu 0 lze provést vstupem **R1** (pokud je **R1 = on** je výstup **cnt** držen na hodnotě 0). Nastavení výstupu **cnt** na hodnotu **n0** zajistí vstup **SETH = on** (pokud **SETH = on** je výstup **cnt** držen na hodnotě **n0**). Vstup **R1** má vyšší prioritu než vstup **SETH**. Vstup **HLD = on** způsobí zastavení čítání. Stav čítače  $\text{cnt} \geq \text{nmax}$  způsobí nastavení výstupu **Q** na hodnotu **on**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>R1</b>	Reset bloku	Bool
<b>n0</b>	Hodnota pro nastavení čítače	Long (I32)
<b>SETH</b>	Nastavení hodnoty čítače	Bool
<b>UP</b>	Vstup pro přičítání	Bool
<b>DN</b>	Vstup pro odečítání	Bool
<b>HLD</b>	Zmrazení čítače	Bool
	<ul style="list-style-type: none"> <li><b>off</b> ... čítač běží</li> <li><b>on</b> ... čítač je zablokován</li> </ul>	
<b>nmax</b>	Cílová hodnota čítače	Long (I32)

### Výstup

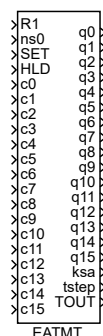
<b>cnt</b>	Celkový počet načtených pulsů	Long (I32)
<b>SGN</b>	Znaménko výstupu <b>cnt</b>	Bool
	<ul style="list-style-type: none"> <li><b>off</b> ... menší nebo rovno nule</li> <li><b>on</b> ... větší než nula</li> </ul>	
<b>Q</b>	Indikátor dosažení cílové hodnoty	Bool
	<ul style="list-style-type: none"> <li><b>off</b> ... cílová hodnota nebyla dosažena</li> <li><b>on</b> ... dosažena cílová hodnota</li> </ul>	

E	Příznak chyby	Bool
	off ... bez chyby	
	on .... nastala chyba	

## EATMT – Rozšířený automat pro sekvenční řízení

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok EATMT implementuje konečný automat s maximálně 256 stavy a 256 pravidly pro přechody, čímž rozšiřuje možnosti bloku [ATMT](#).

Současný stav automatu  $i$ ,  $i = 0, 1, \dots, 255$ , je indikován jednotlivými bity celočíselných výstupů  $q_0, q_1, \dots, q_{15}$ . Pouze jeden bit s indexem  $i, \text{MOD}, 16$  na výstupu  $q(i, \text{DIV}, 16)$  je nastaven na 1. Ostatní bity tohoto výstupu a ostatní výstupy jsou nulové. Bity jsou číslovány od nuly, začínající nejméně významným bitem. Všimněte si, že operátory DIV a MOD označují celočíselné dělení a zbytek po celočíselném dělení. Současný stav je také indikován výstupem  $ksa \in 0, 1, \dots, 255$ .

Podmínky pro přechod  $C_k$ ,  $k = 0, 1, \dots, 255$ , jsou aktivovány jednotlivými bity vstupů  $c_0, c_1, \dots, c_{15}$ .  $k$ -tá podmínka pro přechod je splněna, když je  $(k, \text{MOD}, 16)$ -tý bit vstupu  $c(k, \text{DIV}, 16)$  roven 1. Přechod jinak nemůže proběhnout.

Pro kompozici vstupních signálů  $c_0, c_1, \dots, c_{15}$  z jednotlivých Booleovských signálů mohou být použity bitové multiplexory [BMHEXD](#) nebo [BMOCT](#). Podobně mohou být výstupní signály  $q_0, q_1, \dots, q_{15}$  dekomponovány pomocí bitových demultiplexorů [BDHEXD](#) nebo [BDOCT](#).

Funkce automatu je definována následující tabulkou přechodů:

$S_1$	$C_1$	$FS_1$
$S_2$	$C_2$	$FS_2$
		$\dots$
$S_n$	$C_n$	$FS_n$

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. první řádek

$S_1$	$C_1$	$FS_1$
-------	-------	--------

má tento význam:

Jestliže (aktuální stav je  $S1$  AND podmínka přechodu  $C1$  je splněna),  
potom přejdi do následujícího stavu  $FS1$ .

Výše popsáný význam řádku tabulky platí pro  $C1 < 1000$ . Pro  $C1 \geq 1000$  se předpokládá negace  $(C1 - 1000)$ -té podmínky pro přechod.

Výše zmíněnou tabulku lze snadno sestavit z diagramu stavů automatu nebo z popisu SFC (Sequential Function Charts, dříve Grafcet).

Vstup  $R1 = on$  resetuje automat do počátečního stavu  $S0$ . Vstup  $SET$  umožňuje manuální přechod ze současného stavu do stavu  $ns0$  při detekci náběžné hrany. Vstup  $R1$  má přednost před vstupem  $SET$ . Vstup  $HLD = on$  zastaví vykonávání automatu, takže zůstává ve svém současném stavu bez ohledu na vstupní signály  $ci$  a časovač  $tstep$  se nezvyšuje. Výstup  $TOUT$  signalizuje, že stroj zůstává v daném stavu déle, než se očekávalo. Časové limity  $TOi$  pro jednotlivé stavy jsou definovány polem  $touts$ . Pokud je  $TOi$  nastaveno na nulu, pro daný stav neexistuje časový limit. Výstup  $TOUT$  je nastaven na  $off$  vždy, když automat změní svůj stav.

Je možné umožnit více přechodů stavů v jednom cyklu pomocí parametru  $moresteps$ . Tato možnost by však měla být pečlivě zvážena a otestována, zejména pokud je výstup  $TOUT$  použit v podmínkách pro přechod. V takovém případě se silně doporučuje zahrnout do podmínek pro přechod také výstup  $ksa$ .

Vývojové nástroje **REXYGEN** zahrnují také program **SFC**Editor. Pomocí tohoto nástroje můžete graficky vytvářet schémata SFC. Tento editor spustíte z **REXYGEN** Studio kliknutím na tlačítko *Konfigurovat* v dialogu parametrů bloku **EATMT**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$R1$	Reset bloku	Bool
$ns0$	Cílový stav pro přechod vynucený vstupem $SET$	Long (I32)
$SET$	Vynucení přechodu do stavu $ns0$	Bool
$HLD$	Pozastavení	Bool
$c0$	Podmínka přechodu	Long (I32)
$c1$	Podmínka přechodu	Long (I32)
$c2$	Podmínka přechodu	Long (I32)
$c3$	Podmínka přechodu	Long (I32)
$c4$	Podmínka přechodu	Long (I32)
$c5$	Podmínka přechodu	Long (I32)
$c6$	Podmínka přechodu	Long (I32)
$c7$	Podmínka přechodu	Long (I32)
$c8$	Podmínka přechodu	Long (I32)
$c9$	Podmínka přechodu	Long (I32)
$c10$	Podmínka přechodu	Long (I32)
$c11$	Podmínka přechodu	Long (I32)
$c12$	Podmínka přechodu	Long (I32)
$c13$	Podmínka přechodu	Long (I32)

c14	Podmínka přechodu	Long (I32)
c15	Podmínka přechodu	Long (I32)

## Parametr

moresteps	Povolit více přechodů mezi stavy v jednom cyklu off ... zakázáno on .... povoleno	Bool
sfcname	Jméno souboru pro ukládání dat speciálního editoru	String
STT	Tabulka přechodů mezi stavy ⊙[0 0 1; 1 1 2; 2 2 3; 3 3 0]	Short (I16)
touts	Časové limity pro jednotlivé stavy ⊙[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]	Double (F64)

## Výstup

q0	Indikátor aktivního stavu	Long (I32)
q1	Indikátor aktivního stavu	Long (I32)
q2	Indikátor aktivního stavu	Long (I32)
q3	Indikátor aktivního stavu	Long (I32)
q4	Indikátor aktivního stavu	Long (I32)
q5	Indikátor aktivního stavu	Long (I32)
q6	Indikátor aktivního stavu	Long (I32)
q7	Indikátor aktivního stavu	Long (I32)
q8	Indikátor aktivního stavu	Long (I32)
q9	Indikátor aktivního stavu	Long (I32)
q10	Indikátor aktivního stavu	Long (I32)
q11	Indikátor aktivního stavu	Long (I32)
q12	Indikátor aktivního stavu	Long (I32)
q13	Indikátor aktivního stavu	Long (I32)
q14	Indikátor aktivního stavu	Long (I32)
q15	Indikátor aktivního stavu	Long (I32)
ksa	Celočíselná reprezentace stavu	Long (I32)
tstep	Čas uplynulý od posledního přechodu mezi stavy	Double (F64)
TOUT	Příznak překročení časového limitu pro aktuální stav	Bool

## EDGE – Detekce hrany logického signálu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok **EDGE** detekuje na vstupním signálu **U** náběžnou (**off**→**on**), sestupnou (**on**→**off**) nebo obě uvedené hrany podle hodnoty parametru **iedge**. V případě nalezení požadované hrany (změny vstupního signálu) je na jeden krok nastavena hodnota výstupu **Y** na **on**. Po dobu, kdy se hodnota vstupního signálu nemění je hodnota výstupu **Y** rovna **off**. Hodnota výstupu **Y** zůstane nulová i v případě, že v parametru **iedge** je zvolena náběžná (sestupná) hrana a v signálu se vyskytne hrana opačná, tj. sestupná (náběžná).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>U</b>	Logický vstupní signál	<b>Bool</b>
----------	------------------------	-------------

### Parametr

<b>iedge</b>	Typ detekovaných hran	⊙1 <b>Long (I32)</b>
	1 ..... náběžná hrana	
	2 ..... sestupná hrana	
	3 ..... obě hrany	

### Výstup

<b>Y</b>	Logický výstupní signál	<b>Bool</b>
----------	-------------------------	-------------



## EQ – Shodnost dvou signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok porovnává dva vstupní signály. Výstup  $Y = \text{on}$  je nastaven, pokud mají oba signály stejnou hodnotu. Oba signály musí být shodně buď číselného typu nebo řetězce. Převod mezi numerickými typy je zajištěn: například hodnoty 2,0 (double) a 2 (long) jsou vyhodnoceny jako ekvivalentní. Porovnávání matic nebo jiných komplexních typů není podporováno.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

u1	Vstup bloku	Any
u2	Vstup bloku	Any

### Výstup

Y	Logický výstupní signál	Bool
NY	Negace výstupního signálu Y	Bool

## INTSM – Bitový posun a maska nad celým číslem

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok INTSM provádí bitový posun vstupního čísla *i* o *shift* bitů doprava (pro kladný *shift*) nebo doleva (záporný *shift*). Volné bity vzniklé posunem jsou vyplněny nulami.

Výstupní hodnota *n* je logickým součinem (AND) bitově posunutého vstupu *i* a bitové masky *mask*.

Typické využití bloku spočívá v extrakci hodnoty jednoho nebo více sousedních bitů z určité pozice v celočíselném registru vyčteném z externího systému.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<i>i</i>	Celočíselný signál pro zpracování	Large (I64)
	↓-9.22337E+18 ↑9.22337E+18	

### Parametr

<i>shift</i>	Bitový posun (záporné číslo=doleva, kladné číslo=doprava)	Long (I32)
	↓-63 ↑63	
<i>mask</i>	Bitová maska (aplikovaná po bitovém posunu)	Large (I64)
	↓0 ↑4294970000 ⊕4294967295	
<i>vtype</i>	Číselný typ	⊙4 Long (I32)
	2 ..... Byte (U8)	
	3 ..... Short (I16)	
	4 ..... Long (I32)	
	5 ..... Word (U16)	
	6 ..... DWord (U32)	
	10 ..... Large (I64)	

### Výstup

<i>n</i>	Výsledná celočíselná hodnota	Large (I64)
----------	------------------------------	-------------

## ISSW – Jednoduchý přepínač celočíselných signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok ISSW je jednoduchý přepínač celočíselných vstupních signálů *i1* a *i2* na základě logického vstupu *SW*. Jestliže *SW* je *off*, pak výstup *n* je roven signálu *i1*. Jestliže *SW* je *on*, pak výstup *n* je roven signálu *i2*.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

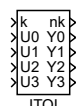
<i>i1</i>	První celočíselný vstup bloku	Long (I32)
<i>i2</i>	Druhý celočíselný vstup bloku	Long (I32)
<i>SW</i>	Přepínací signál	Bool
	<i>off</i> ... je zvolen vstupní signál <i>i1</i>	
	<i>on</i> .... je zvolen vstupní signál <i>i2</i>	

### Výstup

<i>n</i>	Celočíselný výstupní signál	Long (I32)
----------	-----------------------------	------------

## ITOI – Transformace celých a binárních čísel

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok ITOI přiřazuje vstupnímu číslu  $k$  respektive binárnímu číslu  $(U_3 U_2 U_1 U_0)_2$  z množiny  $\{0, 1, 2, \dots, 15\}$  výstupní číslo  $nk$  a jeho binární reprezentaci  $(Y_3 Y_2 Y_1 Y_0)_2$  z téže množiny. Příslušné zobrazení je popsáno tabulkou

$k$	0	1	2	...	15
$nk$	$n_0$	$n_1$	$n_2$	...	$n_{15}$

kde  $n_0, \dots, n_{15}$  jsou dány převodním vektorem  $fktab$ . Je-li  $BINF = off$ , potom se za významný považuje vstup  $k$ , zatímco pro  $BINF = on$  se za vstup bloku považuje číslo  $(U_3 U_2 U_1 U_0)_2$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

$k$	Celočíselný vstupní signál	Long (I32)
$U_0$	Binární číslice vstupu s vahou 1	Bool
$U_1$	Binární číslice vstupu s vahou 2	Bool
$U_2$	Binární číslice vstupu s vahou 4	Bool
$U_3$	Binární číslice vstupu s vahou 8	Bool

## Parametr

$BINF$	Výběr pomocí binárních vstupů off ... zakázáno (výběr přes iSW) on .... povoleno (výběr přes SW1 a SW2)	$\odot on$ Bool
$fktab$	Mapovací tabulka $\odot [0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15]$	Byte (U8)

## Výstup

$nk$	Celočíselný výstupní signál	Long (I32)
$Y_0$	Binární číslice výstupu s vahou 1	Bool
$Y_1$	Binární číslice výstupu s vahou 2	Bool

Y2	Binární číslice výstupu s vahou 4	Bool
Y3	Binární číslice výstupu s vahou 8	Bool

**NOT – Logická negace**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok NOT dělá logickou negaci vstupního signálu  $Y = \neg U$ .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

U	Logický vstupní signál	Bool
---	------------------------	------

## Výstup

Y	Logický výstupní signál	Bool
---	-------------------------	------

## OR – Logický součet dvou signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **OR** dělá logický součet dvou vstupních signálů  $Y = U1 \vee U2$ . Pokud potřebujete pracovat s více vstupními signály, použijte jeden z bloků [ORQUAD](#), [OROCT](#) nebo [ORHEXD](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

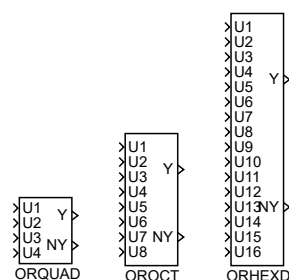
U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

### Výstup

Y	Výstup bloku, logický součet	Bool
NY	Negace výstupního signálu Y	Bool

## ORQUAD, OROCT, ORHEXD – Logický součet více signálů

Symboly bloků

Licence: [STANDARD](#)

## Popis funkce

Bloky **ORQUAD**, **OROCT**, **ORHEXD** provádí logický součet až šestnácti vstupních signálů  $U1$  až  $U16$ . Signály, jejichž seznam je uveden v parametru  $n1$  se před provedením logického součinu negují.

Pokud je tedy parametr  $n1$  prázdný, tak se provádí logický součet  $Y = U1 \vee U2 \vee U3 \vee U4 \vee U5 \vee \dots \vee U16$ . Pokud bude například  $n1=1,3..5$ , pak  $Y = \neg U1 \vee U2 \vee \neg U3 \vee \neg U4 \vee \neg U5 \vee U6 \vee \dots \vee U16$ .

Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [OR](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

U1	Logický vstupní signál	Bool
U2	Logický vstupní signál	Bool
U3	Logický vstupní signál	Bool
U4	Logický vstupní signál	Bool
U5	Logický vstupní signál	Bool
U6	Logický vstupní signál	Bool
U7	Logický vstupní signál	Bool
U8	Logický vstupní signál	Bool
U9	Logický vstupní signál	Bool
U10	Logický vstupní signál	Bool
U11	Logický vstupní signál	Bool
U12	Logický vstupní signál	Bool
U13	Logický vstupní signál	Bool
U14	Logický vstupní signál	Bool
U15	Logický vstupní signál	Bool
U16	Logický vstupní signál	Bool



## Parametr

n1	Seznam negovaných signálů	Long (I32)
----	---------------------------	------------

## Výstup

Y	Výstup bloku, logický součet	Bool
NY	Negace výstupního signálu Y	Bool

## RS – Klopný obvod

Symbol bloku

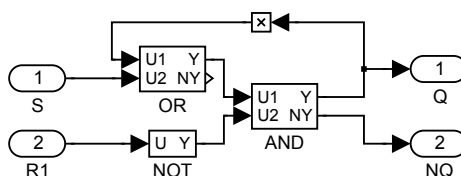
Licence: [STANDARD](#)



Popis funkce

Blok RS je klopný obvod, který v případě, že vstup **S** má hodnotu **on**, nastaví trvale výstup **Q** na **on**. Druhý vstupní signál **R1** resetuje výstup **Q** na hodnotu **off** a to i tehdy, když vstup **S** má hodnotu **on**. Výstup **NQ** je pouhou negací výstupu **Q**.

Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

S	Nahození klopného obvodu	Bool
R1	Přednostní shození klopného obvodu	Bool

### Výstup

Q	Stav klopného obvodu	Bool
NQ	Negace výstupního signálu Q	Bool

## SR – Klopný obvod

Symbol bloku

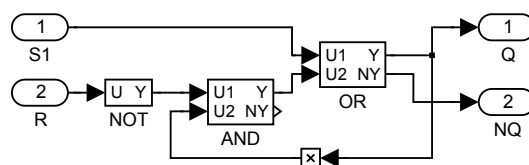
Licence: [STANDARD](#)



Popis funkce

Blok SR je klopný obvod, který v případě, že vstup S1 má hodnotu **on**, nastaví trvale výstup Q na **on**. Druhý vstupní signál R resetuje výstup Q na hodnotu **off**, ale pouze tehdy, když vstup S1 má hodnotu **off**. Výstup NQ je pouhou negací výstupu Q.

Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

S1	Přednostní nahození klopného obvodu	Bool
R	Shození klopného obvodu	Bool

Výstup

Q	Stav klopného obvodu	Bool
NQ	Negace výstupního signálu Q	Bool

## TIMER – Vícefunkční časovač

Symbol bloku

Licence: [STANDARD](#)

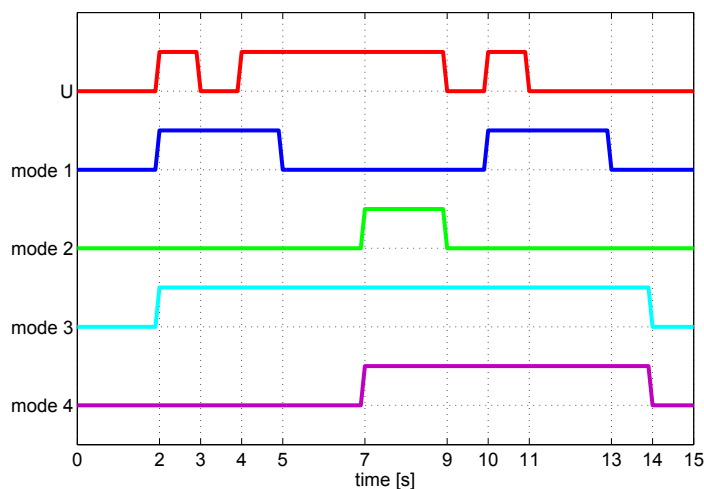


### Popis funkce

Blok **TIMER** umožňuje buď vygenerovat impuls zadané délky **pt** (v sekundách) nebo filtrovat pulzy na vstupním signálu **U** užší než **pt** sekund. Režim funkce bloku se volí pomocí parametru **mode**. Podporované režimy jsou:

- 1: **Pulse**: Na výstupu je pulz délky **pt** sekund, který začíná náběžnou hranou na vstupu **U**; další náběžné hrany na vstupu **U** během trvání pulzu jsou ignorovány.
- 2: **Delayed ON**: Signál ze vstupu **U** je kopírován na výstup **Q** tak, že začátek impulsu na výstupu je opožděn o **pt** sekund proti začátku pulzu na vstupu; pulzy kratší než **pt** sekund se na výstupu neobjeví.
- 3: **Delayed OFF**: Signál ze vstupu **U** je kopírován na výstup **Q** tak, že konec impulsu na výstupu je opožděn o **pt** sekund proti konci pulzu na vstupu; pokud je mezera mezi vstupními pulzy kratší než **pt** sekund, výstup je trvale aktivní.
- 4: **Delayed change**: Výstupní signál **Q** se přepne na hodnotu vstupu **U** až tehdy, když je vstup po dobu **pt** sekund neměnný.

Následující obrázek ilustruje chování bloku v jednotlivých režimech při nastavení **pt = 3**:



Čítání času je možno pozastavit pomocí vstupu **HLD**. Vstup **R1** resetuje časovač. Signál pro reset má přednost před vstupem **U**, obdobně jako u bloku **RS**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

<b>U</b>	Signál spouštějící časovač	<b>Bool</b>
<b>HLD</b>	Pozastavení časovače	<b>Bool</b>
<b>R1</b>	Reset časovače	<b>Bool</b>

## Parametr

<b>mode</b>	Režim činnosti časovače	⊙1	<b>Long (I32)</b>
	1 . . . . . generovaný pulz		
	2 . . . . . zpožděné zapnutí		
	3 . . . . . zpožděné vypnutí		
	4 . . . . . zpožděná změna		
<b>pt</b>	Doba časování [s]	⊙1.0	<b>Double (F64)</b>

## Výstup

<b>Q</b>	Výstupní signál časovače	<b>Bool</b>
<b>et</b>	Doba uplynulá od startu časovače [s]	<b>Double (F64)</b>
<b>rt</b>	Zbývající doba [s]	<b>Double (F64)</b>



## Kapitola 9

# TIME – Bloky pro práci s časem

### Obsah

---

<a href="#">DATE – Aktuální datum</a> . . . . .	<b>296</b>
<a href="#">DATETIME – Čtení, nastavování a konverze času</a> . . . . .	<b>297</b>
<a href="#">TC – Řízení časovače</a> . . . . .	<b>299</b>
<a href="#">TIME – Aktuální čas</a> . . . . .	<b>301</b>
<a href="#">WSCH – Týdenní časovač</a> . . . . .	<b>302</b>

---

Knihovna TIME se specializuje na časově založené operace a plánování v systému REXYGEN. Obsahuje bloky jako [DATE](#), [TIME](#) a [DATETIME](#) pro manipulaci s datem a poskytuje základní nástroje pro práci s časovými daty. Knihovna obsahuje [TC](#) pro ovládání vnitřního časovače. Kromě toho se pro plánování používá [WSCH](#), což umožňuje efektivní správu časově závislých úloh. Tato knihovna je zvláště cenná pro systémy vyžadující přesné řízení času a možnosti plánování.

**DATE – Aktuální datum**

Symbol bloku

Licence: **STANDARD****Popis funkce**

Výstupy bloku **DATE** odpovídají datu operačního systému. Pro pokročilé operace s časem a datem použijte blok **DATETIME**.

**Výstupy**

<b>year</b>	Rok	Long (I32)
<b>month</b>	Měsíc	Long (I32)
<b>day</b>	Den	Long (I32)
<b>dow</b>	Den v týdnu, první den je neděle (1)	Long (I32)

**Parametr**

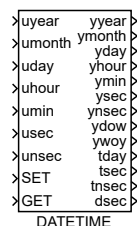
<b>tz</b>	Časové pásmo	⊙1 Long (I32)
	1 ..... lokální čas	
	2 ..... UTC	



## DATE TIME – Čtení, nastavování a konverze času

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **DATE TIME** je určen pro pokročilé operace s časem řídicího systému REXYGEN a operačního systému.

Blok umožňuje synchronizaci hodin operačního systému a řídicího systému REXYGEN. V okamžiku spuštění exekutivy systému REXYGEN jsou hodiny synchronizovány, ale během dlouhodobého provozu se mohou tyto dva údaje rozcházet (např. při přechodu na letní čas). Pokud je potřeba provést opětovnou synchronizaci, hodiny systému REXYGEN se při náběžné hraně (**off**→**on**) na vstupu **SET** aktualizují dle vstupů a parametrů bloku.

Je však důrazně doporučeno neaktualizovat hodiny systému REXYGEN, pokud je řízený stroj či technologie v provozu, neboť by to mohlo vést k nepředvídatelnému chování.

Pokud je potřeba číst nebo konvertovat údaje o čase, je možno příslušnou akci spustit náběžnou hranou (**off**→**on**) na vstupu **GET** a hodnoty přečíst na výstupech bloku. Výstupy začínající na 't' označují celkový počet daných jednotek od 1.1.2000 UTC.

Pokud jsou nastaveny parametry **getper** a **setper** na nenulové hodnoty, je čtení a nastavování hodin prováděno periodicky.

Při menší odchylce hodin systému REXYGEN a operačního systému, než udává parametr **settol**, nejsou hodiny systému REXYGEN nastaveny jednorázově, synchronizace probíhá postupně. Toho je dosaženo zanedbatelnými změnami v časování exekutivy systému REXYGEN, čímž po nějaké době dojde k dosažení synchronizace. Následně je použito standardní časování systému REXYGEN.

Pro jednoduché čtení data a/nebo času použijte bloky [DATE\\_](#) a [TIME](#).

### Vstupy

<b>yyear</b>	Vstup pro nastavení roku	Long (I32)
<b>umonth</b>	Vstup pro nastavení měsíce	Long (I32)
<b>uday</b>	Vstup pro nastavení dne	Long (I32)
<b>uhour</b>	Vstup pro nastavení hodin	Long (I32)
<b>umin</b>	Vstup pro nastavení minut	Long (I32)

<code>usec</code>	Vstup pro nastavení sekund		Long (I32)
<code>unsec</code>	Vstup pro nastavení nanosekund	↓-9.22E+18 ↑9.22E+18	Large (I64)
<code>SET</code>	Nastavení času pomocí náběžné hrany		Bool
<code>GET</code>	Přečtení času pomocí náběžné hrany		Bool

## Výstupy

<code>yyear</code>	Rok		Long (I32)
<code>ymonth</code>	Měsíc		Long (I32)
<code>yday</code>	Den		Long (I32)
<code>yhour</code>	Hodiny		Long (I32)
<code>ymin</code>	Minuty		Long (I32)
<code>ysec</code>	Sekundy		Long (I32)
<code>ynsec</code>	Nanosekundy		Long (I32)
<code>ydw</code>	Den v týdnu		Long (I32)
<code>ywoy</code>	Týden v roce		Long (I32)
<code>tday</code>	Počet dní od začátku epochy		Long (I32)
<code>tsec</code>	Počet sekund od začátku epochy		Long (I32)
<code>tnsec</code>	Počet nanosekund od začátku epochy		Large (I64)
<code>dsec</code>	Počet sekund od půlnoci		Long (I32)

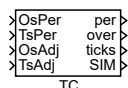
## Parametry

<code>isetmode</code>	Zdroj podle kterého nastavit čas	⊙1	Long (I32)
	1 ..... čas OS		
	2 ..... vstupy bloku		
	3 ..... vstup unsec		
	4 ..... vstup usec		
	5 ..... vstup unsec relativně		
<code>igetmode</code>	Zdroj ze kterého přečíst čas pro nastavení či konverzi	⊙6	Long (I32)
	1 ..... čas OS		
	2 ..... vstupy bloku		
	3 ..... vstup unsec		
	4 ..... vstup usec		
	5 ..... vstup uday		
	6 ..... čas systému REXYGEN		
<code>settol</code>	Tolerance pro nastavení času systému REXYGEN [s]	⊙1.0	Double (F64)
<code>setper</code>	Perioda nastavování času [s] (0=bez opakování)		Double (F64)
<code>getper</code>	Perioda čtení času [s] (0=bez opakování)	⊙0.001	Double (F64)
<code>FDOW</code>	První den v týdnu je neděle		Bool
	off ... týden začíná pondělím		
	on ... týden začíná nedělí		
<code>tz</code>	Časové pásmo	⊙1	Long (I32)
	1 ..... lokální čas		
	2 ..... UTC		

## TC – Řízení časovače

Symbol bloku

Licence: **STANDARD**



### Popis funkce

Blok TC řídí interní časovač systému REXYGEN. Bloku umožňuje modifikovat základní tik algoritmu (která se zadává parametrem `tick` bloku **EXEC**) a to jak skutečnou délku tiků, tak i logickou délku (kolik se kund se přičte do časové značky při každém tiků). Parametr `EXEC:tick` nastavuje logickou i fyzickou periodu a také periodu spouštění bloků (některé bloky potřebují pro diskretizaci algoritmu). Perioda pro bloky není blokem TC ovlivněna.

Skutečná (fyzická) perioda se nastavuje vstupem `OsPer`. Dále je možné nastavit posun tiků o několik sekund. To se provede nastavením posunu na vstup `OsAdj` na jeden tik. Aby nebolo příliš narušeno časování, jsou větší posuny realizovány tak, že je dočasně snížena nebo zvýšena perioda tiků, dokud nedojde k požadovanému posunu. Jak se změní perioda ovlivňuje parametr `OsMax`.

Příklad: předpokládejme periodu tiků 0.1s a `OsMax=0.2`, pak nastavení `OsAdj=1.0` (na jeden tik) dočasně zvýší periodu na 0.12s (tj. o 20% jak určuje parametr `OsMax`), dokud nedojde k celkovému posunu 1sd, tj. na 50 tiků.

Logická perioda se řídí stejně s využitím vstupů/parametrů `TsPer`, `TsAdj`, `TsMax`.

Poznámka 1: Nepřipojené vstupy a vstupy s hodnotou 0 jsou ignorovány (nevyvolávají žádnou akci).

Poznámka 2: Nastavení skutečné periody není na windows platformách momentálně podporováno.

Poznámka 3: Hlavní účel bloku je synchronizovat čas/tiky systému REXYGEN s jiným systémem, takže změny period i offsety se předpokládají malé. Pro simulační a ladící účely je možné změnit periodu výrazně a tím zrychlit pomalu probíhající proces (nebo naopak zpomalit příliš rychle probíhající proces). Je to potřeba to dělat s rozmyslem, protože návaznost na další systémy pomocí driverů se v podstatě přestane fungovat a také se musí při zkrácené periodě stihnout všechny výpočty. Navíc se v tomto případě objevují v logu warningy o chybějících ticích, špatné periodě a pod. Pro tyto účely je lépe použít simulační režim.

### Vstupy

<code>OsPer</code>	Skutečná perioda tiků [s]	Double (F64)
<code>"TsPer</code>	Perioda tiků pro časovou značku[s]	Double (F64)
<code>OsAdj</code>	Posun skutečné polohy tiků [s]	Double (F64)

TsAdj	Posun časové značky tiků [s]	Double (F64)
-------	------------------------------	--------------

### Parametry

OsMax	Maximální relativní změna tiků pro vstup OsAdj ↓0.0 ↑1.0 ⊙0.1	Double (F64)
TsMax	Maximální relativní změna tiků pro vstup TsAdj ↓0.0 ↑1.0 ⊙0.1	Double (F64)

### Výstupy

per	Skutečná délka posledního tiků [s]	Double (F64)
over	Počet ztracených tiků v poslední periodě	Long (I32)
ticks	Počet tiků od startu	Large (I64)
SIM	Časovač v simulačním režimu	Double (F64)

## TIME – Aktuální čas

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Výstupy bloku TIME odpovídají času operačního systému. Pro pokročilé operace s časem a datem použijte blok [DATETIME](#).

### Výstupy

hour	Hodiny	Long (I32)
min	Minuty	Long (I32)
sec	Sekundy	Long (I32)

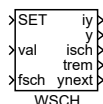
### Parametr

tz	Časové pásmo	⊙1 Long (I32)
	1 ..... lokální čas	
	2 ..... UTC	

## WSCH – Týdenní časovač

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok `WSCH` je určen pro generování týdenních programů, například pro vytápění (den, noc, útlum), větrání (high, low, off), osvětlení, zavlažování apod. Jeho výstupy mohou být využity pro spínání jednotlivých zařízení nebo pro regulaci jejich výkonu.

V běžném provozu jsou v průběhu týdne na výstupech `iy` a `y` generovány hodnoty dle tabulky `wst`, která obsahuje trojice hodnot *den-hodina-hodnota*. Například zápis `[2 6.5 21.5]` znamená, že se v úterý v 6:30 hodin ráno nastaví na výstup `y` hodnota 21.5 a na výstupu `iy` bude hodnota 22 (zaokrouhlení na celé číslo). Jednotlivé trojice hodnot se oddělují středníkem.

Dny jsou číslovány od 1 (pondělí) do 7 (neděle). Vyšší čísla je možno využít pro speciální denní programy, které je možno vynutit pomocí vstupu `fsch` nebo tabulky speciálních dnů `specdays`. Aktuálně platný denní program je indikován výstupem `isch`.

Rovněž je možné dočasně nastavit výstupní hodnotu pomocí vstupů `SET` a `val`. Při náběžné hraně na vstupu `SET` (`off`→`on`) je hodnota `val` zkopírována na výstup `y` a výstup `isch` je přenastaven na hodnotu 0. Ruční hodnota zůstává nastavena, dokud:

- nenastane další přepnutí výstupní hodnoty dle tabulky `wst` nebo
- není přenastavena pomocí další náběžné hrany na vstupu `SET` nebo
- není vynucen jiný denní program pomocí vstupu `fsch`.

Seznam speciálních dní `specdays` lze využít pro vynucení konkrétního denního programu v daný den. Například ve dnech státních svátků můžeme vynutit nedělní režim. Datum se zadává ve formátu `YYYYMMDD`. Zápis `[20160328 7]` tak znamená, že 28. března 2016 se má generovat nedělní program. Jednotlivé dvojice hodnot se oddělují středníkem.

Výstupy `trem` a `ynext` mohou být využity, pokud je potřeba provést nějaké úkony v předstihu ještě před přepnutím výstupních hodnot `iy` a `y`.

Výstup `iy` je určen pro přímé napojení na funkční bloky se vstupy typu Boolean (konverze typu `long` na `bool` se provádí automaticky).

Parametr `nmax` určuje, kolik paměti je alokováno pro pole `wst` a `codespecdays`. Při `nmax = 100` může parametr `wst` obsahovat až 100 trojic *den-hodina-hodnota*. Pro běžné použití není potřeba velikost `nmax` měnit.

## Vstupy

SET	Nastavení výstupů <i>y</i> a <i>iy</i> pomocí náběžné hrany	Bool
val	Hodnota pro dočasné nastavení výstupů <i>y</i> a <i>iy</i>	Double (F64)
fsch	Vynucený denní program	Long (I32)
	0 ..... provoz dle týdenního programu	
	1 ..... pondělí	
	2 ..... úterý	
	..... ..	
	7 ..... neděle	
	8 a více další denní programy dle tabulky <i>wst</i>	

## Výstupy

<i>iy</i>	Celočíselná výstupní hodnota	Long (I32)
<i>y</i>	Výstupní hodnota	Double (F64)
<i>isch</i>	Identifikace denního programu	Long (I32)
<i>trem</i>	Čas zbývající v aktuálním intervalu [s]	Double (F64)
<i>ynext</i>	Výstupní hodnota v dalším intervalu	Double (F64)

## Parametry

<i>tz</i>	Časové pásmo	⊙1	Long (I32)
	1 ..... lokální čas		
	2 ..... UTC		
<i>nmax</i>	Velikost alokovaných polí	↓10 ↑1000000 ⊙100	Long (I32)
<i>wst</i>	Tabulka týdenního programu (seznam trojic <i>den-hodina-hodnota</i> )		Double (F64)
	⊙[1 0.01 18.0; 2 6.0 22.0; 2 18.0 18.0; 3 6.0 22.0; 3 18.0 18.0; 4 6.0 22.0; 4 18.0 18.0		
<i>specdays</i>	Seznam speciálních dní (seznam dvojic <i>datum-denní program</i> )		Long (I32)
	⊙[20150406 1; 20151224 1; 20151225 1; 20151226 1; 20160101 1; 20160328 1; 20170417 1; 20		





# Kapitola 10

## ARC – Archivace dat

### Obsah

---

<b>10.1</b>	<b>Funkce archivačního subsystému</b>	<b>306</b>
<b>10.2</b>	<b>Generování alarmů u a událostí</b>	<b>307</b>
	ALB, ALBI – Alarmy pro logickou hodnotu	307
	ALM, ALMI – Aktivace alarmu	309
	ALN, ALNI – Alarmy pro číselnou hodnotu	310
	ARS – Uložení hodnoty do archivu	313
<b>10.3</b>	<b>Záznam trendů</b>	<b>315</b>
	ACD – Archivní komprese s použitím delta kritéria	315
	TRND – Záznam trendů v reálném čase	317
	TRNDV – Záznam trendů v reálném čase (vektorová forma)	320
<b>10.4</b>	<b>Správa archivů</b>	<b>322</b>
	AFLUSH – Vynucené zapsání archivu	322
	ACLEAR – * Vynucené smazání archivu	323

---

Exekutiva reálného času RexCore se skládá z několika vzájemně spolupracujících subsystémů (subsystém reálného času, diagnostický subsystém, subsystém ovladačů, atd.) Jedním z těchto subsystémů je i *archivační subsystém*.

Archivační subsystém slouží k zaznamenávání a uchovávání historie řídicího systému. Funkce archivačního subsystému je předmětem první podkapitoly.

Ve zbývajících dvou podkapitolách jsou popsány bloky spolupracující s archivačním subsystémem řídicího systému REXYGEN. Podle funkce lze archivační bloky rozdělit do dvou skupin:

- Bloky pro generování alarmů a událostí
- Bloky pro zaznamenávání trendů
- Bloky pro správu archivů

## 10.1 Funkce archivačního subsystému

Archiv slouží v řídicím systému REXYGEN pro ukládání historie událostí, alarmů a trendů vybraných veličin. Řídicí systém může současně obsluhovat až 15 archivů v každé řídicí stanici. Systém rozlišuje tři druhy archivů:

**Archiv v paměti RAM.** Vhodný pro krátkodobé ukládání dat. Výhodou je rychlý přístup k uloženým datům, nevýhodou ztráta dat po restartu systému.

**Archiv v zálohované paměti.** Podobný archivu v paměti RAM. Největší výhodou je zachování uložených dat i při opakovaných restartech systému, navíc přístup k datům zůstává velmi rychlý. Nevýhodou může být někdy jeho nepříliš velká kapacita (závisí na konkrétní hardwarové platformě).

**Archiv v souboru na disku.** Archivy na disku jsou soubory speciálního formátu. Výhodami jsou snadná přenositelnost (kopírování) a zejména velký rozsah dat omezený jen kapacitou disku. Nevýhodou je pomalejší přístup k datům.

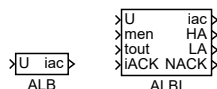
Daná hardwarová platforma nemusí podporovat všechny druhy archivů. Příznaky podporovaných druhů archivů jsou součástí verze řídicího systému cílového zařízení a lze je zjistit v diagnostickém panelu v REXYGEN Studio po kliknutí na jméno cílového zařízení (IP adresu) ve stromu exekutivy. Nachází se na kartě **Target** v levé spodní části.

## 10.2 Generování alarmů u a událostí

### ALB, ALBI – Alarmy pro logickou hodnotu

Symbole bloků

Licence: [STANDARD](#)



#### Popis funkce

Bloky ALB a ALBI jsou určeny pro generování alarmů nebo událostí při změně logické hodnoty přivedené na vstup  $U$ . Výstup  $iac$  indikuje stav alarmu (události). Parametr  $men$  určuje, při jaké změně vstupu  $U$  bude alarm generován. Dále bude popsán blok ALBI. Blok ALB se liší pouze tím, že nemá výstupy HA, LA a  $men$ ,  $iACK$  není jeho vstupem, ale parametrem.

Události a alarmy jsou v systému REXYGEN rozlišeny pomocí parametru  $lv1$ . Pokud je  $1 \leq lv1 \leq 127$ , jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah  $128 \leq lv1 \leq 255$  je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

Poznámka: Vstup (parametr)  $iACK$  se automaticky nuluje po zpracování blokem. Potvrzení alarmu se předpokládá z vizualizace operátorem, tak aby nebylo nutné zapisovat ještě 0 dalším dotazem. Je to podobný princip jako parametr  $BSTATE$  v bloku MP.

#### Vstupy

$U$	Logický vstupní signál	Bool
$men$	Povolení alarmů	Long (I32)
	0 ..... žádný alarm není povolen	
	1 ..... povoleno generování dolního alarmu (LA, sestupná hrana vstupu $U$ )	
	2 ..... povoleno generování horního alarmu (HA, náběžná hrana vstupu $U$ )	
	3 ..... povoleno generování obou alarmů	
$tout$	Doba zpoždění aktivace alarmu [s]	↓0.0 Double (F64)
$iACK$	Potvrzení alarmů (při náběžné hraně)	Byte (U8)
	1 ..... potvrzení dolního alarmu (LA)	
	2 ..... potvrzení horního alarmu (HA)	
	3 ..... potvrzení obou alarmů	

## Výstupy

iac	Kód aktuálního stavu alarmového bloku 0 ..... žádný alarm není aktivní 1 ..... dolní alarm (LA) je aktivní 2 ..... horní alarm (HA) je aktivní 256 ... dolní alarm není potvrzený 512 ... dolní alarm není potvrzený Kladné hodnoty kódů mohou být sčítány, např. hodnota 514 značí, že nepotvrzený horní alarm. Ne všechny kombinace však mají smysl.	Long (I32)
HA	Indikátor horního alarmu	Bool
LA	Indikátor dolního alarmu	Bool
NACK	Indikátor nepotvrzení alarmu	Bool

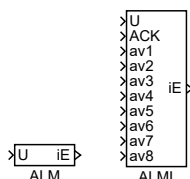
## Parametry

arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok <a href="#">ARC</a> . Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	Word (U16)
id	Identifikační kód alarmu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REXYGEN (tzn. ve všech archivačních i alarmových blocích). Deaktivováno pro id = 0. <span style="float: right;">⊙1</span>	Word (U16)
lvl	Úroveň (závažnost) alarmu, určující, zda jde o skutečný alarm či jen o událost. <span style="float: right;">↓1 ⊙1</span>	Byte (U8)
Desc	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REXYGEN. <span style="float: right;">⊙Alarm Description</span>	String

## ALM, ALMI – Aktivace alarmu

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Blok ALM a ALMI jsou určeny pro generování alarmu. Alarm je aktivní, když vstup  $U = \text{on}$ . Alarm musí být předem definován pomocí bloku [ALARMS](#) a je jednoznačně identifikován pomocí  $id$  parametru. Změna stavu alarmu a jeho potvrzení se také ukládá do archivu (pokud je to nastaveno v konfiguraci bloku [ALARMS](#)). Alarm je možné potvrdit nastavením parametru  $ACK = \text{on}$ .

Poznámka: Systém zobrazuje stav potvrzení, umožňuje potvrdit i trvajícím alarm a může zobrazovat stav potvrzení v HMI a v archivu. Systém REXYGEN dále s potvrzením nijak nepracuje a nic na něm nezávisí. Zda se budou alarmy potvrzovat, závisí na nastavení filtru ve vizualizaci a na návrhu celého systému.

### Vstupy

$U$	Stav alarmu. Alarm je aktivní, pokud $U = \text{on}$	Bool
-----	--	------

### Výstupy

$iE$	Kód chyby	Error
------	-----------	-------

### Parametry

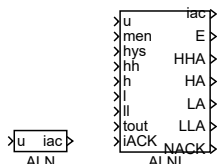
$id$	Identifikační kód alarmu i události v archivu. Tento kód musí být volen jednoznačně v celém projektu s řídicím systémem REXYGEN (tzn. ve všech archivačních i alarmových blocích). Pokud je $id = 0$ alarm se negeneruje. Pokud je $id = -1$ alarm se identifikuje jménem (tj. název bloku musí být stejný jako sloupec Name v definiční tabulce alarmů)	Long (I32)
$ACK$	Potvrzení alarmu nastavením $ACK = \text{on}$ . Blok po zpracování potvrzení sám nastaví $ACK = \text{off}$ .	Bool
$av1..av8$	Hodnota spojená s alarmem. Podrobný popis, jak se hodnota projeví v alarmu (jeho popisu) je v bloku <a href="#">ALARMS</a> .	Double (F64)

↓1.79769e+308 ⊙ -1.79769e+308

## ALN, ALNI – Alarmy pro číselnou hodnotu

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky ALN a ALNI jsou určeny pro generování dvouúrovňových alarmů nebo událostí při překročení (podkročení) číselné hodnoty vstupu *u* některé z horních mezí *h*, *hh* (dolních mezí *l*, *ll*). Výstup *iac* indikuje stav alarmu (události). Vhodnými alarmovými mezemi lze zvolit, při jaké změně vstupu *u* bude alarm generován. Dále bude popsán blok ALNI. Blok ALN se liší pouze tím, že nemá výstupy *HHA*, *HA*, *LA*, *LLA* a místo vstupů *hys*, *hh*, *h*, *l*, *ll*, *iACK* má stejně pojmenované parametry.

Události a alarmy jsou v systému REXYGEN rozlišeny pomocí parametru *lv1*. Pokud je  $1 \leq lv1 \leq 127$ , jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah  $128 \leq lv1 \leq 255$  je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

Poznámka 1: Vstup (parametr) *iACK* se automaticky nuluje po zpracování blokem. Potvrzení alarmu se předpokládá z vizualizace operátorem tak, aby nebylo nutné zapisovat ještě 0 dalším dotazem. Je to podobný princip jako parametr *BSTATE* v bloku *MP*.

Poznámka 2: Do parametru *Desc* lze vkládat formátovací příkazy (hodnoty připojené k alarmu, vícejazyčný text). Jejich podrobný popis je uveden u bloku [ALARMS](#).

### Vstupy

<i>u</i>	Analogový vstupní signál, podle jehož hodnoty se generují alarmy	Double (F64)
<i>hys</i>	Velikost hystereze, určující ukončení alarmu. Význam hystereze i ostatních vstupů je dobře patrný z grafu v příkladu k bloku ALNI. ↓1e-10 ↑1e+10	Double (F64)
<i>hh</i>	Mez pro druhý horní alarm. Musí být větší než <i>h</i> .	Double (F64)
<i>h</i>	Mez pro horní alarm. Musí být větší než <i>l</i> .	Double (F64)
<i>l</i>	Mez pro dolní alarm. Musí být větší než <i>ll</i> .	Double (F64)
<i>ll</i>	Mez pro druhý dolní alarm	Double (F64)
<i>tout</i>	Doba zpoždění aktivace alarmu [s]	↓0.0 Double (F64)

<b>iACK</b>	Potvrzení alarmů	Byte (U8)
	1 . . . . . potvrzení dolního alarmu (LA)	
	2 . . . . . potvrzení horního alarmu (HA)	
	4 . . . . . potvrzení druhého dolního alarmu (LLA)	
	8 . . . . . potvrzení druhého horního alarmu (HHA)	
	Alarm se potvrdí při náběžné hraně. Hodnoty kódů mohou být sčítány, např. hodnota 15 značí potvrzení všech alarmů.	

V případě, že stačí daným blokem generovat jen jednoúrovňové alarmy, stačí nastavit `lv12=0`. Alternativně je možné druhou horní mez `hh` nastavit na větší a druhou dolní mez `ll` na menší hodnotu, než může vstup `u` dosáhnout.

### Výstupy

<b>iac</b>	Kód aktuálního stavu alarmového bloku	Long (I32)
	0 . . . . . žádný alarm není aktivní ani nepotvrzený	
	1 . . . . . dolní alarm (LA) je aktivní	
	2 . . . . . horní alarm (HA) je aktivní	
	4 . . . . . druhý dolní alarm (LLA) je aktivní	
	8 . . . . . druhý horní alarm (HHA) je aktivní	
	256 . . . dolní alarm (LA) není potvrzen	
	512 . . . horní alarm (HA) není potvrzen	
	1024 . . druhý dolní alarm (LLA) není potvrzen	
	2048 . . druhý horní alarm (HHA) není potvrzen	
	-1 . . . . . nesprávné uspořádání alarmových mezí	
	Kladné hodnoty kódů mohou být sčítány, např. hodnota 12 značí, že současně probíhají oba horní alarmy. Ne všechny kombinace však mají smysl.	
<b>E</b>	Příznak chyby uspořádání alarmových mezí	Bool
	off . . . bez chyby      on . . . . nastala chyba	
<b>HHA</b>	Indikátor druhého horního alarmu	Bool
<b>HA</b>	Indikátor (prvního) horního alarmu	Bool
<b>LA</b>	Indikátor (prvního) dolního alarmu	Bool
<b>LLA</b>	Indikátor druhého dolního alarmu	Bool
<b>NACK</b>	Indikátor nepotvrzení alarmu	Bool

### Parametry

<b>ac1s</b>	Třída alarmu (typ proměnné, která bude do archivu ukládána)	Byte (U8)
		⊙8
	1 . . . . . Bool	6 . . . . . DWord (U32)
	2 . . . . . Byte (U8)	7 . . . . . Float (F32)
	3 . . . . . Short (I16)	8 . . . . . Double (F64)
	4 . . . . . Long (I32)	-- . . . .
	5 . . . . . Word (U16)	10 . . . . Large (I64)

arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	Word (U16)
id	Identifikační kód alarmu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REXYGEN (tzn. ve všech archivačních i alarmových blocích). Deaktivováno pro id = 0. <span style="float: right;">⊙1</span>	Word (U16)
lv11	Úroveň (závažnost) prvních horních a dolních alarmů (HA a LA), určující, zda jde o skutečný alarm či jen o událost <span style="float: right;">↓1 ⊙1</span>	Byte (U8)
lv12	Úroveň (závažnost) druhých horních a dolních alarmů (HHA a LLA) <span style="float: right;">↓1 ⊙10</span>	Byte (U8)
Desc	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REXYGEN. <span style="float: right;">⊙Alarm Description</span>	String



## ARS – Uložení hodnoty do archivu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Pokud je `RUN=on`, blok uloží hodnotu na vstupu `u` do archivu. Typ hodnoty na vstupu je určen parametrem `type` a stejný je i typ úložky v archivu. Parametr `subtype` umožňuje zadat typ alarmu, který zapisují alarmové bloky (například `L->H` pro logický alarm, nebo `HiHi` pro číselný alarm). Hodnota parametru může být 0 až 7 a nepoužívá se u polí. Tento parametr se obvykle nevyužívá. Význam ostatních parametrů je stejný jako u ostatních bloků pro zápis do archivu.

Pokud je `type=Reference`, očekává se pole (sloupcový vektor nebo matice). Pokud je to matice, uloží se každý její sloupec jako samostatná úložka do archivu (tj. v jednom tahu s tímto blokem vynikne v archivu tolik položek, kolik má matice sloupců).

Poznámka1: V případě polí, je archivní subsystém omezen na 255 hodnot v jedné úložce. Současně platí omezení na 512 byte dat v jedné úložce, takže pro typ Short se uloží nejvýše 128 hodnot, pro typ Long nejvýše 64 hodnot, atd. Pokud je vstupní pole delší, blok uloží uvedené počty hodnot od začátku pole a nehlásí žádnou chybu.

Poznámka2: V případě stringu je archivní subsystém omezen na 65535 byte (znaků v UTF8 kódování může být méně). Pokud je vstupní text delší, blok uloží prvních 65635 byte od začátku pole a nehlásí žádnou chybu. Některé čtecí funkce mohou mít malý buffer a takto dlouhý text pak nelze vyčíst, doporučuje se proto nepřekračovat 4080 byte (znaků, pokud se používají jen znaky z anglická klávesnice).

Poznámka3: Parametr `id` obvykle slouží k provázání položky v archivu se zdrojovým blokem/signálem (a alarmem v některých případech). Proto se kontroluje jeho unikátnost v rámci celé konfigurace. Blok `ARS` je považován za nízkoúrovňový blok, který запиše událost do archivu bez dalších souvislostí a kontrol. Proto se zde unikátnost parametru `id` nekontroluje. Pokud se například u binárního alarmu začnou v archivu objevovat číselné nebo textové položky, generuje je téměř jistě nějaký blok `ARS` (nebo analogická funkce ve skriptu bloku [REXLANG](#)).

### Vstupy

<code>u</code>	Signál pro uložení do archivu	Any
<code>RUN</code>	Povolení běhu algoritmu	Bool

## Parametry

type	Typ všech použitých bufferů	⊙12	Byte (U8)
	1 ..... Bool		
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	7 ..... Float (F32)		
	8 ..... Double (F64)		
	9 ..... Time		
	10 .... Large (I64)		
	11 .... Error		
	12 .... String		
	13 .... Reference		
arc	Seznam archivů pro zápis alarmů		Word (U16)
id	Identifikátor události v archivu. Unikátnost není v tomto případě kontrolována.	⊙1	Word (U16)
lvl	Úroveň (závažnost) alarmu	⊙1	Word (U16)
Desc	Bližší popis události	⊙Value Description	String

## Výstup

iE	Kód chyby	Error
----	-----------	-------

## 10.3 Záznam trendů

### ACD – Archivní komprese s použitím delta kritéria

Symbol bloku

Licence: [STANDARD](#)

#### Popis funkce

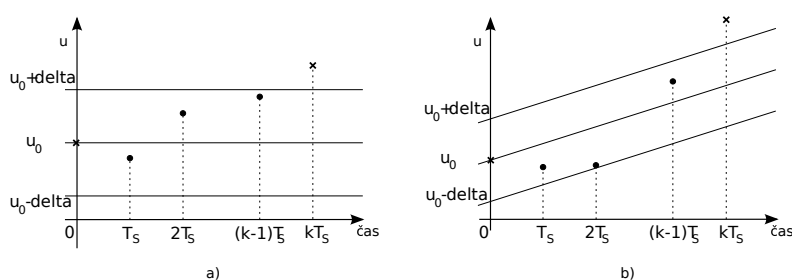
Blok ACD (Archive Compression using Delta criterion) je určen pro ukládání komprimovaných analogových signálů do archivu pomocí archivních událostí.

Základní myšlenkou bloku je archivovat vstupní signál  $u$  jen tehdy, pokud se mění. Doba mezi uložením dvou po sobě následujících hodnot signálu je v intervalu  $\langle t_{\min}, t_{\max} \rangle$  sekund (doby jsou zaokrouhleny na nejbližší násobek periody vzorkování). Pokud se hodnota signálu „hodně“ mění, ukládá se signál jednou za čas  $t_{\min}$ , pokud se hodnota signálu mění „málo“ nebo je konstantní, ukládá se signál jednou za čas  $t_{\max}$ . Po spuštění bloku se vždy uloží první hodnota vstupu  $u$ , označme ji  $u_0$ . Přesná pravidla ukládání dalších vzorků jsou určena vstupem  $\mathit{delta}$  a parametrem  $\mathit{TR}$ .

Je-li  $\mathit{TR}=\mathit{off}$ , testuje se podmínka  $|u - u_0| > \mathit{delta}$ . Pokud je splněna a od minulého uložení uplynul alespoň čas  $t_{\min}$  uloží se tato hodnota  $u$  do archivu a nastaví se  $u_0 = u$ . Je-li podmínka splněna dříve než za čas  $t_{\min}$  od posledního uložení nastaví se chybový výstup  $\mathit{E}$  na 1 a počká se s uložením na první vzorek po uplynutí času  $t_{\min}$ , v tomto okamžiku se nastavuje  $\mathit{E}=0$ . Pak se celý postup opakuje od začátku.

Je-li  $\mathit{TR}=\mathit{on}$ , pracuje blok tak, že ukládá první vzorek, který se odchyluje o více než toleranci  $\mathit{delta}$  od signálu s kompenzovaným trendem. Podmínka na minimální čas ukládání platí obdobně jako v předcházejícím případě.

Chování bloku v obou případech ukazuje následující obrázek: a) pro  $\mathit{TR}=\mathit{off}$ , b) pro  $\mathit{TR}=\mathit{on}$ . Ukládané vzorky jsou označeny symbolem  $\times$ .



#### Vstupy

$u$	Komprimovaně ukládaný signál	Double (F64)
$\mathit{delta}$	Práh pro ukládání signálu do archivu	$\downarrow 0.0 \uparrow 1e+10$ Double (F64)

## Výstupy

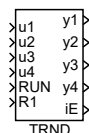
y	Poslední hodnota uložená do archivu	Double (F64)
E	Příznak chyby – nastaven, pokud by měl být vstup u uložen dřív než za čas tmin off ... bez chyby      on .... nastala chyba	Bool

## Parametry

acls	Třída alarmu, určující typ proměnné, která bude do archivu ukládána 1 ..... Bool    5 ..... Word (U16) .... 2 ..... Byte (U8) ..... DWord (U32)... Large (I64) 3 ..... Short (I16) ..... Float (F32) 4 ..... Long (I32) ..... Double (F64)	Byte (U8) ⊙8
arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (details o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	Word (U16)
id	Identifikační kód události v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REXYGEN (tzn. ve všech archivačních blocích). Deaktivováno pro id = 0.    ⊙1	Word (U16)
tmin	Nejkratší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu u do archivu ↓0.001 ↑1000000.0 ⊙1.0	Double (F64)
tmax	Nejdelší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu u do archivu ↓1.0 ↑1000000.0 ⊙1000.0	Double (F64)
TR	Příznak vyhodnocování trendu signálu. Pro TR = off se vyhodnocuje odchylka od poslední uložené hodnoty, v případě TR = on odchylka od trendu posledně uložené hodnoty. off ... vyhodnocuje se odchylka od poslední uložené hodnoty on .... vyhodnocuje se odchylka od trendu posledně uložené hodnoty	Bool
Desc	Řetězec blíže specifikující danou událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REXYGEN. ⊙Value Description	String

## TRND – Záznam trendů v reálném čase

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok **TRND** slouží pro ukládání průběhů až čtyř vstupních signálů **u1** až **u4** do cyklických trendových bufferů v paměti cílového zařízení (target). Výhodou bloku **TRND** je synchronní ukládání dat s během exekutivy reálného času, které umožňuje ukládat do trendu i velmi rychlé signály. Na rozdíl od asynchronního ukládání dat na nadřazeném operátorském počítači (host) nedochází ke ztrátě některých vzorků nebo jejich vícenásobnému uložení. Data lze blokem **TRND** ukládat i pro velmi krátké periody spouštění úloh.

Skutečný počet ukládaných průběhů určuje parametr **n**. V případě, že se trendové buffery s délkou **l** vzorků zaplní, začnou se přepisovat nejstarší vzorky. Do trendových bufferů se mohou ukládat data jednou za **pfac** spuštění bloku (decimace) a ukládaná data mohou být zpracována podle hodnoty parametrů **p1** až **p4**. Další decimace s faktorem **afac** může být použita pro ukládání do archivů.

Pro úsporu paměti na cílovém zařízení může být parametrem **btype** specifikován typ použitých trendových bufferů. Velikost paměti obsazená trendovými buffery je dána vztahem  $s \cdot n \cdot l$ , kde **s** je velikost proměnné daného typu v bytech. Přednastavený typ **Double** zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů  $n = 4$ , délka každého trendu  $l = 1000$ , pak pro typ **Double** je zapotřebí  $8 \cdot 4 \cdot 1000 = 32000$  bytů. V případě, že by byly vstupní signály měřeny z A/D převodníku s rozlišením do 16 bitů, mohly by být ukládány v typu **Word** s velikostí 2 byty na vzorek a velikost potřebné paměti by se zmenšila na jednu čtvrtinu. Velikosti jednotlivých datových typů a jejich rozsahy jsou uvedeny v tabulce 1.1 na straně 24.

Při použití jiného typu pro trendové buffery než je typ **Double** může nastat případ, že se zpracovaná hodnota některého vstupu „nevejde“ do zvoleného typu bufferu a má hodnotu menší (větší) než je nejmenší (největší) zobrazitelné číslo v daném typu. V takovém případě se do bufferu uloží nejmenší (největší) zobrazitelné číslo v daném typu a chyba se binárně zakóduje do chybového výstupu **iE** podle následující tabulky (nepoužité bity jsou vypuštěny):

Chyba	Podkročení rozsahu				Překročení rozsahu			
Vstup	u4	u3	u2	u1	u4	u3	u2	u1
Číslo bitu	11	10	9	8	3	2	1	0
Váha bitu	2048	1024	512	256	8	4	2	1

V případě, že nastane najednou několik chyb, je výsledný chybový kód dán součtem vah jednotlivých chyb. Poznamenejme, že současné překročení a podkročení rozsahu na daném vstupu nemohou nastat zároveň.

Číst, zobrazovat a exportovat průběžně ukládaná data je možné v REXYGEN Studio ve Watch režimu. Po dvojkliku na příslušný TRND blok se otevře nová karta s předponou Trend.

POZOR: nastavení kteréhokoliv z parametrů `arc`, `afac`, `id` na 0 (prázdný) způsobí, že data se nezapíší do archivu a jsou dostupná jen v diagnostických nástrojích.

## Vstupy

<code>u1..u4</code>	Analogové vstupy bloku určené pro zpracování a ukládání do trendu	Double (F64)
<code>RUN</code>	Povolení běhu algoritmu. Data se zpracovávají a ukládají jen pokud je <code>RUN = on</code> .	Bool
<code>R1</code>	Signál pro vymazání obsahu trendového bloku. Data jsou mazána při každém spuštění bloku, je-li <code>R1 = on</code> . Vstup má přednost před vstupem <code>RUN</code> .	Bool

## Výstupy

<code>y1..y4</code>	Analogové výstupy bloku nastavované jednou za <code>pfac</code> spuštění bloku na poslední hodnoty uložené do trendových bufferů	Double (F64)
<code>iE</code>	Kód chyby ukládání do trendových bufferů, viz popis v textu výše	Long (I32)

## Parametry

<code>n</code>	Počet signálů (bufferů) v trendu	↓1 ↑4 ⊙4	Long (I32)
<code>l</code>	Počet vzorků vyhrazený v paměti pro každý buffer trendu	↓0 ↑268435000 ⊙1000	Long (I32)
<code>btype</code>	Typ všech použitých bufferů trendu	⊙8	Long (I32)
	1 ..... Bool    4 ..... Long    7 ..... Float		
	2 ..... Byte    5 ..... Word    8 ..... Double		
	3 ..... Short    6 ..... DWord    10 ..... Large		
<code>pctype<sub>i</sub></code>	Způsob zpracování signálu <code>u<sub>i</sub></code> , $i = 1 \dots 4$ . Zvolený způsob se aplikuje na posledních <code>pfac</code> vzorků a výsledek se uloží do $i$ -tého trendového bufferu	⊙1	Long (I32)
	1 ..... ukládání bez zpracování		
	2 ..... minimum z <code>pfac</code> vzorků		
	3 ..... maximum z <code>pfac</code> vzorků		
	4 ..... součet <code>pfac</code> vzorků		
	5 ..... aritmetický průměr z <code>pfac</code> vzorků		
	6 ..... směrodatná odchylka <code>pfac</code> vzorků		
	7 ..... rozptyl <code>pfac</code> vzorků		

pfac	Násobek periody spouštění bloku pro uložení zpracovaných hodnot do trendových bufferů. Pokud je vstup RUN = on, ukládají se zpracovaná data do trendu s periodou $\text{pfac} \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách. $\downarrow 1 \uparrow 1000000 \odot 1$	Long (I32)
afac	Archivační faktor je číslem udávajícím po kolika uložených vzorcích do trendu se mají ukládané hodnoty navíc uložit do archivů zadaných příznaky arc. Je-li afac = 0, neukládají se trendy do žádného archivu, jinak se ukládají s periodou $\text{afac} \cdot \text{pfac} \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách. $\downarrow 0 \uparrow 1000000$	Long (I32)
arc	Seznam archivů, kam budou ukládána data z trendu. Zadává se ve tvaru např. 1,3..5,8. Data budou uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	Word (U16)
id	Identifikační kód trendu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REXYGEN (tzn. ve všech archivačních blocích). Deaktivováno pro id = 0. $\odot 1$	Word (U16)
Title	Text hlavičky trendu pro zobrazení v diagnostických nástrojích systému REXYGEN, např. ve Watch režimu programu REXYGEN Studio. $\odot$ Trend Title	String
timesrc	Zdroj časových značek. Součástí každého vzorku v trendovém bufferu je časová značka. Pro rychlé nebo krátkodobé trendy, kde nás zajímá přesný čas mezi vzorky odpovídající periodě spouštění úlohy spíše než absolutní čas, vybereme CORETIMER – interní technologický čas systému REXYGEN, který je inkrementován o nominální periodu s každým základním tikem. Pro dlouhodobé trendy, kde nás zajímá spíše absolutní čas sdílený s operačním systémem (a případně synchronizovaný přes NTP), vybereme SYSCLOCK. Ostatní volby jsou určeny pouze pro ladící nebo speciální účely. $\odot 1$ 1 ..... CORETIMER – Čas udržovaný jádrem časovače (v aktuálním ticku) 2 ..... CORETIMER-PRECISE – Čas udržovaný jádrem časovače (při spuštění bloku) 3 ..... SYSCLOCK – Čas udržovaný systémem (v aktuálním ticku) 4 ..... SYSCLOCK-PRECISE – Čas udržovaný systémem (při spuštění bloku) 5 ..... PFC – Čas čítače s vysokým rozlišením (PerFormanceCounter)	Long (I32)
SigNames	Názvy jednotlivých signálů pro zobrazení v diagnostických nástrojích systému REXYGEN, např. ve Watch režimu programu REXYGEN Studio. Zadává se na každou řádku název jednoho signálu. Pokud je parametr nevyplněný, použije se náhradní hodnota (interní identifikace).	String

## TRNDV – Záznam trendů v reálném čase (vektorová forma)

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok TRNDV slouží pro ukládání průběhů vstupních signálů, které jsou bloku předávány ve vektorové podobě. Narozdíl od bloku TRND tedy umožňuje současné ukládání více než 4 signálů, konkrétně je jejich počet určen pomocí parametru *n*. Signály jsou ukládány do cyklických trendových bufferů v paměti cílového zařízení (target). Výhodou bloku TRNDV je synchronní ukládání dat s během exekutivy reálného času, které umožňuje ukládat do trendu i velmi rychlé signály. Na rozdíl od asynchronního ukládání dat na nadřazeném operátorském počítači (host) nedochází ke ztrátě některých vzorků nebo jejich vícenásobnému uložení. Data lze blokem TRNDV ukládat i pro velmi krátké periody spouštění úloh.

V případě, že se trendové buffery s délkou 1 vzorků zaplní, začnou se přepisovat nejstarší vzorky. Do trendových bufferů se mohou ukládat data jednou za *pfac* spuštění bloku (decimace). Další decimace s faktorem *afac* může být použita pro ukládání do archivů.

Pro úsporu paměti na cílovém zařízení může být parametrem *btype* specifikován typ použitých trendových bufferů. Velikost paměti obsazená trendovými buffery je dána vztahem  $s \cdot n \cdot 1$ , kde *s* je velikost proměnné daného typu v bytech. Přednastavený typ *Double* zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů  $n = 4$ , délka každého trendu  $l = 1000$ , pak pro typ *Double* je zapotřebí  $8 \cdot 4 \cdot 1000 = 32000$  bytů. V případě, že by byly vstupní signály měřeny z A/D převodníku s rozlišením do 16 bitů, mohly by být ukládány v typu *Word* s velikostí 2 byty na vzorek a velikost potřebné paměti by se zmenšila na jednu čtvrtinu. Velikosti jednotlivých datových typů a jejich rozsahy jsou uvedeny v tabulce 1.1 na straně 24.

Číst, zobrazovat a exportovat průběžně ukládaná data je možné v REXYGEN Studio ve *Watch* režimu. Po dvojkliku na příslušný TRNDV blok se otevře nová karta s předponou *Trend*.

**POZOR:** nastavení kteréhokoliv z parametrů *arc*, *afac*, *id* na 0 (prázdný) způsobí, že data se nezapisují do archivu a jsou dostupná jen v diagnostických nástrojích.

### Vstupy

<i>uVec</i>	Vektorový signál určený k uložení	<b>Reference</b>
<i>HLD</i>	Pozastavení ukládání dat do cyklických bufferů, při <i>HLD = on</i> se neukládají žádná data	<b>Bool</b>



R1	Signál pro vymazání obsahu trendového bloku. Data jsou mazána při každém spuštění bloku, je-li R1 = on. Vstup má přednost před vstupem HLD.	Bool
----	---	------

## Výstup

iE	Kód chyby i ..... obecná chyba systému REXYGEN	Error
----	---	-------

## Parametry

n	Počet signálů (bufferů) v trendu	↓1 ↑64 ⊙8	Long (I32)
l	Počet vzorků pro každý buffer trendu	↓2 ↑268435000 ⊙1000	Long (I32)
btype	Typ všech použitých bufferů trendu	⊙8	Long (I32)
	1 ..... Bool    4 ..... Long    7 ..... Float 2 ..... Byte    5 ..... Word    8 ..... Double 3 ..... Short   6 ..... DWord 10 ..... Large		
pfac	Násobek periody spouštění bloku pro uložení zpracovaných hodnot do trendových bufferů. Pokud je vstup RUN = on, ukládají se zpracovaná data do trendu s periodou $pfac \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách.	↓1 ↑1000000 ⊙1	Long (I32)
afac	Archivační faktor je číslem udávajícím po kolika uložených vzorcích do trendu se mají ukládané hodnoty navíc uložit do archivů zadaných příznaky arc. Je-li afac = 0, neukládají se trendy do žádného archivu, jinak se ukládají s periodou $afac \cdot pfac \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách.	↓0 ↑1000000	Long (I32)
arc	Seznam archivů, kam budou ukládána data z trendu. Zadává se ve tvaru např. 1,3..5,8. Data budou uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.		Word (U16)
id	Identifikační kód trendu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REXYGEN (tzn. ve všech archivačních blocích). Deaktivováno pro id = 0.	⊙1	Word (U16)
Title	Text hlavičky trendu pro zobrazení v diagnostických nástrojích systému REXYGEN, např. ve Watch režimu programu REXYGEN Studio.	⊙Trend Title	String

<code>timesrc</code>	Zdroj časových značek. Součástí každého vzorku v trendovém bufferu je časová značka. Pro rychlé nebo krátkodobé trendy, kde nás zajímá přesný čas mezi vzorky odpovídající periodě spouštění úlohy spíše než absolutní čas, vybereme <code>CORETIMER</code> – interní technologický čas systému REXYGEN, který je inkrementován o nominální periodu s každým základním tikem. Pro dlouhodobé trendy, kde nás zajímá spíše absolutní čas sdílený s operačním systémem (a případně synchronizovaný přes NTP), vybereme <code>SYSCLOCK</code> . Ostatní volby jsou určeny pouze pro ladicí nebo speciální účely. <span style="float: right;">⊙1</span>	Long (I32)
	<ul style="list-style-type: none"> <li>1 ..... <code>CORETIMER</code> – Čas udržovaný jádrem časovače (v aktuálním ticku)</li> <li>2 ..... <code>CORETIMER-PRECISE</code> – Čas udržovaný jádrem časovače (při spuštění bloku)</li> <li>3 ..... <code>SYSCLOCK</code> – Čas udržovaný systémem (v aktuálním ticku)</li> <li>4 ..... <code>SYSCLOCK-PRECISE</code> – Čas udržovaný systémem (při spuštění bloku)</li> <li>5 ..... <code>PFC</code> – Čas čítače s vysokým rozlišením (PerFormanceCounter)</li> </ul>	
<code>SigNames</code>	Názvy jednotlivých signálů pro zobrazení v diagnostických nástrojích systému REXYGEN, např. ve <code>Watch</code> režimu programu REXYGEN Studio. Zadává se na každou řádku název jednoho signálu. Pokud je parametr nevyplněný, použije se náhradní hodnota (pořadí signálu).	String

## 10.4 Správa archivů

### AFLUSH – Vynucené zapsání archivu

Symbol bloku

Licence: [STANDARD](#)

```

┌FLUSH┐
└AFLUSH┘

```

Popis funkce

Blok `AFLUSH` slouží k vynucenému zapsání dat archivu na disk v situaci, kdy hrozí vypnutí napájení řídicího systému, které by vedlo ke ztrátě archivních dat, která ještě nebyla uložena na disk. V okamžiku náběžné hrany na vstupu `FLUSH` (`off`→`on`) se uloží data na disk bez ohledu na nastavení parametru `period` bloku [ARC](#).

Vstup

<code>FLUSH</code>	Vynucené zapsání archivů	Bool
--------------------	--------------------------	------

## Parametr

arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok <a href="#">ARC</a> . Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	Word (U16)
-----	---	------------

## ACLEAR – \* Vynucené smazání archivu

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

CLEAR	Smazání archivů (náběžná hrana)	Bool
-------	---------------------------------	------

## Parametr

arc	Seznam archivů pro zápis alarmů	Word (U16)
-----	---------------------------------	------------



## Kapitola 11

# STRING – Bloky pro práci s řetězci

### Obsah

---

CNS – * Textová konstanta . . . . .	326
CONCAT – * Spojení stringů (podle vzoru) . . . . .	327
FIND – * Nalezení textu . . . . .	328
ITOS – Konverze celého čísla na text . . . . .	329
LEN – * Délka textu . . . . .	330
MID – * Výřez textu . . . . .	331
PJROCT – Získání číselných hodnot z textu ve formátu JSON . . .	332
PJSOCT – Získání textových hodnot z textu ve formátu JSON . .	334
PJSEXOCT – Získání textových hodnot z textu ve formátu JSON	336
REGEXP – * Regular expression parser . . . . .	337
REPLACE – * Náhrada textu . . . . .	338
RTOS – Konverze čísla na text . . . . .	339
SELSOCT – * Výběr textu z několika vstupů . . . . .	340
STOR – * Koverze textu na číslo . . . . .	341
TRIM – * Odstranění mezer na začátku a konci textu . . . . .	342

---

Knihovna `STRING` je věnována manipulaci a analýze řetězců v systému `REXYGEN`. Zahrnuje bloky jako `CONCAT` pro zřetězení řetězců, `FIND` pro vyhledávání v řetězcích a `REPLACE` pro nahrazování segmentů řetězců. Knihovna nabízí `LEN` a `MID` pro určení délky řetězce a extrakci podřetězců. Pokročilé porovnávání vzorů poskytuje `REGEXP`. Konverzní bloky jako `ITOS`, `STOR` a `RTOS` převádějí celá a reálná čísla na řetězce, zatímco jednoduchý blok `CNS` definuje řetězcovou konstantu. Knihovna navíc obsahuje bloky jako `PJROCT` pro analýzu JSON. Tato kolekce bloků je nezbytná pro manipulaci a zpracování řetězcových dat v různých aplikacích.

**CNS – \* Textová konstanta**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Parametry

<code>scv</code>	Textová hodnota		String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

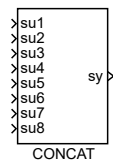
## Výstup

<code>sy</code>	Výstupní textová hodnota		String
-----------------	--------------------------	--	--------

## CONCAT – \* Spojení stringů (podle vzoru)

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String
su3	Vstupní textová hodnota	String
su4	Vstupní textová hodnota	String
su5	Vstupní textová hodnota	String
su6	Vstupní textová hodnota	String
su7	Vstupní textová hodnota	String
su8	Vstupní textová hodnota	String

### Parametry

ptrn	0	⊙%1%2%3%4	String
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

### Výstup

sy	Výstupní textová hodnota	String
----	--------------------------	--------

## FIND – \* Nalezení textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
------	-------------------------------	-----------	------------

Výstup

pos	Poloha hledaného textu	Long (I32)
iE	Kód chyby	Error



## ITOS – Konverze celého čísla na text

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **ITOS** slouží k převedení celého čísla na text. Parametr **len** určuje minimální délku výstupního řetězce. Pokud má číslo menší počet číslic, budou podle parametru **mode** doplněny nuly nebo mezery. Parametr **radix** určuje číselnou soustavu, ve které se má převod provést. Výstupní řetězec neobsahuje žádnou identifikaci použité číselné soustavy (např. předponu 0x u šestnáctkové soustavy).

### Vstup

n	Celočíselný vstupní signál	Long (I32)
---	----------------------------	------------

### Výstup

sy	Výstupní textová hodnota	String
----	--------------------------	--------

### Parametry

<b>len</b>	Minimální délka výstupního řetězce	↓0 ↑30	Long (I32)
<b>mode</b>	Formát výstupního textu	⊙1	Long (I32)
	1 . . . . . zarovnat vpravo, vyplnit mezerami		
	2 . . . . . zarovnat vpravo, vyplnit nulami		
	3 . . . . . zarovnat vlevo, vyplnit mezerami		
<b>radix</b>	Číselná soustava	⊙10	Long (I32)
	2 . . . . . dvojková		
	8 . . . . . osmičková		
	10 . . . . . desítková		
	16 . . . . . šestnáctková		

**LEN – \* Délka textu**

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

**Vstup**

<code>su</code>	Vstupní textová hodnota	<code>String</code>
-----------------	-------------------------	---------------------

**Parametr**

<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520	<code>Long (I32)</code>
-------------------	-------------------------------	-----------	-------------------------

**Výstup**

<code>len</code>	Délka vstupního textu	<code>Long (I32)</code>
------------------	-----------------------	-------------------------

## MID – \* Výřez textu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

su	Vstupní textová hodnota	String
l	Délka výstupního textu	Long (I32)
p	Pozice výstupního textu	Long (I32)

### Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
------	-------------------------------	-----------	------------

### Výstup

sy	Výstupní textová hodnota	String
iE	Kód chyby	Error

## PJROCT – Získání číselných hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Tento blok má stejnou funkci jako [PJSOCT](#), jen výstupy jsou číselné. Pokud v JSON textu požadovaný objekt neexistuje nebo hodnota objektu nejde převést na číslo, nastaví se na výstup hodnota parametru `yerr` a je hlášena chyba.

Blok na vstupu `jtxt` očekává text ve formátu JSON. Na výstupech `y1` až `y7` jsou pak po řadě hodnoty objektů identifikovaných parametry `name1` až `name7`. Pokud je některý z parametrů parametry `name1` až `name7` prázdný, bude prázdný i příslušný výstup a není to považováno za chybu. Blok vstupní string vyhodnocuje jen pokud je `RUN = on`. Na výstupu `iE` je indikována chyba. Mohou nastávat tyto případy:

- 0 – bez chyby
- -1 – některý z parametrů `name1` až `name7` odkazuje na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje
- -103 – textu na vstupu `jtxt` neodpovídá JSON formátu
- -106 – všechny z parametrů `name1` až `name7` odkazují na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje

Příklad: předpokládejme

```
jtxt = "{\"id\": 12345, \"params\":{\"temperature\": 23, \"pressure\": 2.34},
\"description\":\"reactor1\",\"values\":[12, 34.5, 45.0, 30.2]}\"
name1 = \"params.temperature\",
name2 = \"values[0]\",
name3 = \"pressure\",
name4 = \"description\",
```

pak na výstupu `sy1` bude hodnota "23", na výstupu `y2` bude hodnota "12", na výstupu `y3` bude hodnota parametru `yerr` a bude signalizována chyba, na výstupu `y4` bude hodnota parametru `yerr` a bude signalizována chyba

### Vstupy

<code>jtxt</code>	Text v JSON formátu	String
-------------------	---------------------	--------

RUN	Povolení běhu algoritmu	Bool
-----	-------------------------	------

## Parametry

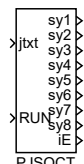
name1..8	Jméno objektu v textu formátu JSON	String
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)
yerr	Náhradní hodnota pro případ chyby	Double (F64)

## Výstupy

y1..8	Výstup bloku	Double (F64)
iE	Kód chyby	Error

## PJSOCT – Získání textových hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Tento blok na vstupu `jtxt` očekává text ve formátu JSON. Na výstupech `sy1` až `sy7` jsou pak po řadě hodnoty objektů identifikovaných parametry `name1` až `name7`. Pokud je některý z parametrů `name1` až `name7` prázdný, bude prázdný i příslušný výstup a není to považováno za chybu. Blok vstupní string vyhodnocuje jen pokud je `RUN = on`. Na výstupu `iE` je indikována chyba. Mohou nastávat tyto případy:

- 0 – bez chyby
- -1 – některý z parametrů `name1` až `name7` odkazuje na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje
- -103 – textu na vstupu `jtxt` neodpovídá JSON formátu
- -106 – všechny z parametrů `name1` až `name7` odkazují na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje

Příklad: předpokládejme

```
jtxt = "{\"id\": 12345, \"params\":{\"temperature\": 23, \"pressure\": 2.34},
\"description\":\"reactor1\",\"values\":[12, 34.5, 45.0, 30.2]}\",
name1 = \"params.temperature\",
name2 = \"values[0]\",
name3 = \"pressure\",
name4 = \"description\",
```

pak na výstupu `sy1` bude hodnota "23", na výstupu `sy2` bude hodnota "12", výstup `sy3` zůstane prázdný a bude signalizována chyba, na výstupu `sy4` bude hodnota "reactor1".

### Vstupy

<code>jtxt</code>	Text v JSON formátu	String
<code>RUN</code>	Povolení běhu algoritmu	Bool

## Parametry

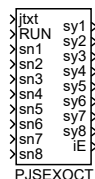
<code>name1..8</code>	Jméno objektu v textu formátu JSON		String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

## Výstupy

<code>sy1..8</code>	Výstupní textová hodnota		String
<code>iE</code>	Kód chyby		Error

## PJSEXOCT – Získání textových hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Tento blok je téměř shodný s blokem [PJSOCT](#). Na vstupu `jtxt` očekává text ve formátu JSON. Na výstupech `sy1` až `sy7` jsou pak po řadě hodnoty objektů identifikovaných parametry `name1` až `name7`. Na rozdíl od bloku [PJSOCT](#) mohou parametry `name1` až `name7` obsahovat zástupný znak `%` + číslo místo kterého se dosadí text ze vstupu `sn` + číslo.

Příklad: předpokládejme `sn1 = "2"`,  
`sn2 = "rpm"`,  
`name1 = "motor[%1].temp"`,  
`name2 = "motor[%1].%2"`,  
pak na výstupu `sy1` bude hodnota objektu `motor[2].temp`, a na výstupu `sy2` bude hodnota objektu `motor[2].rpm`.

## Vstupy

<code>jtxt</code>	Text v JSON formátu	String
<code>RUN</code>	Povolení běhu algoritmu	Bool
<code>sn1..8</code>	Jméno objektu v textu formátu JSON	String

## Parametry

<code>name1..8</code>	Jméno objektu v textu formátu JSON	String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

## Výstupy

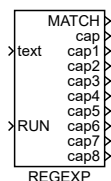
<code>sy1..8</code>	Výstupní textová hodnota	String
<code>iE</code>	Kód chyby	Error



## REGEXP – \* Regular expression parser

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

<code>text</code>	Text k rozpoznání	String
<code>RUN</code>	Povolení běhu algoritmu	Bool

### Parametry

<code>expr</code>	Regulární výraz k rozpoznání	String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65534 Long (I32)
<code>bufmax</code>	Velikost intrní pracovní paměti (0 = automaticky)	↓0 ↑10000000 Long (I32)

### Výstupy

<code>MATCH</code>	Příznak rozpoznání	Bool
<code>cap</code>	Rozpoznaný text odpovídající celému regulárnímu výrazu	String
<code>cap1</code>	Rozpoznaný text odpovídající 1. závorce v regulárním výrazu	String
<code>cap2</code>	Rozpoznaný text odpovídající 2. závorce v regulárním výrazu	String
<code>cap3</code>	Rozpoznaný text odpovídající 3. závorce v regulárním výrazu	String
<code>cap4</code>	Rozpoznaný text odpovídající 4. závorce v regulárním výrazu	String
<code>cap5</code>	Rozpoznaný text odpovídající 5. závorce v regulárním výrazu	String
<code>cap6</code>	Rozpoznaný text odpovídající 6. závorce v regulárním výrazu	String
<code>cap7</code>	Rozpoznaný text odpovídající 7. závorce v regulárním výrazu	String
<code>cap8</code>	Rozpoznaný text odpovídající 8. závorce v regulárním výrazu	String

REPLACE – \* **Náhrada textu**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String
l	Délka původního textu	Long (I32)
p	Pozice původního textu	Long (I32)

## Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
------	-------------------------------	-----------	------------

## Výstup

sy	Výstupní textová hodnota	String
iE	Kód chyby	Error

## RTOS – Konverze čísla na text

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **RTOS** slouží ke konverzi desetinného čísla ze vstupu **u** na výstupní řetězec **su**. Přesnost a formát převodu určují parametry **prec** a **mode**.

### Vstup

<b>u</b>	Analogový vstupní signál	Double (F64)
----------	--------------------------	--------------

### Výstup

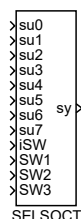
<b>sy</b>	Výstupní textová hodnota	String
-----------	--------------------------	--------

### Parametry

<b>prec</b>	Přesnost (počet cifer)	↓0 ↑20	Long (I32)
<b>mode</b>	Formát výstupního textu	⊙1	Long (I32)
	1 . . . . . nejvhodnější – automaticky se vybere normální nebo exponenciální formát; <b>prec</b> je maximální počet číslic (nevýznamné nuly se nezobrazují)		
	2 . . . . . normální – formát s pevným počtem desetinných míst; <b>prec</b> je počet míst za desetinou tečkou (nevýznamné nuly se zobrazují)		
	3 . . . . . exponenciální – vědecký formát; <b>prec</b> je počet míst za desetinou tečkou (nevýznamné nuly se zobrazují)		

## SELSOCT – \* Výběr textu z několika vstupů

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

su0	Vstupní textová hodnota	String
su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String
su3	Vstupní textová hodnota	String
su4	Vstupní textová hodnota	String
su5	Vstupní textová hodnota	String
su6	Vstupní textová hodnota	String
su7	Vstupní textová hodnota	String
iSW	Selektor aktivního signálu	Long (I32)
SW1	Binární vstup pro výběr	Bool
SW2	Binární vstup pro výběr	Bool
SW3	Binární vstup pro výběr	Bool

## Parametry

BINF	Výběr pomocí binárních vstupů	Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

## Výstup

sy	Zvolený vstupní signál	String
----	------------------------	--------

## STOR – \* Koverze textu na číslo

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstup

su	Vstupní textová hodnota	String
----	-------------------------	--------

### Parametr

yerr	Náhradní hodnota pro případ chyby	Double (F64)
------	-----------------------------------	--------------

### Výstupy

y	Analogový výstupní signál	Double (F64)
E	Příznak chyby	Bool

**TRIM – \* Odstranění mezer na začátku a konci textu**

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

**Vstup**

su	Vstupní textová hodnota	String
----	-------------------------	--------

**Parametr**

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
------	-------------------------------	-----------	------------

**Výstup**

sy	Výstupní textová hodnota	String
----	--------------------------	--------

## Kapitola 12

# PARAM – Bloky pro manipulaci s parametry

### Obsah

---

GETPA – Blok pro vzdálené získání vektorového parametru . . . .	344
GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru . . .	346
GETPS – * Blok pro vzdálené získání parametru typu string . . .	348
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu	349
PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu . . . . .	350
PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu	351
PARS – * Blok s parametrem typu string nastavitelným ze vstupu	353
SETPA – Blok pro vzdálené nastavování vektorového parametru .	354
SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru	356
SETPS – * Blok pro vzdálené nastavování parametru typu string	358
SGSLP – Nastavování, čtení, ukládání a načítání parametrů . . . .	359
SILO – Uložení vstupního signálu, načtení výstupního signálu . .	363
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce .	365

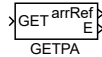
---

Knihovna PARAM je určena pro správu parametrů a zpracování signálů v systému REXYGEN. Obsahuje bloky jako **PARR** a jeho varianty pro definování a úpravu různých typů parametrů. Vyhledávací bloky jako **GETPA** a **GETPS** umožňují extrakci těchto parametrů pro další zpracování. Naopak **SETPA**, **SETPR** a **SETPS** se používají pro dynamické nastavení hodnot parametrů. Knihovna navíc obsahuje **SILO** a **SILOS** pro export a import hodnot ze souboru. Tato knihovna je klíčová pro systémy vyžadující dynamickou manipulaci s parametry a možnosti čtení/ukládání hodnot do souboru.

## GETPA – Blok pro vzdálené získání vektorového parametru

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **GETPA** slouží ke vzdálenému získávání vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem **GETF**. Pro **GETF = off** je na výstup **arrRef** vyveden vzdálený vektorový parametr při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr **GETF** je **on**, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (**off**→**on**) na vstupu **GET**.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn blok **GETPA**. V tomto případě text začíná znakem `'.'`. Příklady hodnot relativních cest: `".CNDR:yp"`, `".Lights.ATMT:touts"`.
- Relativní k tasku – začíná v základní úrovni tasku, do které je umístěn daný blok **GETPA**. V tomto případě text začíná znakem `'%'`. Příklady hodnot cest: `"%CNDR:yp"`, `"%Lights.ATMT:touts"`.
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok **IOTASK**), je v první úrovni hierarchie uveden znak `'&'` následovaný názvem ovladače. Příklady hodnot absolutních cest: `"uloha1.vstupy.ATMT:touts"`, `"&EfaDrv.merani.CNDR:yp"`.

Poznámka 1: Pokud se čte hodnota pole z jiné úlohy, je pro zajištění konzistence hodnot nutné použít tzv. semafor a počkat na dokončení úlohy, ze které se hodnota pole čte. Po celou dobu čekání na dokončení je úloha s blokem **GETPA** pozastavena! Z praktického hlediska to znamená, že blok **GETPA** se musí umístit do úlohy, která trvá dlouho a čítá hodnotu pole z úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **SETPA** v déletrvajícím úloze.

Poznámka 2: Pokud je parametr **GETF = off** a navíc zdrojové pole (určené parametrem **sc**) je ze stejného tasku jako blok **GETPA**, na výstup se dává přímý odkaz na původní pole. To šetří paměť i procesorový čas. V tomto případě jsou parametry **nmax**, **etype** ignorovány.



Poznámka 3: Pokud se použije více bloků **GETPA** pro čtení polí v jiném tasku, není zajištěno, že se všechny pole přečtou v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **GETPA** přečte pole ze stejné nebo dřívější periody druhého tasku než později provedený blok **GETPA**. Pořadí spouštění je vidět v diagnostice programu **REXYGEN Studio**.

Poznámka 4: Vzdálené pole (parametr na který blok odkazuje) musí být primární pole (např. **CNA:acn**, **RTOV:xVec**, **MX\_MAT:ay**). Není podpořeno použít jako vzdálené pole odkaz (např. **CNA:vec**, **RTOV:yVec**, **SUBSYSTEM:Outport**).

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu **REXYGEN Studio**.

## Vstup

**GET** Vstup pro jednorázové přečtení parametru. Pokud je **GETF = on** **Bool**  
data se čtou jen při náběžné hraně na tomto vstupu.

## Výstupy

**arrRef** Odkaz na pole (vektor nebo matice) **Reference**  
**E** Příznak chyby **Bool**

## Parametry

**sc** Jméno vzdáleného parametru **String**  
**GETF** Načtení parametru pouze po vyžádání **Bool**  
    **off** ... režim průběžného čtení parametru  
    **on** .... režim jednorázového přečtení parametru po náběžné  
            hraně na vstupu **GET**  
**nmax** Maximální velikost vektoru (pole)  $\downarrow 10 \odot 256$  **Long (I32)**  
**etype** Type položek pole. Jde o typ v přechodném (stavovém) poli, kam se originální data kopírují. Je provedena konverze, pokud původní data mají jiný typ.  $\odot 8$   
    2 ..... Byte    5 ..... Word    8 ..... Double  
    3 ..... Short   6 ..... DWord 10 .... Large  
    4 ..... Long    7 ..... Float

## GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky `GETPR`, `GETPI` a `GETPB` slouží pro vzdálené získávání parametrů ostatních bloků v modelu. Bloky mají identickou funkci, liší se pouze v typu parametru, který získávají. Blok `GETPR` je pro reálné číslo, `GETPI` pro celé číslo a `GETPB` pro Booleovskou hodnotu.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem `GETF`. Pro `GETF = off` je hodnota výstupu `y` (nebo `k`, `Y`) nastavena na hodnotu vzdáleného parametru při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr `GETF` je `on`, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (`off`→`on`) na vstupu `GET`.

Jméno vzdáleného parametru určuje textový parametr `sc` (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Rovněž je možné přistupovat k jednotlivým prvkům parametrů typu pole (např. parametr `touts` bloku `ATMT`). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy `.ATMT:touts[2]`, číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn daný blok `GETPR` (nebo `GETPI`, `GETPB`). V tomto případě text začíná znakem `'.'`. Příklady hodnot relativních cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.
- Relativní k tasku – začíná v základní úrovni tasku, do které je umístěn daný blok `GETPR` (nebo `GETPI`, `GETPB`). V tomto případě text začíná znakem `'%'`. Příklady hodnot cest: `"%GAIN:k"`, `"%Motor1.Poloha:ycn"`.
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok `IOTASK`), je v první úrovni hierarchie uveden znak `'&'` následovaný názvem ovladače. Příklady hodnot absolutních cest: `"uloha1.vstupy.lin1:u2"`, `"&EfaDrv.mereni.DER1:n"`.

Poznámka 1: Od verze řídicího systému REXYGEN 2.7 došlo ke změně práce s absolutními a relativními cestami. Ve starších verzích měla absolutní cesta prefix `'^'` a relativní cesta neměla prefix žádný. Ke změně bylo přistoupeno z důvodu sjednocení formátu cest s blokem `SGSLP`. Z důvodu maximální možné kompatibility se staršími verzemi je znak `'^'` na začátku řetězců ignorován, je však doporučeno cesty aktualizovat.

Poznámka 2: Pokud se čte hodnota z jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy, ze které se hodnota čte. Po celou dobu čekání na dokončení je úloha s blokem **GETPx** pozastavena! Z praktického hlediska to znamená, že blok **GETPx** se musí umístit do úlohy, která trvá dlouho a číst hodnotu z úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **SETPx** v déletrvajících úloze.

Poznámka 3: Pokud se použije více bloků **GETPx** pro čtení dat v jiném tasku, není zajištěno, že se všechny hodnoty přečtou v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **GETPx** přečte hodnotu ze stejné nebo dřívější periody druhého tasku než později provedený blok **SETPx**. Pořadí spouštění je vidět v diagnostice programu **REXYGEN Studio**. Pokud je důležité přečíst všechny hodnoty ve stejné periodě, musí se přenášet pomocí bloků **Inport** a **Outport** nebo z hodnot vytvořit pole a číst je najednou blokem **GETPA**.

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu **REXYGEN Studio**.

## Vstup

<b>GET</b>	Vstup pro jednorázové přečtení parametru	<b>Bool</b>
------------	--	-------------

## Výstupy

<b>y</b>	Hodnota parametru, výstup bloku <b>GETPR</b>	<b>Double (F64)</b>
<b>k</b>	Hodnota parametru, výstup bloku <b>GETPI</b>	<b>Long (I32)</b>
<b>Y</b>	Hodnota parametru, výstup bloku <b>GETPB</b>	<b>Bool</b>
<b>E</b>	Příznak chyby <b>off</b> ... bez chyby <b>on</b> ... nastala chyba	<b>Bool</b>

## Parametry

<b>sc</b>	Jméno vzdáleného parametru podle výše uvedených pravidel	<b>String</b>
<b>GETF</b>	Zapnutí manuálního čtení vzdáleného parametru <b>off</b> ... režim průběžného čtení parametru <b>on</b> ... režim jednorázového přečtení parametru po náběžné hraně na vstupu <b>GET</b>	<b>Bool</b>

## GETPS – \* Blok pro vzdálené získání parametru typu string

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstup

GET	Vstup pro jednorázové přečtení parametru	Bool
-----	--	------

### Parametry

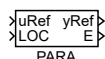
sc	Jméno vzdáleného parametru	String
GETF	Načtení parametru pouze po vyžádání	Bool
	off ... režim průběžného čtení parametru	
	on .... režim jednorázového přečtení parametru	
nmax	Rezervovaná paměť pro řetězec	Long (I32)

### Výstupy

sy	Hodnota parametru	String
E	Příznak chyby	Bool
	off ... bez chyby	
	on .... nastala chyba	

## PARA – Blok s vektorovým parametrem nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok **PARA** je speciální blok, který kromě klasické metody zadávání svých parametrů umožňuje změnu jednoho svého parametru změnou vstupu. Parametr **apar** se může měnit podle vstupu **uRef**.

Logický vstup **LOC** (**LOC**al) určuje, zda bude hodnota vnitřního parametru **apar** čtena ze vstupu **uRef**, v tomto případě je **LOC = off**, nebo hodnota vnitřního parametru nebude na vstupu závislá (**LOC = on**). Pokud je blok v lokálním režimu **LOC = on**, je ve vnitřním parametru **apar** uložena poslední hodnota, která byla na vstupu **uRef** těsně před tím, než byl aktivován lokální režim (**LOC = off → on**).

Výstupní hodnota je shodná s hodnotou parametru **yRef = apar**.

### Vstupy

<b>uRef</b>	Odkaz na pole (vektor nebo matice)	Reference
<b>LOC</b>	Aktivace lokálního režimu	Bool
	<b>off</b> ... parametr je ovládán vstupním signálem	
	<b>on</b> .... lokální režim aktivován	

### Výstup

<b>yRef</b>	Odkaz na pole (vektor nebo matice)	Reference
-------------	------------------------------------	-----------

### Parametry

<b>SETS</b>	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.	Bool
<b>nmax</b>	Maximální (reservovaná) velikost pole <b>apar</b>	↓10 ⊙100 Long (I32)
<b>etype</b>	Typ položek pole <b>apar</b>	⊙8 Long (I32)
	2 ..... Byte    5 ..... Word    8 ..... Double	
	3 ..... Short    6 ..... DWord    10 ..... Large	
	4 ..... Long    7 ..... Float	
<b>par</b>	Interní hodnota parametru	⊙[0.0 1.0 2.0 3.0 4.0 5.0] Double (F64)

## PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Blok je podobný bloku PARI s možností přiřadit texty číselným hodnotám. Odpovídající text je nastaven na výstupu *sy*. Blok má dva režimy a aktivní režim je určen parametrem LIST. Pokud je LIST=off, pak je na výstup *sy* zapsán odpovídající text. Pokud je LIST=on, pak je vstupní číslo považováno za bitové pole, texty jsou definovány pro každý bit a výstup *sy* je složen z textů, které odpovídají nastaveným bitům. Chování pro neznámé hodnoty určuje parametr SATF. Pokud je SATF=off, neznámá hodnota se nastaví na výstup *iy* a výstup *sy* je prázdný text. Při SATF=on jsou neznámé hodnoty ignorovány. Parametr *pupstr* má stejný formát jako v bloku CNE: <number1>: <description1>|<number2>: <description2>|<number3>: <description3> ...

### Vstupy

<i>ip</i>	Hodnota parametru	Long (I32)
<i>LOC</i>	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on .... lokální režim aktivován	

### Parametry

<i>ipar</i>	Interní hodnota parametru	⊙1	Long (I32)
<i>pupstr</i>	Definice seznamu konstant		String
		⊙1: option A 2: option B 3: option C	
<i>NUM</i>	Číslo ve výstupním textu		Bool
<i>LIST</i>	Režim bitového pole		Bool
<i>SATF</i>	Saturace (zda se na výstup nastaví nedefinované hodnoty)		Bool

### Výstupy

<i>iy</i>	Celočíselný výstupní signál	Long (I32)
<i>sy</i>	Výstupní textová hodnota	String

## PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu

### Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky PARR, PARI a PARB jsou speciální bloky, které kromě klasické metody zadávání svých parametrů umožňují změnu jednoho svého parametru změnou vstupu. U bloku PARR změnu parametru `par` změnou vstupu `p`, u PARI změnu `ipar` vstupem `ip` a u PARB změnu `PAR` vstupem `P`.

Logický vstup LOC (LOCal) určuje, zda bude hodnota vnitřního parametru `par` (nebo `ipar`, `PAR`) čtena ze vstupu `p` (nebo `ip`, `P`), v tomto případě je `LOC = off`, nebo hodnota vnitřního parametru nebude na vstupu závislá (`LOC = on`). Pokud je blok v lokálním režimu `LOC = on`, je ve vnitřním parametru `par` (nebo `ipar`, `PAR`) uložena poslední hodnota, která byla na vstupu `p` (nebo `ip`, `P`) těsně před tím, než byl aktivován lokální režim (`LOC = off → on`). Následně je možno tuto hodnotu modifikovat ručně.

Výstupní hodnota je shodná s hodnotou parametru `y = par`, (nebo `k = ipar`, `Y = PAR`). Bloky PARR a PARI mají navíc možnost omezení výstupního signálu `y` a `k` saturačními mezemi `(lolim, hilim)`. Saturační omezení je uvažováno pouze v případě `SATF = on`.

Zvažte také použití bloku [SHLD](#), který lze rovněž použít pro ukládání číselné hodnoty, podobně jako u bloku PARR.

### Vstupy

<code>p</code>	Hodnota parametru (blok PARR)	Double (F64)
<code>ip</code>	Hodnota parametru (blok PARI)	Long (I32)
<code>P</code>	Hodnota parametru (blok PARB)	Bool
<code>LOC</code>	Aktivace lokálního režimu	Bool
	<code>off</code> ... parametr je ovládán vstupním signálem	
	<code>on</code> ... lokální režim aktivován	

### Výstup

<code>y</code>	Analogový výstupní signál bloku PARR	Double (F64)
<code>k</code>	Celočíselný výstupní signál bloku PARI	Long (I32)
<code>Y</code>	Logický výstupní signál bloku PARB	Bool

### Parametry

<code>par</code>	Interní hodnota parametru bloku PARR	⊙1.0 Double (F64)
------------------	--------------------------------------	-------------------

<code>ipar</code>	Interní hodnota parametru bloku PARI	⊙1	Long (I32)
<code>PAR</code>	Interní hodnota parametru bloku PARB	⊙on	Bool
<code>SATF</code>	Omezení výstupu pro bloky PARR a PARI off ... signál není omezen on .... saturační meze jsou aktivní		Bool
<code>hilim</code>	Horní saturační mez (bloky PARR a PARI)	⊙1.0	Double (F64)
<code>lolim</code>	Dolní saturační mez (bloky PARR a PARI)	⊙-1.0	Double (F64)



## PARS – \* Blok s parametrem typu string nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

sp	Hodnota parametru	String
LOC	Aktivace lokálního režimu	Bool

### Parametry

spar	Interní hodnota parametru	String
nmax	Rezervovaná paměť pro řetězec	Long (I32)

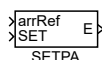
### Výstup

sy	Výstupní řetězec	String
----	------------------	--------

## SETPA – Blok pro vzdálené nastavování vektorového parametru

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SETPA** slouží ke vzdálenému nastavování vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem **SETF**. Pro **SETF = off** je hodnota vzdáleného parametru **sc** nastavena na hodnotu vstupního vektoru **arrRef** při startu a dále pak při každé změně vstupního signálu. Jestliže parametr **SETF** je **on**, pak blok pracuje v režimu jednorázového zápisu vzdáleného parametru, který se nastaví vždy, když nastane náběžná hrana (**off**→**on**) na vstupu **SET**.

Poznámka 1: Pokud se zapisuje hodnota do jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy, do které se hodnota zapisuje. Po celou dobu čekání na dokončení je úloha s blokem **SETPA** pozastavena! Z praktického hlediska to znamená, že blok **SETPA** se musí umístit do úlohy, která trvá dlouho a zapisovat hodnotu do úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **GETPA** v déletrvající úloze.

Poznámka 2: Pokud se použije více bloků **SETPA** pro nastavení hodnot v jiném tasku, není zajištěno, že se všechny hodnoty nastaví v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **SETPA** nastaví hodnotu ve stejné nebo dřívější periodě druhého tasku než později provedený blok **SETPA**. Pořadí spouštění je vidět v diagnostice programu **REXYGEN Studio**.

Poznámka 3: Vzdálené pole (parametr na který blok odkazuje) musí být primární pole (např. **CNA:acn**, **RTOV:xVec**, **MX\_MAT:ay**). Není podpořeno použít jako vzdálené pole odkaz (např. **CNA:vec**, **RTOV:yVec**, **SUBSYSTEM:Output**).

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru **<cesta\_k\_bloku:jmeno\_parametru>**. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn blok **SETPA**. V tomto případě text začíná znakem **'.'**. Příklady hodnot relativních cest: **".CNRD:yp"**, **".Lights.ATMT:touts"**.
- Relativní k tasku – začíná v základní úrovni tasku, do které je umístěn daný blok **SETPA**. V tomto případě text začíná znakem **'%'**. Příklady hodnot cest: **"%CNRD:yp"**, **"%Lights.ATMT:touts"**.

- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok [IOTASK](#)), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.ATMT:touts", "&EfaDrv.mereni.CNDR:yp".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu REXYGEN Studio.

## Vstupy

arrRef	Odkaz na pole (vektor nebo matice)	Reference
SET	Vstup pro jednorázový zápis parametru	Bool

## Výstup

E	Příznak chyby	Bool
---	---------------	------

## Parametry

sc	Jméno vzdáleného parametru	String
SETF	Nastavení parametru pouze na vyžádání off ... režim průběžného nastavování parametru on .... režim jednorázového nastavení parametru po náběžné hraně na vstupu SET	Bool
SETS	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.	Bool

## SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **SETPR**, **SETPI**, **SETPB** a **SETPS** slouží pro vzdálené nastavování parametrů ostatních bloků v modelu. Bloky mají identickou funkci, liší se pouze v typu parametru, který nastavují. Blok **SETPR** je pro reálné číslo, **SETPI** pro celé číslo, **SETPB** pro Booleovskou hodnotu a **SETPS** pro text.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem **SETF**. Pro **SETF = off** je hodnota vzdáleného parametru **sc** nastavena na hodnotu vstupního parametru **p** (nebo **ip**, **P**) při startu a dále pak při každé změně vstupního parametru **p** (nebo **ip**, **P**). V případě **SETF = on** bloky pracují v režimu jednorázového zápisu vzdáleného parametru, který se zapíše při každé náběžné hraně (**off** → **on**) na vstupu **SET**. Po úspěšném zápisu je výstup **y** (nebo **k**, **Y**) nastaven na zapisovanou hodnotu a chybový výstup **E = off**. Při neúspěšném zápisu je **E = on**.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Rovněž je možné přistupovat k jednotlivým prvkům parametru typu pole (např. parametr **tout** bloku **ATMT**). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy `.ATMT:touts[2]`, číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být nastavován, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn daný blok **SETPR** (nebo **SETPI**, **SETPB**). V tomto případě text začíná znakem `'.'`. Příklady hodnot relativních cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.
- Relativní k tasku – začíná v základní úrovni tasku, do které je umístěn daný blok **SETPR** (nebo **SETPI**, **SETPB**, **SETPS**). V tomto případě text začíná znakem `'%'`. Příklady hodnot cest: `"%GAIN:k"`, `"%Motor1.Poloha:ycn"`.
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být nastavován parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok **IOTASK**), je v první úrovni hierarchie uveden znak `'&'` následovaný názvem ovladače. Příklady hodnot absolutních cest: `"uloha1.vstupy.lin1:u2"`, `"&EfaDrv.merени.DER1:n"`.

Poznámka 1: Od verze řídicího systému REXYGEN 2.7 došlo ke změně práce s absolutními a relativními cestami. Ve starších verzích měla absolutní cesta prefix '´ a relativní cesta neměla prefix žádný. Ke změně bylo přistoupeno z důvodu sjednocení formátu cest s blokem **SGSLP**. Z důvodu maximální možné kompatibility se staršími verzemi je znak '´ na začátku řetězců ignorován, je však doporučeno cesty aktualizovat.

Poznámka 2: Pokud se zapisuje hodnota do jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy do které se hodnota zapisuje. Po celou dobu čekání na dokončení je úloha s blokem **SETPx** pozastavena! Z praktického hlediska to znamená, že blok **SETPx** se musí umístit do úlohy, která trvá dlouho a zapisovat hodnotu do úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **GETPx** v déletrvající úloze.

Poznámka 3: Pokud se použije více bloků **SETPx** pro nastavení hodnot v jiném tasku, není zajištěno, že se všechny hodnoty nastaví v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **SETPx** nastaví hodnotu ve stejné nebo dřívější periodě druhého tasku než později provedený blok **SETPx**. Pořadí spouštění je vidět v diagnostice programu REXYGEN Studio. Pokud je důležité zapsat všechny hodnoty ve stejné periodě, musí se přenášet pomocí bloků **Inport** a **Outport** nebo z hodnot vytvořit pole a zapsat je najednou blokem **SETPA**.

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu REXYGEN Studio.

## Vstupy

<b>p</b>	Požadovaná hodnota parametru, vstup bloku <b>SETPR</b>	Double (F64)
<b>ip</b>	Požadovaná hodnota parametru, vstup bloku <b>SETPI</b>	Double (F64)
<b>P</b>	Požadovaná hodnota parametru, vstup bloku <b>SETPB</b>	Double (F64)
<b>SET</b>	Vstup pro jednorázový zápis parametru	Bool

## Výstupy

<b>y</b>	Hodnota parametru, výstup bloku <b>SETPR</b>	Double (F64)
<b>k</b>	Hodnota parametru, výstup bloku <b>SETPI</b>	Long (I32)
<b>Y</b>	Hodnota parametru, výstup bloku <b>SETPB</b>	Bool
<b>E</b>	Příznak chyby	Bool
	<b>off</b> ... bez chyby	
	<b>on</b> ... nastala chyba	

## Parametry

<b>sc</b>	Jméno vzdáleného parametru podle výše uvedených pravidel	String
<b>SETF</b>	Zapnutí manuálního zápisu vzdáleného parametru	Bool
	<b>off</b> ... režim průběžného nastavování parametru	
	<b>on</b> ... režim jednorázového nastavení parametru po náběžné hraně na vstupu <b>SET</b>	

## SETPS – \* Blok pro vzdálené nastavování parametru typu string

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

sp	Požadovaná hodnota parametru	String
SET	Vstup pro jednorázový zápis parametru	Bool

### Parametry

sc	Jméno vzdáleného parametru	String
SETF	Nastavení parametru pouze na vyžádání	Bool
nmax	Rezervovaná paměť pro řetězec	Long (I32)

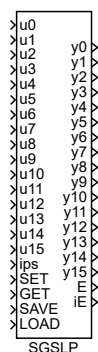
### Výstupy

sy	Hodnota parametru	String
E	Příznak chyby	Bool

## SGSLP – Nastavování, čtení, ukládání a načítání parametrů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok SGSLP (z anglického *Set, Get, Save and Load Parameters*) je speciálním blokem pro správu připojených parametrů jiných bloků v konfiguraci řídicího systému REXYGEN. Blok pracuje i v systému Matlab-Simulink, jeho dosah je však omezen jen na bloky téhož souboru `.mdl`, v němž je vložen.

Blok může pracovat až se šestnácti sadami parametrů, které jsou číslovány od 0 do 15 a volí se vstupem `ips`, aktuální počet sad je určen parametrem `nps`. Je-li vstup `ips` nepřipojen, pracuje blok se sadou `ips = 0`. V každé sadě může být zkonfigurováno až 16 různých parametrů daných řetězcovými parametry `sc0` až `sc15`, takže jeden blok SGSLP může pracovat s maximálně 256 parametry v řídicím systému REXYGEN. Je-li řetězec `sci` prázdný (nezadaný), není žádný parametr specifikován, jinak parametry `sci` mohou používat dvě syntaxe:

1. `<blok>:<param>` – specifikují jeden blok se jménem `blok` s parametrem `param`. V tomto případě je použit tentýž blok a parametr pro všech `nps` sad parametrů.
2. `<blok>:<param><sep>...<blok>:<param>` – v tomto případě je pro každou sadu parametrů `ips` uvažován obecně různý parametr, první dvojice `<blok>:<param>` odpovídá `ips = 0`. Oddělovačem `<sep>` může být buď čárka nebo středník. Specifikovaných dvojic `<blok>:<param>` by mělo být právě `nps`. V případě, že jich je méně, a má se provést některá z operací (viz níže) se sadou, pro niž specifikace bloku a parametru chybí, požadovaná operace se neprovede.

Přestože lze obecně pro každý z indexů  $i$ ,  $i = 0 \dots 15$  volit různý způsob zadání `sci`, doporučuje se pro celý blok volit buď syntaxi 1 nebo 2. První případ (několik hodnot pro stejný parametr) odpovídá např. výrobě `nps` druhů zboží, kde pro každé je nastavena jiná hodnota daného parametru. Druhý případ lze použít např. pro uložení co největšího

počtu uživatelsky definovaných hodnot parametrů na disk (viz operaci **SAVE** níže), kde je vhodné blok **SGSLP** doplnit o logiku přepínání vstupu **ips** (např. pomocí bloku **ATMT** z knihovny **LOGIC**).

Pokud všechny bloky, jejichž parametry mají být nastavovány daným blokem **SGSLP**, leží v hierarchii bloků v nějakém subsystému nebo níže, lze výhodně použít řetězcový parametr **broot**, v němž se uvede jméno tohoto subsystému. Toto jméno se připojuje před každou specifikací **<blok>** v parametrech **sci**. V případě, že je **broot = '.'**, je výsledek stejný, jako by parametr obsahoval cestu k subsystému, do něž je daný blok **SGSLP** vložen (parametr se zadává bez uvozovek, ty jsou použity pouze v tomto textu pro zvýraznění jednotlivého znaku). Je-li hodnota parametru **broot** prázdná, musí každý výskyt **<blok>** v parametrech **sci** specifikovat úplnou cestu k bloku, v níž jsou jednotlivé hierarchické úrovně odděleny tečkami. Například tedy volba **broot = .** a **sc0 = CNR:ycn** zajišťuje propojení na blok **CNR** a jeho parametr **ycn**, který se nachází ve stejném subsystému jako blok **SGSLP**. Případně můžeme ponechat parametr **broot** prázdný a umístit znak **'.'** na začátek řetězce **sc0**. Bližší informace o cestách v systému **REXYGEN** jsou uvedeny u bloků **GETPR** a **SETPR**.

Blok **SGSLP** může při náběžné hraně (**off**→**on**) na některém ze stejnojmenných vstupů provádět následující operace:

**SET** – nastavit parametry dané množiny **ips** na hodnoty přivedené na vstupy **ui**. V případě, že je parametr úspěšně nastaven, je na stejnou hodnotu nastaven i výstup **yi**.

**GET** – získat parametry dané množiny **ips**. V případě, že je parametr úspěšně získán, je jeho hodnota nastavena na výstup **yi**.

**SAVE** – uložit parametry dané množiny **ips** do souboru (tzv. stavový soubor) na cílovém zařízení. Parametry a formát souboru jsou popsány níže.

**LOAD** – načíst parametry dané množiny **ips** ze souboru na cílovém zařízení. Kromě načtení parametrů při náběžné hraně vstupu **LOAD** se parametry sady **ips0** načtou při inicializaci bloku v případě, že je hodnota parametru **ips0** v rozsahu od 0 do **nps – 1**. Parametry a formát souboru jsou popsány níže.

Operace **LOAD** a **SAVE** pracují se souborem na cílovém zařízení, jehož jméno je uvedeno v parametru **fname**. Práce s parametrem **fname** se řídí následujícími pravidly:

- Pokud jméno souboru neobsahuje příponu, přidává se automaticky přípona **.rxs** (ReX Status file).
- Při ukládání bude vytvářen záložní soubor se stejným jménem, avšak s příponou modifikovanou přidáním znaku **'~'** ihned za znak **'.'**, např. pokud jméno souboru neobsahuje příponu, je přípona záložního souboru **~rxs**.
- Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení. Data se typicky ukládají na pevný disk nebo flash disk nebo jiné médium, které po vypnutí a opětovném zapnutí zachovává soubory.



Data jsou příkazem **SAVE** ukládána do textového souboru, ze kterého jsou příkazem **LOAD** načítána zpět do bloku **SGSLP**. Pro každý parametr  $sci$ ,  $i = 0, \dots, m$ , kde  $m < 16$  je maximální číslo, pro něž je parametr  $scm$  neprázdný řetězec, obsahuje soubor dva řádky ve tvaru:

```
"<blok>:<param>", ..., "<blok>:<param>"
<hodnota>, ..., <hodnota>
```

Jednotlivé položky "**<blok>:<param>**" jsou mezi sebou odděleny čárkami a jejich počet odpovídá parametru **nps**, obdobně to platí i o položkách **<hodnota>** obsahujících hodnotu parametru, jehož jméno je uvedeno ve stejné pozici v předchozím řádku. Poznamenejme, že pro  $nps > 1$  má první z těchto dvou řádků vždy právě uvedený tvar (dvojice "**<blok>:<param>**" se opakuje **nps**-krát) a to i v případě, že parametr  $sci$  obsahuje jedinou dvojici **<blok>:<param>** (viz 1. syntaxe výše). Tato skutečnost umožňuje přecházet mezi oběma syntaxemi parametrů  $sci$ , aniž by musel být soubor upravován.

Při ukládání malého počtu hodnot můžete rovněž využít blok **SIL0**.

## Vstupy

$ui$	$i$ -tý analogový vstupní signál, $i = 0, \dots, 15$	Double (F64)
$ips$	Číslo sady parametrů (číslováno od 0)	Long (I32)
<b>SET</b>	Přečtení vstupů $ui$ a nastavení parametrů sady $ips$ na jejich hodnoty	Bool
<b>GET</b>	Přečtení parametrů sady $ips$ a nastavení výstupů $yi$ na jejich hodnoty	Bool
<b>SAVE</b>	Uložení parametrů sady $ips$ do souboru na disk cílového zařízení	Bool
<b>LOAD</b>	Načtení parametrů sady $ips$ ze souboru na disku cílového zařízení	Bool

## Výstupy

$yi$	$i$ -tý analogový výstupní signál, $i = 0, \dots, 15$	Double (F64)
<b>E</b>	Příznak chyby <b>off</b> ... bez chyby <b>on</b> ... nastala chyba, viz výstup <b>iE</b>	Bool

iE	Chybový nebo varovný výstup poslední operace	Long (I32)
	0 ..... nenastala žádná chyba ani varování	
	1 ..... fatální chyba volání systému Matlab (jen pro Simulink), blok dále není spouštěn	
	2 ..... chyba otvírání souboru pro čtení (příkaz LOAD)	
	3 ..... chyba otvírání souboru pro zápis (příkaz SAVE)	
	4 ..... nesprávný formát souboru	
	5 ..... dané číslo <i>ips</i> nebylo v souboru nalezeno	
	6 ..... jména parametru v souboru a v konfiguraci bloku si neodpovídají	
	7 ..... byl nalezen neočekávaný konec souboru	
	8 ..... chyba zápisu do souboru (plný disk?)	
	9 ..... chyba syntaxe parametru (chybí znak ':' )	
	10 .... připojení parametru je tvořeno jen bílými znaky	
	11 .... nelze vytvořit záložní soubor	
	12 .... hodnotu parametru nelze získat operací GET (neexistující parametr?)	
	13 .... hodnotu parametru nelze nastavit operací SET (neexistující parametr?)	
	14 .... překročení času při získávání/nastavování parametru (timeout)	
	15 .... připojenou hodnotu (parametr) není dovoleno zapisovat	
	16 .... číslo sady <i>ips</i> je mimo přípustný rozsah	

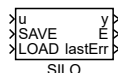
## Parametry

nps	Počet sad parametrů	↓1 ↑16 ⊙1	Long (I32)
ips0	Číslo sady parametrů, která se načte ze souboru při inicializaci bloku. Je-li $ips0 < 0$ nebo $ips0 \geq nps$ , načte se při inicializaci žádná sada	↓-1 ↑15	Long (I32)
iprec	Počet platných číslic pro zápis hodnoty typu <i>double</i> do souboru	↓2 ↑15 ⊙12	Long (I32)
icolw	Šířka sloupce v souboru. Je-li skutečná šířka menší, je doplněna zprava mezerami. Pokud je $icolw < iprec$ , nebudou žádné mezery přidávány.	↓0 ↑22	Long (I32)
fname	Jméno souboru, do kterého se ukládají parametry příkazem SAVE a ze kterého se načítají příkazem LOAD	⊙status	String
broot	Cesta k subsystému, přidávaná na začátek specifikace bloků v parametrech <i>sci</i> , viz popis v textu výše	⊙.	String
sci	Řetězce specifikující připojení vstupů <i>ui</i> a výstupů <i>yi</i> , $i = 0, \dots, 15$ k požadovaným parametrům, viz popis v textu výše		String

## SILO – Uložení vstupního signálu, načtení výstupního signálu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SILO** je určen pro export nebo import jednoho signálu (hodnoty) do nebo ze souboru. Hodnota je uložena při náběžné hraně (**off**→**on**) na vstupu **SAVE** a po úspěšném uložení je nastavena také na výstup **y**. Načtení hodnoty probíhá při startu a při náběžné hraně (**off**→**on**) na vstupu **LOAD**.

Chyba diskové operace je indikována na výstupech **E** a **lastErr**. Příznak **E** je shozen při sestupné hraně na vstupu **SAVE** nebo **LOAD**, zatímco výstup **lastErr** drží hodnotu až do další operace. Pokud chyba nastala při operaci **LOAD**, je na výstup **y** nastavena náhradní hodnota **yerr**.

Alternativně lze zapnout průběžné ukládání nebo čtení pomocí příslušného parametru (**CSF**, **CLF**). Diskové operace pak probíhají kontinuálně, ovšem pouze když je příslušný vstupní signál nastaven na **on**. Pozor však na to, že zápis/čtení pak probíhá při každém spuštění bloku, což může mít za následek nadměrné zatížení úložného zařízení, proto je potřeba použití tohoto režimu vždy důkladně zvážit.

Parametr **fname** určuje umístění souboru. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení.

Pro pokročilé a hromadné operace je určen blok [SGSLP](#).

### Vstupy

<b>u</b>	Vstupní signál	Double (F64)
<b>SAVE</b>	Uložení vstupní hodnoty do souboru	Bool
<b>LOAD</b>	Načtení hodnoty výstupu ze souboru	Bool

### Parametry

<b>fname</b>	Jméno souboru pro ukládání/načítání parametrů	String
<b>CSF</b>	Příznak pro průběžné ukládání	Bool
<b>CLF</b>	Příznak pro průběžné načítání	Bool
<b>yerr</b>	Náhradní hodnota pro případ chyby	Double (F64)

### Výstupy

<b>y</b>	Výstupní signál	Double (F64)
<b>E</b>	Příznak chyby	Bool

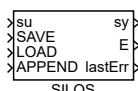
`lastErr`      Výsledek poslední operace

`Long (I32)`

## SILOS – Uložení vstupního řetězce, načtení výstupního řetězce

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SILOS** je určen pro export nebo import jednoho řetězce do nebo ze souboru. Řetězec je uložen při náběžné hraně (**off**→**on**) na vstupu **SAVE** a po úspěšném uložení je nastavena také na výstup **sy**. Načtení hodnoty probíhá při startu a při náběžné hraně (**off**→**on**) na vstupu **LOAD**.

Pokud je na vstup **APPEND** přivedena hodnota **on**, řetězec ze vstupu je při ukládání přidán na konec souboru. Tento režim se hodí pro logování událostí do textových souborů. Na načítání ze souboru nemá tento vstup žádný vliv.

Pomocí parametru **LLO** lze zvolit, zda se má načítat celý soubor (**off**) nebo pouze jeho poslední řádek (**on**).

Chyba diskové operace je indikována na výstupech **E** a **lastErr**. Příznak **E** je shozen při sestupné hraně na vstupu **SAVE** nebo **LOAD**, zatímco výstup **lastErr** drží hodnotu až do další operace.

Alternativně lze zapnout průběžné ukládání nebo čtení pomocí příslušného parametru (**CSF**, **CLF**). Diskové operace pak probíhají kontinuálně, ovšem pouze když je příslušný vstupní signál nastaven na **on**. Pozor však na to, že zápis/čtení pak probíhá při každém spuštění bloku, což může mít za následek nadměrné zatížení úložného zařízení, proto je potřeba použití tohoto režimu vždy důkladně zvážit.

Parametr **fname** určuje umístění souboru. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení.

### Vstupy

<b>su</b>	Vstupní řetězec	⊙0	String
<b>SAVE</b>	Uložení vstupního řetězce do souboru		Bool
<b>LOAD</b>	Načtení řetězce ze souboru		Bool
<b>APPEND</b>	Načtení řetězce ze souboru		Bool

### Výstupy

<b>sy</b>	Výstupní řetězec		String
-----------	------------------	--	--------

E	Příznak chyby off ... bez chyby on .... nastala chyba	Bool
lastErr	Výsledek poslední operace	Long (I32)

### Parametry

fname	Jméno souboru pro ukládání/načítání parametrů	String
CSF	Průběžné ukládání	Bool
CLF	Průběžné načítání	Bool
LL0	Načítání jen poslední řádky	Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

## Kapitola 13

# MODEL – Simulace dynamických systémů

### Obsah

---

CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním . . . . .	369
CSSM – Stavový model spojitého lineárního systému . . . . .	372
DDELSSM – Stavový model diskretního lineárního systému s dopravním zpožděním . . . . .	374
DFIR – * Filtr s konečnou impulzní odezvou . . . . .	376
DSSM – Stavový model diskretního lineárního systému . . . . .	377
EKF – Rozšířený (nelineární) Kalmanův filtr . . . . .	379
FOPDT – Model systému 1. řádu s dopravním zpožděním . . . . .	382
IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry . . . . .	383
IPEN2 <sub>pu</sub> , IPEN3 <sub>pu</sub> – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry . . . . .	386
MDL – Model procesu . . . . .	388
MDLI – Model procesu s proměnnými parametry . . . . .	389
MVD – Motorizovaný pohon ventilu . . . . .	390
NSSM – Nelineární stavový model . . . . .	391
SOPDT – Model systému 2. řádu s dopravním zpožděním . . . . .	394

---

Knihovna MODEL je zaměřena na modelování a simulaci systému. Zahrnuje bloky jako **CSSM** a **DSSM** pro spojité a diskretní modely stavového prostoru a **DFIR** pro digitální filtry s konečnou impulzní odezvou. Knihovna nabízí **EKF** pro implementace rozšířeného Kalmanova filtru a **FOPDT**, **SOPDT** pro modely časového zpoždění procesu prvního a druhého řádu. Navíc poskytuje **FMUCS** a **FMUINFO** pro propojení s funkčními mock-upovými jednotkami a **MDL**, **MDLI** pro rozhraní generických modelů. Pokročilé funkce jsou pokryty

bloky jako **CDELSSM**, **DDELSSM** pro stavové modely kontinuálních a diskrétních systémů se zpožděním a **MVD** pro modelové proměnné zpoždění, které splňují širokou škálu požadavků na modelování. v systému REXYGEN.



## CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním

Symbol bloku

Licence: [ADVANCED](#)

```

xR1   iE >
xHLD  y1 >
xu1   y2 >
xu2   y3 >
xu3   y4 >
xu4   y5 >
xu5   y6 >
xu6   y7 >
xu7   y8 >
xu8   y9 >
xu9   y10 >
xu10  y11 >
xu11  y12 >
xu12  y13 >
xu13  y14 >
xu14  y15 >
xu15  y16 >
xu16  >
CDELSSM

```

### Popis funkce

Funkční blok CDELSSM (Continuous State Space Model with time DELay) simuluje chování lineárního spojitého systému s dopravním zpožděním *del* ve stavové reprezentaci

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t - del), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde  $x(t) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  je počáteční hodnota vektoru stavu,  $u(t) \in \mathbb{R}^m$  je vektor vstupu,  $y(t) \in \mathbb{R}^p$  je vektor výstupu. Matice  $A_c \in \mathbb{R}^{n \times n}$  určuje dynamiku systému, matice  $B_c \in \mathbb{R}^{n \times m}$  určuje působení vstupu na stav systému, matice  $C_c \in \mathbb{R}^{p \times n}$  určuje působení stavu na výstup systému a matice  $D_c \in \mathbb{R}^{p \times m}$  určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor  $x_0$  je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskrétního (diskretizovaného) stavového modelu

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_{d1} u((k-d)T) + B_{d2} u((k-d+1)T), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde  $k \in \{1, 2, \dots\}$  je krok simulace,  $T$  je perioda spouštění bloku v [s] a  $d$  je zpoždění v krocích simulace tak, aby  $(d-1)T < del \leq d.T$ . Perioda  $T$  se v bloku nezadává, je určena automaticky jako perioda úlohy ([TASK](#), [QTASK](#) nebo [IOTASK](#)), do níž je blok zařazen.

Pokud se vstup  $u(t)$  mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj.  $u(t) = u(kT)$  pro  $t \in [kT, (k+1)T)$ , pak matice  $A_d$ ,  $B_{d1}$  a  $B_{d2}$  jsou určeny vztahy

$$\begin{aligned} A_d &= e^{A_c T} \\ B_{d1} &= e^{A_c(T-\Delta)} \int_0^\Delta e^{A_c \tau} B_c d\tau \\ B_{d2} &= \int_0^{T-\Delta} e^{A_c \tau} B_c d\tau, \end{aligned}$$

kde  $\Delta = del - (d-1)T$ .

Výpočet diskretních matic  $A_d$ ,  $B_{d1}$  a  $B_{d2}$  je založen na metodě popsané v [4], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše.

### Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor $\mathbf{x}$ nastaven na počáteční hodnotu $\mathbf{x}0$ . Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	Bool
HLD	Zmrazení simulace po dobu, kdy je HLD=on.	Bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních $m$ vstupů, kde $m$ je počet sloupců matice Bc.	Double (F64)

### Výstupy

iE	Kód chyby bloku 0 .. . . . vše v pořádku, blok simuluje správně -213 .. nekompatibilita rozměrů matic stavového modelu -510 .. úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízká singulární matici) xxx ... chybový kód xxx systému REXYGEN, více viz přílohu C	Error
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice Cc.	Double (F64)

### Parametry

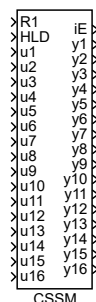
UD	Příznak použití matice Dc. Pokud je UD=off, matice Dc se při simulaci nepoužívá (chová se jako by byla nulová).	Bool
del	Dopravní zpoždění modelu [s].	↓0.0 Double (F64)
is	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému.	↓0 ↑4 ⊙2 Long (I32)
eps	Požadovaná přesnost Padéovy aproximace.	↓0.0 ↑1.0 ⊙1e-15 Double (F64)
Ac	Matice (typu [n,n]) dynamiky spojitého lineárního systému.	Double (F64)

<b>Bc</b>	Vstupní matice (typu [n,m]) spojitého lineárního systému.	Double (F64)
<b>Cc</b>	Výstupní matice (typu [p,n]) spojitého lineárního systému.	Double (F64)
<b>Dc</b>	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr UD=on. Je-li UD=off, rozměry matice Dc se nekontrolují.	Double (F64)
<b>x0</b>	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.	Double (F64)

## CSSM – Stavový model spojitého lineárního systému

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok **CSSM** (Continuous State Space Model) simuluje chování lineárního spojitého systému ve stavové reprezentaci

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde  $x(t) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  je počáteční hodnota vektoru stavu,  $u(t) \in \mathbb{R}^m$  je vektor vstupu,  $y(t) \in \mathbb{R}^p$  je vektor výstupu. Matice  $A_c \in \mathbb{R}^{n \times n}$  určuje dynamiku systému, matice  $B_c \in \mathbb{R}^{n \times m}$  určuje působení vstupu na stav systému, matice  $C_c \in \mathbb{R}^{p \times n}$  určuje působení stavu na výstup systému a matice  $D_c \in \mathbb{R}^{p \times m}$  určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor  $x_0$  je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskretního (diskretizovaného) stavového modelu

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_d u(kT), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde  $k \in \{1, 2, \dots\}$  je krok simulace,  $T$  je perioda spouštění bloku v [s]. Perioda  $T$  se v bloku nezadává, je určena automaticky jako perioda úlohy (**TASK**, **QTASK** nebo **IOTASK**), do níž je blok zařazen.

Pokud se vstup  $u(t)$  mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj.  $u(t) = u(kT)$  pro  $t \in$

$[kT, (k+1)T)$ , pak matice  $A_d$  a  $B_d$  jsou určeny vztahy

$$A_d = e^{A_c T}$$

$$B_d = \int_0^T e^{A_c \tau} B_c d\tau$$

Výpočet diskretních matic  $A_d$  a  $B_d$  je založen na metodě popsané v [4], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše.

## Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor $x$ nastaven na počáteční hodnotu $x_0$ . Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	Bool
HLD	Zmrazení simulace po dobu, kdy je HLD=on.	Bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních $m$ vstupů, kde $m$ je počet sloupců matice $B_c$ .	Double (F64)

## Výstupy

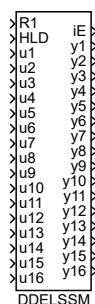
iE	Kód chyby bloku 0 ..... vše v pořádku, blok simuluje správně -213 .. nekompatibilita rozměrů matic stavového modelu -510 .. úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízká singulární matici) xxx ... chybový kód xxx systému REXYGEN, více viz přílohu C	Error
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice $C_c$ .	Double (F64)

## Parametry

UD	Příznak použití matice $D_c$ . Pokud je UD=off, matice $D_c$ se při simulaci nepoužívá (chová se jako by byla nulová).	Bool
is	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému. ↓0 ↑4 ⊙2	Long (I32)
eps	Požadovaná přesnost Padéovy aproximace. ↓0.0 ↑1.0 ⊙1e-15	Double (F64)
Ac	Matice (typu [n,n]) dynamiky spojitého lineárního systému.	Double (F64)
Bc	Vstupní matice (typu [n,m]) spojitého lineárního systému.	Double (F64)
Cc	Výstupní matice (typu [p,n]) spojitého lineárního systému.	Double (F64)
Dc	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr UD=on. Je-li UD=off, rozměry matice $D_c$ se nekontrolují.	Double (F64)
x0	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.	Double (F64)

## DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním

Symbol bloku

Licence: [ADVANCED](#)

### Popis funkce

Funkční blok DDELSSM (Discrete State Space Model with time DELay) simuluje chování lineárního diskrétního systému s dopravním zpožděním  $del$  ve stavové reprezentaci

$$\begin{aligned}x(k+1) &= A_d x(k) + B_d u(k-d), \quad x(0) = x_0 \\y(k) &= C_d x(k) + D_d u(k),\end{aligned}$$

kde  $k$  je krok simulace,  $x(k) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  je počáteční hodnota vektoru stavu,  $u(k) \in \mathbb{R}^m$  je vektor vstupu,  $y(k) \in \mathbb{R}^p$  je vektor výstupu. Matice  $A_d \in \mathbb{R}^{n \times n}$  určuje dynamiku systému, matice  $B_d \in \mathbb{R}^{n \times m}$  určuje působení vstupu na stav systému, matice  $C_d \in \mathbb{R}^{p \times n}$  určuje působení stavu na výstup systému a matice  $D_d \in \mathbb{R}^{p \times m}$  určuje přímé působení vstupu na výstup systému. Počet kroků zpoždění  $d$  je největší celé číslo takové, že  $d.T \leq del$ , kde  $T$  je perioda spouštění bloku.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor  $x_0$  je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

### Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor $x$ nastaven na počáteční hodnotu $x_0$ . Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	Bool
HLD	Zmrazení simulace po dobu, kdy je HLD=on.	Bool

**u1..u16** Vstupy simulovaného systému. Pro danou simulaci se používá **Double (F64)** prvních  $m$  vstupů, kde  $m$  je počet sloupců matice **Bd**.

## Výstupy

**iE** Kód chyby bloku **Error**  
 0 ..... vše v pořádku, blok simuluje správně  
 -213 .. nekompatibilita rozměrů matic stavového modelu  
 xxx ... chybový kód xxx systému REXYGEN, více viz přílohu **C**

**y1..y16** Výstupy simulovaného systému. Pro danou simulaci se používá **Double (F64)** prvních  $p$  výstupů, kde  $p$  je počet řádků matice **Cd**.

## Parametry

**UD** Příznak použití matice **Dd**. Pokud je **UD=off**, matice **Dd** se při simulaci nepoužívá (chová se jako by byla nulová). **Bool**

**del** Dopravní zpoždění modelu [s].  $\downarrow 0.0$  **Double (F64)**

**Ad** Matice (typu [n,n]) dynamiky diskrétního lineárního systému. **Double (F64)**

**Bd** Vstupní matice (typu [n,m]) diskrétního lineárního systému. **Double (F64)**

**Cd** Výstupní matice (typu [p,n]) diskrétního lineárního systému. **Double (F64)**

**Dd** Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr **UD=on**. Je-li **UD=off**, rozměry matice **Dd** se nekontrolují. **Double (F64)**

**x0** Počáteční hodnota vektoru stavu (typu [n]) diskrétního lineárního systému. **Double (F64)**

DFIR – \* **Filtr s konečnou impulzní odezvou**

Symbol bloku

Licence: [ADVANCED](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

RST	Reset bloku	Bool
HLD	Pozastavení	Bool
u0	počáteční hodnota po resetu modelu	Double (F64)

## Parametr

nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊖100	Long (I32)
hk	impulzní funkce (diskretizovaná)	⊖ [0.6 0.3 0.1]	Double (F64)

## Výstup

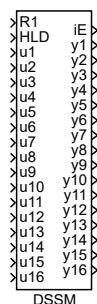
y	Analogový výstupní signál	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool



## DSSM – Stavový model diskrétního lineárního systému

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok **DSSM** (Discrete State Space Model) simuluje chování lineárního diskrétního systému ve stavové reprezentaci

$$\begin{aligned}x(k+1) &= A_d x(k) + B_d u(k), \quad x(0) = x_0 \\y(k) &= C_d x(k) + D_d u(k),\end{aligned}$$

kde  $k$  je krok simulace,  $x(k) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  je počáteční hodnota vektoru stavu,  $u(k) \in \mathbb{R}^m$  je vektor vstupu,  $y(k) \in \mathbb{R}^p$  je vektor výstupu. Matice  $A_d \in \mathbb{R}^{n \times n}$  určuje dynamiku systému, matice  $B_d \in \mathbb{R}^{n \times m}$  určuje působení vstupu na stav systému, matice  $C_d \in \mathbb{R}^{p \times n}$  určuje působení stavu na výstup systému a matice  $D_d \in \mathbb{R}^{p \times m}$  určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor  $x_0$  je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

### Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor $x$ nastaven na počáteční hodnotu $x_0$ . Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	Bool
HLD	Zmrazení simulace po dobu, kdy je HLD=on.	Bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních $m$ vstupů, kde $m$ je počet sloupců matice $B_d$ .	Double (F64)

## Výstupy

iE	Kód chyby bloku 0 ..... vše v pořádku, blok simuluje správně -213 .. nekompatibilita rozměrů matic stavového modelu xxx ... chybový kód xxx systému REXYGEN, více viz přílohu C	Error
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice Cd.	Double (F64)

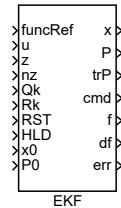
## Parametry

UD	Příznak použití matice Dd. Pokud je UD=off, matice Dd se při simulaci nepoužívá (chová se jako by byla nulová).	Bool
Ad	Matice (typu [n,n]) dynamiky diskrétního lineárního systému.	Double (F64)
Bd	Vstupní matice (typu [n,m]) diskrétního lineárního systému.	Double (F64)
Cd	Výstupní matice (typu [p,n]) diskrétního lineárního systému.	Double (F64)
Dd	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr UD=on. Je-li UD=off, rozměry matice Dd se nekontrolují.	Double (F64)
x0	Počáteční hodnota vektoru stavu (typu [n]) diskrétního lineárního systému.	Double (F64)

## EKF – Rozšířený (nelineární) Kalmanův filtr

Symbol bloku

Licence: [MODEL](#)



### Popis funkce

Funkční blok implementuje algoritmus nelineárního rekonstruktoru stavu známého jako Rozšířený Kalmanův filtr. Cílem je poskytnout odhad přímo neměřitelných stavových veličin nelineárního dynamického systému popsaného stavovou a výstupní rovnicí ve tvaru  $dx/dt = f(x, u) + w(t)$ ,  $y = h(x, u) + v(t)$  pro případ spojitého času a  $x(k+1) = f(x(k), u(k)) + w(k)$ ,  $y(k) = h(x(k), u(k)) + v(k)$  pro případ diskrétního systému. Veličiny  $w, v$  označují náhodný šum modelu a pozorování. Předpokládá se, že jde o náhodné procesy s nulovou střední hodnotou a Gaussovým rozdělením hustoty pravděpodobnosti definované kovariancemi  $Q$  and  $R$ , které se zadávají jako parametr bloku. Rozšířený Kalmanův filtr je nelineární verze algoritmu Kalmanova filtru pracující na principu linearizace stavové a výstupní rovnice v okolí aktuálního pracovního bodu. Jde o algoritmus typu prediktor-korektor, který střídá fázi predikce stavu v otevřené smyčce s využitím modelu a korekce (filtrace) na základě přímo měřených pozorování. Vektor výstupů- pozorování může být dodáván asynchronně vůči periodickému běhu algoritmu filtrace v libovolných okamžicích spouštění bloku.

Krok predikce je vykonáván každou periodu běhu bloku a řeší stavovou rovnici technikami numerické integrace, počínaje zadaným stavem  $x_0$  a počáteční kovariancí  $P_0$ . Volbou parametru *solver* uživatel vybírá numerickou metodu integrace příslušné vektorové diferenciální rovnice. Pro speciální případ volby *solver* = 1 algoritmus přechází na diskrétní variantu modelu a numerická integrace se redukuje na pouhé vyhodnocení pravé strany rekurze definované stavovou diferenční rovnicí  $x(k+1) = f(x(k), u(k))$ . Kromě vektoru stavu je v čase propagována také příslušná kovarianční matice  $P$ , která uchovává informaci o neurčitosti odhadu. Více detailů o jednotlivých numerických metodách lze nalézt v dokumentaci k bloku [NSSM](#).

Krok filtrace je proveden vždy když je v daném okamžiku spuštění bloku na vstup přivedena hodnota  $nz > 0$ . Toto signalizuje dostupný vektor měření na vstupu  $z$ , který je následně použit pro opravu odhadu stavu a jeho kovariance. Je možné kombinovat více pravých stran výstupní rovnice pomocí kooperujícího bloku [REXLANG](#). Toto může být užitečné v aplikacích s větším počtem senzorů, které dodávají data s různou periodou vzorkování nebo nepravidelně oproti periodickému spouštění bloku. Pro nastavení

$nz = 0$  signalizuje uživatelský algoritmus bloku nedostupnost měření a v dané periodě je provedena extrapolace stavu pozorovaného systému na základě modelu.

Kalmanův filtr obecně není optimální rekonstruktor stavu ve smyslu minimalizace střední kvadratické chyby odhadu. Nicméně, v praktických úlohách s dostatečně hladkou nelineární dynamikou systému poskytuje odhady stavu v rozumné kvalitě a je považován za de facto standard v oblasti nelineární filtrace. V případě zadání lineární pravé strany stavové a výstupní rovnice přechází algoritmus odhadu na standardní Kalmanův filtr, který je již optimální pro danou stochastickou formulaci problému odhadu stavu lineárního systému.

## Vstupy

funcRef	Odkaz na spolupracující blok REXLANG	Reference
u	Vektor vstupů modelu	Reference
z	Vektor výstupů (měření) modelu	Reference
nz	Číslo sady měřených výstupů	↓1 Long (I32)
Qk	Kovarianční matice stavového šumu	Reference
Rk	Kovarianční matice výstupního šumu	Reference
RST	Reset bloku	Bool
HLD	Pozastavení	Bool
x0	Vektor počáteční hodnoty stavu modelu	Reference
P0	Počáteční hodnota kovarianční matice modelu	Reference

## Parametry

nmax	Rezervovaná paměť pro výstupní matici (celkový počet prvků)	Long (I32)
		↓5 ↑10000 ⊙20
solver	Metoda numerické integrace	⊙2 Long (I32)
	1 ..... diskrétní model	
	2 ..... eulerova metoda (1. řád)	
	3 ..... metoda Adams-Bashforth 2. řádu	
	4 ..... metoda Adams-Bashforth 3. řádu	
	5 ..... metoda Adams-Bashforth 4. řádu	
	6 ..... metoda Adams-Bashforth 5. řádu	
	7 ..... Runge-Kutta 4. řádu	
	8 ..... implicitní Euler	
	9 ..... implicitní Euler (více iterací)	
	10 .... implicitní Adams-Moulton 2. řádu	
	11 .... implicitní Adams-Moulton 2. řádu (více iterací)	
	12 .... implicitní Adams-Moulton 3. řádu	
	13 .... implicitní Adams-Moulton 3. řádu (více iterací)	
	14 .... implicitní RadauIIA 2. řádu	
	15 .... implicitní RadauIIA 2. řádu (více iterací)	
	16 .... implicitní RadauIIA 3. řádu	
	17 .... implicitní RadauIIA 3. řádu (více iterací)	

## Výstupy

x	Vektor stavu modelu	Reference
P	Kovarianční matice stavu modelu	Reference
trP	Stopa kovarianční matice	Reference
cmd	Číslo příkazu navázaného bloku REXLANG	Long (I32)
f	Odkaz na vektor nastavovaný navázaným blokem REXLANG	Reference
df	Odkaz na matici nastavovanou navázaným blokem REXLANG	Reference
err	Kód chyby (0 bez chyby, jinak popis v system logu)	Long (I32)

## FOPDT – Model systému 1. řádu s dopravním zpožděním

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok FOPDT realizuje diskrétní simulátor lineárního systému prvního řádu s přidavným dopravním zpožděním, který je popsán následující přenosovou funkcí:

$$P(s) = \frac{k_0}{(\tau \cdot s + 1)} \cdot e^{-\text{del} \cdot s}$$

Diskrétní simulace používá přesnou diskretizaci přenosu  $P(s)$  pro periodu  $T_S$ , s níž je blok FOPDT spouštěn.

## Vstup

u	Analogový vstupní signál	Double (F64)
---	--------------------------	--------------

## Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

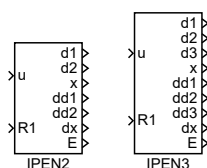
## Parametry

k0	Statické zesílení	⊙1.0	Double (F64)
del	Dopravní zpoždění [s]		Double (F64)
tau	Časová konstanta	⊙1.0	Double (F64)
nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (používá se pro interní alokaci paměti)	↓10 ↑10000000 ⊙1000	Long (I32)

## IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry

Symbole bloků

Licence: [MODEL](#)



### Popis funkce

Bloky IPEN2 a IPEN3 simulují dynamiku dvojitého a trojitého inverzního kyvadla na vozíku. Tyto modely umožňují uživatelům provádět experimenty s různými řídicími strategiemi, čímž jsou vhodné jak pro vzdělávací účely, tak pro vědecký výzkum.

Hlavním vstupem do modelů je analogový signál  $u$ , jehož interpretace závisí na nastavení parametru **IACC**:

- pro **IACC=on** je vstup  $u$  považován za sílu působící na vozík [N],
- pro **IACC=off** modely předpokládají, že vstup představuje rychlost [m/s].

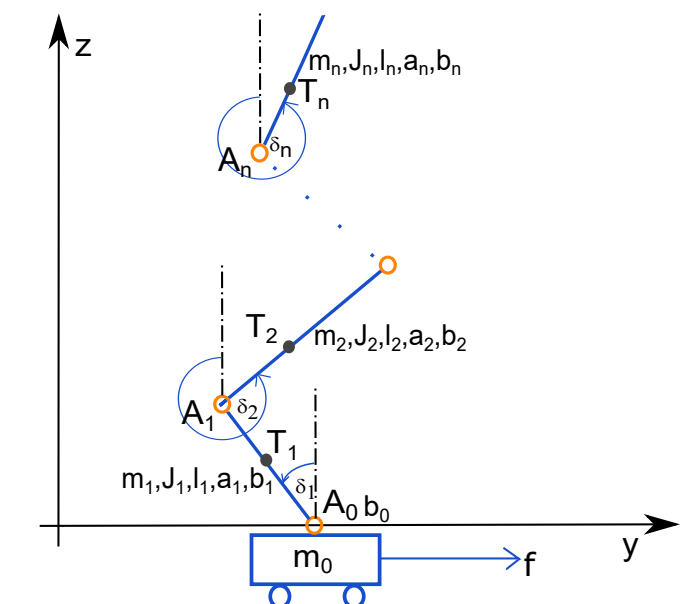
Signál **R1** slouží k resetování každého modelu do jeho počáteční konfigurace.

Oba modely lze detailně konfigurovat pomocí série parametrů, které odpovídají fyzikálním charakteristikám systému. Tyto parametry zahrnují relativní polohu těžiště kyvadel  $\mathbf{a}$ , moment setrvačnosti  $J$ , délku  $l$  a hmotnost kyvadel  $\mathbf{m}$ , koeficienty tlumení  $\mathbf{b}$  a počáteční stav systému (polohy  $\mathbf{d}_0$ , rychlosti  $\mathbf{dd}_0$ ). Schematické znázornění systému s parametry je na obrázku níže. Většina parametrů má jasnou definici, relativní poloha těžiště  $i$ -tého kyvadla  $a_i$  je dána vztahem

$$|A_{i-1}T_i| = a_i l_i,$$

kde  $A_{i-1}$  je pozice předchozího kloubu,  $T_i$  je pozice těžiště kyvadla a  $l_i$  je délka kyvadla.

Výpočet modelů je v souladu s matematickým modelem a fyzikálními parametry, které jsou podrobně popsány v literatuře [5]. Bloky [IPEN2pu](#) a [IPEN3pu](#) jsou využívány pro simulaci modelů inverzních kyvadel s dynamickými parametry.



Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool

### Parametr

a1	Relativní poloha těžiště	⊙1.0	Double (F64)
a2	Relativní poloha těžiště	⊙1.0	Double (F64)
a3	Relativní poloha těžiště	⊙1.0	Double (F64)
J1	Moment setrvačnosti kyvadla [kg.m <sup>2</sup> ]	⊙1.0	Double (F64)
J2	Moment setrvačnosti kyvadla [kg.m <sup>2</sup> ]	⊙1.0	Double (F64)
J3	Moment setrvačnosti kyvadla [kg.m <sup>2</sup> ]	⊙1.0	Double (F64)
l1	Délka kyvadla [m]	⊙1.0	Double (F64)
l2	Délka kyvadla [m]	⊙1.0	Double (F64)
l3	Délka kyvadla [m]	⊙1.0	Double (F64)
m1	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m2	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m3	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m0	Hmotnost vozíku [kg]	⊙1.0	Double (F64)
b1	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b2	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b3	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b0	Koeficient tlumení vozíku	⊙1.0	Double (F64)
d1_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)



d2_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
d3_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
x_0	Počáteční poloha vozíku [m]	⊙1.0	Double (F64)
dd1_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd2_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd3_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dx_0	Počáteční rychlost vozíku [m/s]	⊙1.0	Double (F64)
IACC	on=Vstup u je síla, off=Vstup u je rychlost		Bool

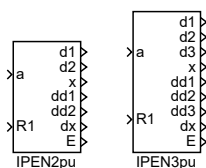
## Výstup

d1	Poloha kyvadla [rad]		Double (F64)
d2	Poloha kyvadla [rad]		Double (F64)
d3	Poloha kyvadla [rad]		Double (F64)
x	Poloha vozíku [m]		Double (F64)
dd1	Úhlová rychlost kyvadla [rad/s]		Double (F64)
dd2	Úhlová rychlost kyvadla [rad/s]		Double (F64)
dd3	Úhlová rychlost kyvadla [rad/s]		Double (F64)
dx	Rychlost vozíku [m/s]		Double (F64)
E	Příznak chyby		Bool

## IPEN2pu, IPEN3pu – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry

Symbole bloků

Licence: [MODEL](#)



### Popis funkce

Bloky IPEN2pu a IPEN3pu simulují dynamiku dvojitého a trojitého inverzního kyvadla na vozíku. Tyto modely umožňují uživatelům provádět experimenty s různými řídicími strategiemi, čímž jsou vhodné jak pro vzdělávací účely, tak pro vědecký výzkum.

Hlavním vstupem modelů je analogový signál **a** reprezentující zrychlení vozíku [ $m/s^2$ ]. Pro resetování modelů do počátečního stavu slouží signál **R1**.

Oba modely lze detailně konfigurovat pomocí dynamických parametrů systému **p1** - **p11** a počátečního stavu systému (polohy **d\_0**, rychlosti **dd\_0**). Dynamické parametry lze určit pomocí fyzikálních parametrů definovaných v blocích [IPEN2](#), [IPEN3](#) a rovnic uvedených níže. Detaily modelů jsou podrobně popsány v literatuře [5].

Pro model IPEN2pu jsou dynamické parametry definovány následovně:

$$p_1 = \frac{(m_1 a_1^2 + m_2) l_1^2 + J_1}{m_2 a_2 l_2 l_1}, \quad p_2 = \frac{a_2^2 l_2^2 m_2 + J_2}{m_2 a_2 l_2 l_1}, \quad p_3 = \frac{m_1 a_1 + m_2}{m_2 a_2 l_2},$$

$$p_4 = \frac{1}{l_1}, \quad p_5 = \frac{b_1}{m_2 a_2 l_2 l_1}, \quad p_6 = \frac{b_2}{m_2 a_2 l_2 l_1}.$$

Pro model IPEN3pu jsou dynamické parametry definovány jako:

$$p_1 = \frac{l_1(m_2 a_2 + m_3)}{a_3 l_3 m_3}, \quad p_2 = \frac{l_1}{l_2}, \quad p_3 = \frac{(m_1 a_1^2 + m_2 + m_3) l_1^2 + J_1}{a_3 l_3 m_3 l_2},$$

$$p_4 = \frac{(m_2 a_2^2 + m_3) l_2^2 + J_2}{a_3 l_3 m_3 l_2}, \quad p_5 = \frac{a_3^2 l_3^2 m_3 + J_3}{a_3 l_3 m_3 l_2}, \quad p_6 = \frac{l_1(m_1 a_1 + m_2 + m_3)}{a_3 l_3 m_3 l_2},$$

$$p_7 = \frac{a_2 m_2 + m_3}{a_3 l_3 m_3}, \quad p_8 = \frac{1}{l_2}, \quad p_9 = \frac{b_1}{a_3 l_3 m_3 l_2}, \quad p_{10} = \frac{b_2}{a_3 l_3 m_3 l_2}, \quad p_{11} = \frac{b_3}{a_3 l_3 m_3 l_2}.$$

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>a</b>	Zrychlení vozíku [ $m/s^2$ ]	Double (F64)
<b>R1</b>	Reset bloku	Bool

## Parametr

p1	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p2	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p3	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p4	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p5	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p6	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p7	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p8	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p9	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p10	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
p11	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0	Double (F64)
d1_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
d2_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
d3_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
x_0	Počáteční poloha vozíku [m]	⊙1.0	Double (F64)
dd1_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd2_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd3_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dx_0	Počáteční rychlost vozíku [m/s]	⊙1.0	Double (F64)

## Výstup

d1	Poloha kyvadla [rad]	Double (F64)
d2	Poloha kyvadla [rad]	Double (F64)
d3	Poloha kyvadla [rad]	Double (F64)
x	Poloha vozíku [m]	Double (F64)
dd1	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd2	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd3	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dx	Rychlost vozíku [m/s]	Double (F64)
E	Příznak chyby	Bool

## MDL – Model procesu

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok MDL realizuje diskretní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde  $K_0 > 0$  je statické zesílení  $k_0$ ,  $D \geq 0$  je dopravní zpoždění  $del$  a  $\tau_1, \tau_2 > 0$  jsou časové konstanty systému  $tau1$  a  $tau2$ .

Vstup

u	Analogový vstupní signál	Double (F64)
---	--------------------------	--------------

Výstup

y	Analogový výstupní signál	Double (F64)
---	---------------------------	--------------

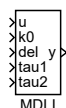
Parametry

k0	Statické zesílení	⊙1.0	Double (F64)
del	Dopravní zpoždění [s]		Double (F64)
tau1	První časová konstanta	⊙1.0	Double (F64)
tau2	Druhá časová konstanta	⊙2.0	Double (F64)
nmax	Délka vyrovnávací paměti pro dopravní zpoždění $del$ (používá se pro interní alokaci paměti)	↓10 ↑10000000 ⊙1000	Long (I32)

## MDLI – Model procesu s proměnnými parametry

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok MDLI realizuje diskrétní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde  $K_0 > 0$  je statické zesílení `k0`,  $D \geq 0$  je dopravní zpoždění `del` a  $\tau_1, \tau_2 > 0$  jsou časové konstanty systému `tau1` a `tau2`. Na rozdíl od bloku [MDL](#) mohou být všechny parametry systému průběžně měněny ze vstupů bloku.

### Vstupy

<code>u</code>	Analogový vstupní signál	Double (F64)
<code>k0</code>	Statické zesílení	Double (F64)
<code>del</code>	Dopravní zpoždění [s]	Double (F64)
<code>tau1</code>	První časová konstanta	Double (F64)
<code>tau2</code>	Druhá časová konstanta	Double (F64)

### Výstup

<code>y</code>	Analogový výstupní signál	Double (F64)
----------------	---------------------------	--------------

### Parametry

<code>nmax</code>	Délka vyrovnávací paměti pro dopravní zpoždění <code>del</code> (používá se pro interní alokaci paměti)	Long (I32) ↓10 ↑10000000 ⊙1000
-------------------	---	-----------------------------------

## MVD – Motorizovaný pohon ventilu

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok MVD je určen pro simulaci servoventilu. Vstup UP (DN) představuje binární povel pro otevírání (zavírání) ventilu konstantní rychlostí  $1/\tau_v$ , kde  $\tau_v$  je parametr bloku. Při UP = on (DN = on) otevírání probíhá až do úplného otevření  $y = \text{hilim}$  (úplného zavření  $y = \text{lolim}$ ) ventilu. Krajní poloha otevření (zavření) je signalizována koncovým spínačem HS (LS). Počáteční poloha ventilu po spuštění je  $y = y_0$ . Jestliže UP = DN = on nebo UP = DN = off, pak se poloha ventilu nemění (ani nezavírá ani neotvírá).

### Vstupy

UP	Otevřít	Bool
DN	Zavřít	Bool

### Výstupy

y	Poloha ventilu	Double (F64)
HS	Horní koncový spínač	Bool
LS	Dolní koncový spínač	Bool

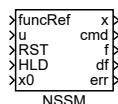
### Parametry

y0	Počáteční poloha ventilu	Double (F64)
$\tau_v$	Čas přejezdu mezi polohami $y = 0$ a $y = 1$ [s]	⊙10.0 Double (F64)
hilim	Horní mezní poloha (otevřeno)	⊙1.0 Double (F64)
lolim	Dolní mezní poloha (zavřeno)	Double (F64)

## NSSM – Nelineární stavový model

Symbol bloku

Licence: [MODEL](#)



### Popis funkce

Blok poskytuje řešení stavové a výstupní rovnice nelineárního dynamického systému popsaného soustavou diferenciálních rovnic  $dx/dt = f(x, u)$ ,  $y = h(x, u)$  pro spojitý čas nebo  $x(k+1) = f(x(k), u(k))$ ,  $y(k) = h(x(k), u(k))$  pro diskretní systém. Spojitá rovnice je interně diskretizována do tvaru  $x(t) = F(x(t-T), u(t))$ , kde  $T$  je perioda vzorkování NSSM bloku.

Metoda diskretizace a numerického řešení stavové rovnice závisí na volbě uživatelského parametru `solver`. K dispozici jsou metody jednokrokové (Runge-Kutta, Euler), vícekrokové (Adams-Bashforth) a implicitní (Adams-Moulton). U některých metod lze volit řád pro nalezení vhodného kompromisu mezi přesností a rychlostí výpočtu modelu.

Blok nepodporuje algoritmy řešení s variabilním krokem. Velikost časového kroku je vždy shodná s periodou spouštění funkčního bloku.

Nelineární vektorové funkce  $f(x, u)$ ,  $h(x, u)$  musí být implementovány v `REXLANG` bloku, který je připojen k bloku `NSSM` specifickým způsobem. Vstup `funcRef` bloku `NSSM` musí být připojen na výstup `y0` spolupracujícího `REXLANG` bloku a výstup `y0` nesmí být použit interně v kódu bloku `REXLANG`. Výstupy `x`, `f` a `df` bloku `NSSM` musí být připojeny na vstupy bloku `REXLANG`. Tyto vstupy musí být zpracovány v bloku `REXLANG` jako vstupní pole. Hlavní `main` funkce bloku `REXLANG` musí nastavovat hodnotu vektorové funkce  $f(x, u)$  do vektoru `f` (tj. do vstupního pole, kam je `f` připojeno), matice  $df(x, u)/dx$  se ukládá do matice `af`.

Blok `NSSM` volá funkci `main` bloku `REXLANG` pokud je potřeba vyhodnotit pravou stranu rovnice pro numerickou integraci (například Runge-Kutta metoda provádí 4 volání v každé periodě spuštění bloku s různými hodnotami argumentu `x`). Blok `REXLANG` musí být ve schématu zakázán, aby se zabránilo jeho automatické spuštění systémem `REXYGEN`. Pokud je potřeba spuštění bloku `REXLANG` (například pro výpočet nelineární výstupní rovnice  $y = h(x, u)$ ), je doporučeno zapojit výstup `cmd` bloku `NSSM` na vstup bloku `REXLANG` pro rozlišení mezi voláním z `NSSM` bloku pro numerickou integraci stavové rovnice (`cmd = 0`) a voláním ze systému `REXYGEN system` (`cmd = -1`).

Poznámky:

- výpočet Jakobiánu  $df(x, u)/dx$  je nutný jen pro implicitní metody.
- velikost pole `x` (a tím také `f`, `df`) je definováno dimenzí pole `x0`. Změna velikosti je dovolena pouze při aktivaci vstupu `reset` (`RST`).

- `solver=1: discrete` signalizuje diskrétní stavovou rovnici s funkcí  $f$  označující pravou stranu příslušné diferenční rovnice. Tento režim nevyžaduje numerickou integraci a algoritmus se redukuje na prosté vyhodnocení pravé strany definované rekurzí; tento režim v principu nevyžaduje použití NSSM bloku (lze řešit přímo pomocí REXLANG); tento režim je zachován zejména z důvodu kompatibility s příbuzným blokem EKF.
- pro blok NSSM je nutné připojení výstupu `cmd`, protože `cmd>0` indikuje počet měření, které musí vracet blok REXLANG při vyhodnocení  $f = h(x, u)$ ,  $df = dh(x, u)/dx$ .

## Vstupy

<code>funcRef</code>	Odkaz na spolupracující blok REXLANG	Reference
<code>u</code>	Vektor vstupů modelu	Reference
<code>RST</code>	Reset bloku	Bool
<code>HLD</code>	Pozastavení	Bool
<code>x0</code>	Vektor počáteční hodnoty stavu modelu	Reference

## Parametry

<code>nmax</code>	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓5 ↑10000 ⊙20	Long (I32)
<code>solver</code>	Metoda numerické integrace 1 ..... diskrétní model 2 ..... Eulerova metoda (1. řád) 3 ..... metoda Adams-Bashforth 2. řádu 4 ..... metoda Adams-Bashforth 3. řádu 5 ..... metoda Adams-Bashforth 4. řádu 6 ..... metoda Adams-Bashforth 5. řádu 7 ..... Runge-Kutta 4. řádu 8 ..... implicitní Euler 9 ..... implicitní Euler (více iterací) 10 .... implicitní Adams-Moulton 2. řádu 11 .... implicitní Adams-Moulton 2. řádu (více iterací) 12 .... implicitní Adams-Moulton 3. řádu 13 .... implicitní Adams-Moulton 3. řádu (více iterací) 14 .... implicitní RadauIIA 2. řádu 15 .... implicitní RadauIIA 2. řádu (více iterací) 16 .... implicitní RadauIIA 3. řádu 17 .... implicitní RadauIIA 3. řádu (více iterací)	⊙2 Long (I32)

## Výstupy

<code>x</code>	Vektor stavu modelu	Reference
<code>y</code>	Výstupní vektor modelu	Reference
<code>cmd</code>	Číslo příkazu navázaného bloku REXLANG	Long (I32)
<code>f</code>	Odkaz na vektor nastavovaný navázaným blokem REXLANG	Reference



df	Odkaz na matici nastavovanou navázaným blokem REXLANG	Reference
err	Kód chyby (0 bez chyby, jinak popis v system logu)	Long (132)

## SOPDT – Model systému 2. řádu s dopravním zpožděním

Symbol bloku

Licence: **STANDARD**

## Popis funkce

Blok SOPDT realizuje diskretní simulátor lineárního systému druhého řádu s přidavným dopravním zpožděním, který je alternativně popsán, v závislosti na parametru **itf**, následujícími přenosovými funkcemi:

$$\begin{aligned} \text{itf} = 1: \quad P(s) &= \frac{\text{pb1} \cdot s + \text{pb0}}{s^2 + \text{pa1} \cdot s + \text{pa0}} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 2: \quad P(s) &= \frac{\text{k0} (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) (\text{tau2} \cdot s + 1)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 3: \quad P(s) &= \frac{\text{k0} \cdot \text{om}^2 \cdot (\text{tau}/\text{om} \cdot s + 1)}{(s^2 + 2 \cdot \text{xi} \cdot \text{om} \cdot s + \text{om}^2)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 4: \quad P(s) &= \frac{\text{k0} (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) s} \cdot e^{-\text{del} \cdot s} \end{aligned}$$

Diskretní simulace používá přesnou diskretizaci přenosu  $P(s)$  pro periodu  $T_S$ , s níž je blok SOPDT spouštěn.

## Vstup

**u** Analogový vstupní signál Double (F64)

## Výstup

**y** Analogový výstupní signál Double (F64)

## Parametry

<b>itf</b>	Tvar přenosové funkce	⊙1	Long (I32)
	1 ..... obecný tvar		
	2 ..... reálné póly ve jmenovateli		
	3 ..... komplexní póly (kmitavý systém)		
	4 ..... systém s integrátorem		
<b>k0</b>	Statické zesílení	⊙1.0	Double (F64)
<b>tau</b>	Časová konstanta v čitateli		Double (F64)
<b>tau1</b>	První časová konstanta ve jmenovateli	⊙1.0	Double (F64)

<code>tau2</code>	Druhá časová konstanta ve jmenovateli	⊙1.0	Double (F64)
<code>om</code>	Vlastní frekvence	⊙1.0	Double (F64)
<code>xi</code>	Relativní koeficient tlumení	⊙1.0	Double (F64)
<code>pb0</code>	Koeficient čitatele: $s^0$	⊙1.0	Double (F64)
<code>pb1</code>	Koeficient čitatele: $s^1$	⊙1.0	Double (F64)
<code>pa0</code>	Koeficient jmenovatele: $s^0$	⊙1.0	Double (F64)
<code>pa1</code>	Koeficient jmenovatele: $s^1$	⊙1.0	Double (F64)
<code>del</code>	Dopravní zpoždění [s]		Double (F64)
<code>nmax</code>	Délka vyrovnávací paměti pro dopravní zpoždění <code>del</code> (používá se pro interní alokaci paměti)	↓10 ↑10000000	⊙1000 Long (I32)



## Kapitola 14

# MATRIX – Bloky pro maticové a vektorové operace

### Obsah

---

14.1 Poznámky k implementaci: . . . . .	399
CNA – * Konstantní pole (vektor/matice) . . . . .	400
MB_DASUM – * Součet absolutních hodnot . . . . .	401
MB_DAXPY – * Provádí $y := a*x + y$ pro vektory $x, y$ . . . . .	402
MB_DCOPY – * Kopíruje vektor $x$ do vektoru $y$ . . . . .	403
MB_DDOT – * Skalární součin dvou vektorů . . . . .	404
MB_DGEMM – * Provádí $C := \alpha*\text{op}(A)*\text{op}(B) + \beta*C$ , where $\text{op}(X) = X$ or $\text{op}(X) = X^T$ . . . . .	405
MB_DGEMV – * Provádí $y := \alpha*A*x + \beta*y$ or $y := \alpha*A^T*x + \beta*y$ . . . . .	406
MB_DGER – * Provádí $A := \alpha*x*y^T + A$ . . . . .	407
MB_DNRM2 – * Eukleidovská norma vektoru . . . . .	408
MB_DROT – * Rovinná rotace vektoru . . . . .	409
MB_DSCAL – * Násobení vektoru konstantou . . . . .	410
MB_DSWAP – * Záměna dvou vektorů . . . . .	411
MB_DTRMM – * Provádí $B := \alpha*\text{op}(A)*B$ or $B := \alpha*B*\text{op}(A)$ , where $\text{op}(X) = X$ or $\text{op}(X) = X^T$ pro trojúhelníkovou matici $A$ . . . . .	412
MB_DTRMV – * Provádí $x := A*x$ or $x := A^T*x$ pro trojúhelníkovou matici $A$ . . . . .	413
MB_DTRSV – * Řeší jednu ze soustav rovnic $A*x = B$ nebo $A^T*x = B$ pro trojúhelníkovou matici $A$ . . . . .	414
ML_DGEBAK – * Zpětná transformace k ML_DGEBAL levých nebo pravých vlastních vektorů . . . . .	415
ML_DGEBAL – * Vyvážení obecné reálné matice . . . . .	416
ML_DGEBRD – * Redukce obedné reálné matice do bidiagonální formy pomocí ortogonální transformace . . . . .	417

ML_DGECON – * Odhad převrácené hodnoty čísla podmíněnosti obecné reálné matice . . . . .	418
ML_DGEES – * Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů . . . . .	419
ML_DGEEV – * Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů . . . . .	420
ML_DGEHRD – * Redukce reálné obecné matice A na horní Hes- senbergovu formu . . . . .	421
ML_DGELQF – * Výpočet LQ factorizace reálné matice A s rozměry M x N . . . . .	422
ML_DGELSD – * Výpočet řešení s minimální normou reálné lineární úlohy nejmenších čtverců . . . . .	423
ML_DGEQRF – * Výpočet QR factorizace reálné matice A s rozměry M x N . . . . .	424
ML_DGESDD – * Výpočet singulární dekompozice (SVD) reálné ma- tice A s rozměry M x N . . . . .	425
ML_DLACPY – * Kopíruje celou nebo část matice do jiné matice . .	426
ML_DLANGE – * Výpočet některé z maticových norem obecné matice	427
ML_DLASET – * Inicializuje mimodiagonální a diagonální prvky ma- tice na zadané hodnoty . . . . .	428
ML_DTRSYL – * Řešení reálné Sylvesterovy rovnice pro kvazitroj- úhelníkové matice A a B . . . . .	429
MX_AT – * Hodnota prvku matice/vektoru . . . . .	430
MX_ATSET – * Nastavení hodnoty prvku matice/vektoru . . . . .	431
MX_CNADD – * Přičte skalár ke každému prvku matice/vektoru . .	432
MX_CNMUL – * Vynásobí matici/vektor skalárem . . . . .	433
MX_CTODPA – * Discretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací . . . . .	434
MX_DIM – * Dimenze matice/vektoru . . . . .	435
MX_DIMSET – * Nastavení dimenze matice/vektoru . . . . .	436
MX_DSAGET – * Uložení submatice A do matice B . . . . .	437
MX_DSAREF – * Nastavení odkazu na submatici A do matice B . .	438
MX_DSASET – * Uložení matice A do submatice v B . . . . .	439
MX_DTRNSP – * Transpozice obecné matice: $B := \alpha * A^T$ . . .	440
MX_DTRNSQ – * Transpozice čtvercové matice na místě: $A := \alpha * A^T$ . . . . .	441
MX_FILL – * Vyplnění reálné matice/vektoru . . . . .	442
MX_MAT – * Blok pro uložení dat matice . . . . .	443
MX_RAND – * Náhodně vygerenovaná matice nebo vektor . . . . .	444
MX_REFCOPY – * Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů . . . . .	445
MX_SLFS – Ukládání a čtení matice/vektoru do souboru nebo tex- tového retězce . . . . .	446

<b>MX_VEC</b> – * Blok pro uložení dat vektoru . . . . .	<b>449</b>
<b>MX_WRITE</b> – * Výpis matice/vektoru do konzole/systemého logu .	<b>450</b>
<b>RTOV</b> – Vektorový multiplexer . . . . .	<b>451</b>
<b>SWVMR</b> – * Přepínač vektorového/maticového/odkazovacího signálu	<b>453</b>
<b>VTOR</b> – * Vektorový demultiplexer . . . . .	<b>454</b>

---

## 14.1 Poznámky k implementaci:

První prvek matice má index(0,0), první prvek vektoru index 0.

Vektor není samostatný objekt, ale matice s jedním sloupcem. Může být i matice s jedním řádkem, které se říká řádkový vektor. Pokud se někdy v systému REXYGEN vyžaduje vektor, myslí se tím sloupcový (tj. matice s jedním sloupcem).

Matice jsou mezi bloky předávány odkazem. V rámci nějakého bloku (nejčastěji **MX\_MAT** nebo **MX\_VEC**) se alokuje paměť pro celou matici a pak se jen předává jedno číslo, které znamená adresu této paměti. To znamená, že všechny bloky, které si matici předávají pořád pracují s jednou kopií a vzájemně si matici přepisují. Pokud je potřeba vytvořit kopii matice, je potřeba použít další blok **MX\_MAT** a data zkopírovat (např. blokem **MB\_DCOPY** ).

Některé bloky pracující s vektorem (např. **MB\_DCPY**, **RTOV**, **VTOR**) nekontrolují přesné rozměry (takže např. matici 10x10 chápou jako 100 prvkový vektor). Pro tento případ je důležité vědět, že se hodnoty ukládají po sloupcích (nejdříve je v paměti první sloupec, pak druhý sloupec, atd.). Tyto bloky pak nepracují správně s odkazy na výřez matice získaný blokem **MX\_DSAREF**.

Většina bloků má odkaz/referenci na matici jak na vstupu tak na výstupu. Obojí je odkaz na stejnou matici, takže je jedno, kam další blok připojíme. Protože ale pořadí spouštění bloků se řídí tokem signálu, připojením na výstup jiného bloku definujeme pořadí spouštění bloků a tím i pořadí vykonávání maticových operací.

CNA – \* **Konstantní pole (vektor/matice)**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Parametry

<code>filename</code>	Datový soubor s hodnotami oddělenými čárkou	String
<code>TRN</code>	Transponuj načtenou matici	Bool
<code>nmax</code>	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓2 ↑10000000 ⊖100	Long (I32)
<code>etype</code>	Typ prvků	⊙8 Long (I32)
	1 ..... Bool	
	2 ..... Byte (U8)	
	3 ..... Short (I16)	
	4 ..... Long (I32)	
	5 ..... Word (U16)	
	6 ..... DWord (U32)	
	7 ..... Float (F32)	
	8 ..... Double (F64)	
	-- .....	
	10 .... Large (I64)	
<code>acn</code>	Počáteční hodnota pole	⊙[0 1 2 3] Double (F64)

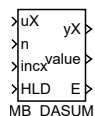
## Výstup

<code>vec</code>	Odkaz na vektor/matici dat	Reference
------------------	----------------------------	-----------



MB\_DASUM – \* **Součet absolutních hodnot**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

<code>uX</code>	Vstupní odkaz na vektor <code>x</code>	Reference
<code>n</code>	Počet zpracovaných prvků vektoru	Long (I32)
<code>incx</code>	Přírůstek indexu vektoru <code>x</code>	Long (I32)
<code>HLD</code>	Pozastavení	Bool

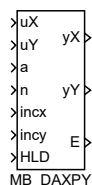
## Výstupy

<code>yX</code>	Výstupní odkaz na vektor <code>x</code>	Reference
<code>value</code>	Návratová hodnota funkce	Double (F64)
<code>E</code>	Příznak chyby	Bool

MB\_DAXPY – \* **Provádí  $y := a \cdot x + y$  pro vektory  $x, y$**

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

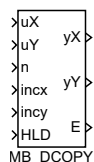
uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
a	Skalární koeficient a	Double (F64)
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

### Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB\_DCOPY – \* **Kopíruje vektor x do vektoru y**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

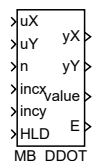
uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

## Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB\_DDOT – \* **Skalární součin dvou vektorů**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

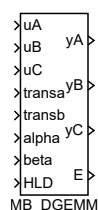
## Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
value	Návratová hodnota funkce	Double (F64)
E	Příznak chyby	Bool

**MB\_DGEMM** – \* **Provádí  $C := \alpha * \text{op}(A) * \text{op}(B) + \text{beta} * C$ , where  $\text{op}(X) = X$  or  $\text{op}(X) = X^T$**

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

<code>uA</code>	Vstupní odkaz na matici A		Reference
<code>uB</code>	Vstupní odkaz na matici B		Reference
<code>uC</code>	Vstupní odkaz na matici C		Reference
<code>transa</code>	Transpozice matice A	↓0 ↑3	Long (I32)
<code>transb</code>	Transpozice matice B	↓0 ↑3	Long (I32)
<code>alpha</code>	Skalární koeficient alpha		Double (F64)
<code>beta</code>	Skalární koeficient beta		Double (F64)
<code>HLD</code>	Pozastavení		Bool

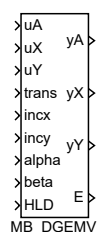
## Výstupy

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yB</code>	Výstupní odkaz na matici B	Reference
<code>yC</code>	Výstupní odkaz na matici C	Reference
<code>E</code>	Příznak chyby	Bool

**MB\_DGEMV** – \* **Provádí**  $y := \alpha * A * x + \beta * y$  or  $y := \alpha * A^T * x + \beta * y$

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uA	Vstupní odkaz na matici A	Reference
uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
trans	Transpozice vstupní matice	↓0 ↑3 Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
alpha	Skalární koeficient alpha	Double (F64)
beta	Skalární koeficient beta	Double (F64)
HLD	Pozastavení	Bool

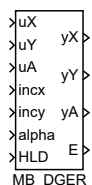
## Výstupy

yA	Výstupní odkaz na matici A	Reference
yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB\_DGER – \* **Provádí  $A := \alpha * x * y^T + A$**

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
uA	Vstupní odkaz na matici A	Reference
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

## Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
yA	Výstupní odkaz na matici A	Reference
E	Příznak chyby	Bool

MB\_DNRM2 – \* **Eukleidovská norma vektoru**

Symbol bloku

Licence: [STANDARD](#)

>uX	yX >
>n	
>incx	value >
>HLD	E >
MB_DNRM2	

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uX	Vstupní odkaz na vektor x	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
HLD	Pozastavení	Bool

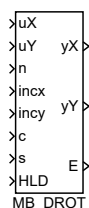
## Výstupy

yX	Výstupní odkaz na vektor x	Reference
value	Návratová hodnota funkce	Double (F64)
E	Příznak chyby	Bool



## MB\_DROT – \* Rovinná rotace vektoru

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

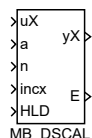
uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
c	Skalární koeficient c	Double (F64)
s	Skalární koeficient s	Double (F64)
HLD	Pozastavení	Bool

## Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB\_DSCAL – \* **Násobení vektoru konstantou**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

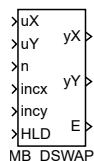
uX	Vstupní odkaz na vektor x	Reference
a	Skalární koeficient a	Double (F64)
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
HLD	Pozastavení	Bool

## Výstupy

yX	Výstupní odkaz na vektor x	Reference
E	Příznak chyby	Bool

## MB\_DSWAP – \* Záměna dvou vektorů

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

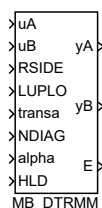
## Výstupy

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

**MB\_DTRMM** – \* Provádí  $B := \alpha * \text{op}(A) * B$  or  $B := \alpha * B * \text{op}(A)$ , where  $\text{op}(X) = X$  or  $\text{op}(X) = X^T$  pro trojúhelníkovou matici  $A$

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

<code>uA</code>	Vstupní odkaz na matici $A$	Reference
<code>uB</code>	Vstupní odkaz na matici $B$	Reference
<code>RSIDE</code>	Operace je aplikována z pravé strany	Bool
<code>LUPLO</code>	Matice $A$ je dolní trojúhelníková matice	Bool
<code>transa</code>	Transpozice matice $A$	↓0 ↑3 Long (I32)
<code>NDIAG</code>	Nepředpokládá se, že matice $A$ má na diagonále jedničky	Bool
<code>alpha</code>	Skalární koeficient $\alpha$	Double (F64)
<code>HLD</code>	Pozastavení	Bool

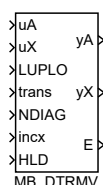
## Výstupy

<code>yA</code>	Výstupní odkaz na matici $A$	Reference
<code>yB</code>	Výstupní odkaz na matici $B$	Reference
<code>E</code>	Příznak chyby	Bool

**MB\_DTRMV** – \* Provádí  $x := A * x$  or  $x := A^T * x$  pro trojúhelníkovou matici **A**

Symbol bloku

Licence: [STANDARD](#)



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

<b>uA</b>	Vstupní odkaz na matici <b>A</b>		Reference
<b>uX</b>	Vstupní odkaz na vektor <b>x</b>		Reference
<b>LUPLO</b>	Matice <b>A</b> je dolní trojúhelníková matice		Bool
<b>trans</b>	Transpozice vstupní matice	↓0 ↑3	Long (I32)
<b>NDIAG</b>	Nepředpokládá se, že matice <b>A</b> má na diagonále jedničky		Bool
<b>incx</b>	Přírůstek indexu vektoru <b>x</b>		Long (I32)
<b>HLD</b>	Pozastavení		Bool

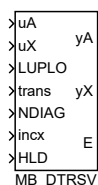
## Výstupy

<b>yA</b>	Výstupní odkaz na matici <b>A</b>		Reference
<b>yX</b>	Výstupní odkaz na vektor <b>x</b>		Reference
<b>E</b>	Příznak chyby		Bool

**MB\_DTRSV** – \* Řeší jednu ze soustav rovnic  $A*x = B$  nebo  $A^T*x = B$  pro trojúhelníkovou matici  $A$

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

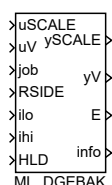
uA	Vstupní odkaz na matici A	Reference
uX	Vstupní odkaz na vektor x	Reference
LUPLO	Matice A je dolní trojúhelníková matice	Bool
trans	Transpozice vstupní matice	↓0 ↑3 Long (I32)
NDIAG	Nepředpokládá se, že matice A má na diagonále jedničky	Bool
incx	Přírůstek indexu vektoru x	Long (I32)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yX	Výstupní odkaz na vektor x	Reference
E	Příznak chyby	Bool

## ML\_DGEBAK – \* Zpětná transformace k ML\_DGEBAL levých nebo pravých vlastních vektorů

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

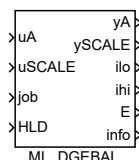
uSCALE	Vstupní odkaz na vektor SCALE		Reference
uV	Odkaz na matici pravých nebo levých vlastních vektorů, které mají být transformovány		Reference
job	Typ požadované zpětné transformace	↓0 ↑4	Long (I32)
RSIDE	Operace je aplikována z pravé strany		Bool
ilo	Dolní index (od nuly) řádku a sloupce pracovní submatice		Long (I32)
ihi	Horní index (od nuly) řádku a sloupce pracovní submatice		Long (I32)
HLD	Pozastavení		Bool

### Výstupy

ySCALE	Výstupní odkaz na vektor SCALE		Reference
yV	Odkaz na matici transformovaných pravých nebo levých vlastních vektorů		Reference
E	Příznak chyby		Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu		Long (I32)

ML\_DGEBAL – \* **Vyvážení obecné reálné matice**

Symbol bloku

Licence: [MATRIX](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uA	Vstupní odkaz na matici A	Reference
uSCALE	Vstupní odkaz na vektor SCALE	Reference
job	Specifikuje operace, které mají být provedeny s maticí A ↓0 ↑4	Long (I32)
HLD	Pozastavení	Bool

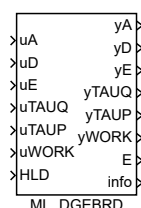
## Výstupy

yA	Výstupní odkaz na matici A	Reference
ySCALE	Výstupní odkaz na vektor SCALE	Reference
ilo	Dolní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
ihi	Horní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)



## ML\_DGEBRD — \* Redukce obedné reálné matice do bidiagonální formy pomocí ortogonální transformace

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici A	Reference
uD	Diagonální prvky bidiagonální matice B	Reference
uE	Mimodiagonální prvky bidiagonální matice B	Reference
uTAUQ	Reference na vektor skalárních faktorů elementárních reflektorů, které reprezentují ortogonální matici Q	Reference
uTAUP	Reference na vektor skalárních faktorů elementárních reflektorů, které reprezentují ortogonální matici P	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

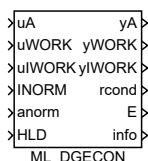
### Výstupy

yA	Výstupní odkaz na matici A	Reference
yD	Výstupní reference na D	Reference
yE	Výstupní reference na E	Reference
yTAUQ	Výstupní reference na TAUQ	Reference
yTAUP	Výstupní reference na TAUP	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

## ML\_DGECON – \* Odhad převrácené hodnoty čísla podmíněnosti obecné reálné matice

Symbol bloku

Licence: [MATRIX](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

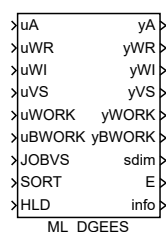
uA	Vstupní odkaz na matici A	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
uIWORK	Vstupní odkaz na celočíselný pracovní vektor WORK	Reference
INORM	Použij tzv. Infinity-norm (maximum z řádkových součtů absolutních hodnot prvků)	Bool
anorm	Norma původní matice A	Double (F64)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
yIWORK	Výstupní odkaz na celočíselný pracovní vektor WORK	Reference
rcond	Převrácená hodnota čísla podmíněnosti matice A	Double (F64)
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

## ML\_DGEES – \* Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

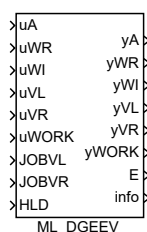
uA	Vstupní odkaz na matici A	Reference
uWR	Vstupní odkaz na vektor reálných částí vlastních čísel	Reference
uWI	Vstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
uVS	Vstupní reference na ortogonální matici Schurových vektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
uBWORK	Vstupní odkaz na Booleovský pracovní vektor WORK	Reference
JOBVS	Je-li true, pak se počítají Schurovy vektory	Bool
SORT	Je-li true, pak se vlastní čísla setřídí	Bool
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yWR	Výstupní odkaz na vektor reálných částí vlastních čísel	Reference
yWI	Výstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
yVS	Výstupní reference na VS	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
yBWORK	Výstupní odkaz na Booleovský pracovní vektor WORK	Reference
sdim	Je-li SORT, pak udává počet vlastních čísel pro něž je SELECT true, jinak 0	Long (I32)
E	Příznak chyby	Bool
info	Informace o výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

## ML\_DGEEV – \* Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

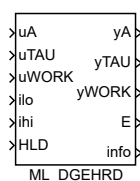
uA	Vstupní odkaz na matici A	Reference
uWR	Vstupní odkaz na vektor reálných částí vlastních čísel	Reference
uWI	Vstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
uVL	Vstupní reference na matici levých vlastních vektorů	Reference
uVR	Vstupní reference na matici pravých vlastních vektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
JOBVL	Je-li true, pak se počítají levé vlastní vektory	Bool
JOBVR	Je-li true, pak se počítají pravé vlastní vektory	Bool
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yWR	Výstupní odkaz na vektor reálných částí vlastních čísel	Reference
yWI	Výstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
yVL	Výstupní reference na VL	Reference
yVR	Výstupní reference na VR	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou hodnotu	Long (I32)

## ML\_DGEHRD – \* Redukce reálné obecné matice A na horní Hessenbergovu formu

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

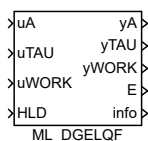
uA	Vstupní odkaz na matici A	Reference
uTAU	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
ilo	Dolní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
ihi	Horní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yTAU	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou hodnotu	Long (I32)

## ML\_DGELQF – \* Výpočet LQ factorizace reálné matice A s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici A	Reference
uTAU	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

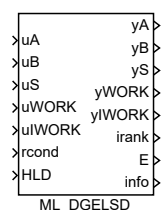
### Výstupy

yA	Výstupní odkaz na matici A	Reference
yTAU	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

## ML\_DGELSD – \* Výpočet řešení s minmální normou reálné lineární úlohy nejmenších čtverců

Symbol bloku

Licence: [MATRIX](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

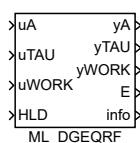
uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uS	Vstupní reference na vektor singulárních hodnot	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
uIWORK	Vstupní odkaz na celočíselný pracovní vektor WORK	Reference
rcond	0	Double (F64)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
yS	Výstupní reference na vektor singulárních hodnot	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
yIWORK	Výstupní odkaz na celočíselný pracovní vektor WORK	Reference
irank	0	Long (I32)
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

## ML\_DGEQRF – \* Výpočet QR factorizace reálné matice $A$ s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici $A$	Reference
uTAU	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

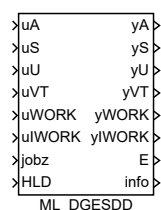
### Výstupy

yA	Výstupní odkaz na matici $A$	Reference
yTAU	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)



## ML\_DGESDD – \* Výpočet singulární dekompozice (SVD) reálné matice $A$ s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici $A$	Reference
uS	Vstupní reference na vektor singulárních hodnot	Reference
uU	Vstupní reference na matici obsahující levé singulární vektory matice $A$	Reference
uVT	Vstupní reference na matici obsahující pravé singulární vektory matice $A$	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
uIWORK	Vstupní odkaz na celočíselný pracovní vektor WORK	Reference
jobz	Specifikuje volby výpočtu	Long (I32)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici $A$	Reference
yS	Výstupní reference na vektor singulárních hodnot	Reference
yU	Výstupní reference na matici obsahující levé singulární vektory matice $A$	Reference
yVT	Výstupní reference na matici obsahující pravé singulární vektory matice $A$	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
yIWORK	Výstupní odkaz na celočíselný pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML\_DLACPY – \* **Kopíruje celou nebo část matice do jiné matice**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

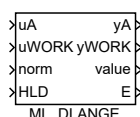
uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 ..... Celá	
	1 ..... Celá	
	2 ..... Horní	
	3 ..... Dolní	
HLD	Pozastavení	Bool

## Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

## ML\_DLANGE – \* Výpočet některé z maticových norem obecné matice

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

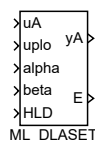
uA	Vstupní odkaz na matici A	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
norm	Zvolená maticová norma	↓0 ↑4 Long (I32)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
value	Návratová hodnota funkce	Double (F64)
E	Příznak chyby	Bool

**ML\_DLASET – \* Inicializuje mimodiagonální a diagonální prvky matice na zadané hodnoty**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

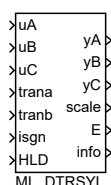
uA	Vstupní odkaz na matici A	Reference
uplo	Kopírovaná část matice 0 ..... Celá 1 ..... Celá 2 ..... Horní 3 ..... Dolní	Long (I32)
alpha	Skalární koeficient alpha	Double (F64)
beta	Skalární koeficient beta	Double (F64)
HLD	Pozastavení	Bool

## Výstupy

yA	Výstupní odkaz na matici A	Reference
E	Příznak chyby	Bool

## ML\_DTRSYL – \* Řešení reálné Sylvesterovy rovnice pro kvazitrojúhelníkové matice A a B

Symbol bloku

Licence: [MATRIX](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici A		Reference
uB	Vstupní odkaz na matici B		Reference
uC	Vstupní odkaz na matici C		Reference
trana	Transpozice matice A	↓0 ↑3	Long (I32)
tranb	Transpozice matice B	↓0 ↑3	Long (I32)
isgn	Znaménko v rovnici (1 nebo -1)	↓-1 ↑1	Long (I32)
HLD	Pozastavení		Bool

### Výstupy

yA	Výstupní odkaz na matici A		Reference
yB	Výstupní odkaz na matici B		Reference
yC	Výstupní odkaz na matici C		Reference
scale	Scale		Double (F64)
E	Příznak chyby		Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu		Long (I32)

MX\_AT – \* **Hodnota prvku matice/vektoru**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

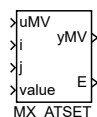
uMV	Vstupní reference na matici nebo vektor	Reference
i	Řádkový index prvku	↓0 Long (I32)
j	Sloupcový index prvku	↓0 Long (I32)

## Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
value	Hodnota prvku v pozici (i,j)	Long (I32)
E	Příznak chyby	Bool

## MX\_ATSET – \* Nastavení hodnoty prvku matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uMV	Vstupní reference na matici nebo vektor		Reference
i	Řádkový index prvku	↓0	Long (I32)
j	Sloupcový index prvku	↓0	Long (I32)
value	Hodnota, která má být nastavena do prvku v pozici (i,j)		Long (I32)

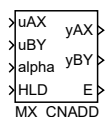
### Výstupy

yMV	Výstupní reference na matici nebo vektor		Reference
E	Příznak chyby		Bool

**MX\_CNADD – \* Přičte skalár ke každému prvku matice/vektoru**

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

uAX	Vstupní reference na matici A nebo vektor X	Reference
uBY	Vstupní reference na matici B nebo vektor Y	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

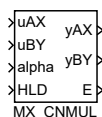
Výstupy

yAX	Výstupní reference na matici A nebo vektor X	Reference
yBY	Výstupní reference na matici B nebo vektor Y	Reference
E	Příznak chyby	Bool



**MX\_CNMUL – \* Vynásobí matici/vektor skalárem**

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

**Vstupy**

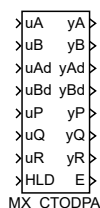
uAX	Vstupní reference na matici A nebo vektor X	Reference
uBY	Vstupní reference na matici B nebo vektor Y	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

**Výstupy**

yAX	Výstupní reference na matici A nebo vektor X	Reference
yBY	Výstupní reference na matici B nebo vektor Y	Reference
E	Příznak chyby	Bool

## MX\_CTODPA – \* Discretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uAd	Vstupní reference na diskretizovanou matici A	Reference
uBd	Vstupní reference na diskretizovanou matici B	Reference
uP	Vstupní reference na pomocnou matici	Reference
uQ	Vstupní reference na pomocnou matici	Reference
uR	Vstupní reference na pomocnou matici	Reference
HLD	Pozastavení	Bool

### Parametry

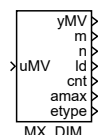
is	Řád Padéovy aproximace	↓0 ↑4 ⊖2	Long (I32)
eps	Přesnost aproximace	↓1e-20 ↑0.001 ⊖1e-15	Double (F64)

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
yAd	Výstupní reference na diskretizovanou matici A	Reference
yBd	Výstupní reference na diskretizovanou matici B	Reference
yP	Výstupní reference na pomocnou matici	Reference
yQ	Výstupní reference na pomocnou matici	Reference
yR	Výstupní reference na pomocnou matici	Reference
E	Příznak chyby	Bool

## MX\_DIM – \* Dimenze matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstup

uMV	Vstupní reference na matici nebo vektor	Reference
-----	---	-----------

## Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
ld	Vedoucí dimenze ( $\geq$ počtu řádků)	Long (I32)
cnt	Počet použitých prvků (počtu řádků * počet sloupců)	Long (I32)
amax	Počet rezervovaných (alokovaných) prvků ( $\geq$ počtu řádků * počet sloupců)	Long (I32)
etype	Typ prvku matice (double, long, byte a pod.)	Long (I32)

## MX\_DIMSET – \* Nastavení dimenze matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

>uMV yMV>
>m cnt>
>n amax>
>ld E>
MX_DIMSET

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

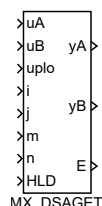
uMV	Vstupní reference na matici nebo vektor	Reference
m	Počet řádků matice	Long (I32)
ld	Vedoucí dimenze ( $\geq$ počtu řádků)	Long (I32)

## Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
n	Počet sloupců matice	Long (I32)
cnt	Počet alokovaných prvků ( $\geq$ počtu řádků * počet sloupců)	Long (I32)
E	Příznak chyby	Bool

## MX\_DSAGET – \* Uložení submatice A do matice B

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

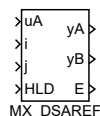
uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 ..... Celá	
	1 ..... Celá	
	2 ..... Horní	
	3 ..... Dolní	
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
HLD	Pozastavení	Bool

## Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

## MX\_DSAREF – \* Nastavení odkazu na submatici A do matice B

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uA	Vstupní odkaz na matici A	Reference
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
HLD	Pozastavení	Bool

## Parametr

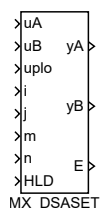
ay	Výstupní reference podmatice	⊙ [0 0]	Double (F64)
----	------------------------------	---------	--------------

## Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

## MX\_DSASET – \* Uložení matice A do submatice v B

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 ..... Celá	
	1 ..... Celá	
	2 ..... Horní	
	3 ..... Dolní	
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
HLD	Pozastavení	Bool

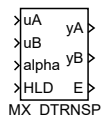
## Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

MX\_DTRNSP – \* **Transpozice obecné matice:  $B := \alpha * A^T$**

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

### Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool



**MX\_DTRNSQ** – \* **Transpozice čtvercové matice na místě:  $A := \alpha * A^T$**

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

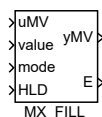
uA	Vstupní odkaz na matici A	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

Výstupy

yA	Výstupní odkaz na matici A	Reference
E	Příznak chyby	Bool

## MX\_FILL – \* Vyplnění reálné matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uMV	Vstupní reference na matici nebo vektor	Reference
value	Hodnota, kterou bude plněna matice/vektor	Double (F64)
mode	Způsob vyplnění	Long (I32)
	0 ..... Hodnota	
	1 ..... Hodnota	
	2 ..... Jedničky	
	3 ..... Hodnota na diagonálu	
	4 ..... Jednotková matice	
HLD	Pozastavení	Bool

## Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

## MX\_MAT – \* Blok pro uložení dat matice

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Parametry

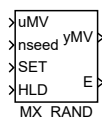
m	Počet řádků matice	↓1 ↑1000000000 ⊙10	Long (I32)
n	Počet sloupců matice	↓1 ↑1000000000 ⊙10	Long (I32)
ld	Vedoucí dimenze (>= počtu řádků)	↓0 ↑1000000000	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 . . . . . Bool		
	2 . . . . . Byte (U8)		
	3 . . . . . Short (I16)		
	4 . . . . . Long (I32)		
	5 . . . . . Word (U16)		
	6 . . . . . DWord (U32)		
	7 . . . . . Float (F32)		
	8 . . . . . Double (F64)		
	-- . . . .		
	10 . . . . Large (I64)		

## Výstup

yMat	Výstupní reference na matici	Reference
------	------------------------------	-----------

## MX\_RAND – \* Náhodně vygenerovaná matice nebo vektor

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstupy

uMV	Vstupní reference na matici nebo vektor	Reference
nseed	Násada generátoru náhodných čísel	Long (I32)
SET	Nastav na náběžnou hranu počáteční hodnotu generátoru náhodných čísel na nseed	Bool
HLD	Pozastavení	Bool

## Parametry

BIP	Příznak náhodných hodnot s oběma polaritami	Bool
scale	Násobitel náhodných hodnot	⊙1.0 Double (F64)

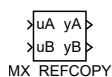
## Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

## MX\_REFCOPY – \* Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference

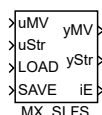
Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference

## MX\_SLFS – Ukládání a čtení matice/vektoru do souboru nebo textového retězce

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok umožňuje konvertovat matici nebo vektor do textové podoby a naopak. Matice se přivádí jako odkaz na vstup `uMV`. Výstup `yMV` odkazuje na stejnou matici, jako vstup `uMV` a je určen pro řetězení maticových bloků ve správném pořadí, jak je obvyklé u všech bloků ze skupiny `MATRIX`.

Text může být buď na vstupu `uStr` ( resp. výstupu `yStr` pro opačný směr konverze) nebo v souboru. Pokud je text v souboru, tak jeho název je text připojený na vstup `uStr`. Pro název souboru platí obvyklá pravidla systému `REXYGEN`, tj. je relativně k datadir a není dovoleno `./` pro opuštění adresáře. Pokud je vstup `uStr` nepřipojený (nebo prázdný text), použije se pro jméno souboru název bloku s celou cestou (tj. včetně jména tasku a všech subsystémů) s příponou `.dat`.

Formát matice v textovém souboru nebo v textovém vstupu a výstupu určuje parametr `format`. Podpořeno je anglické i české CSV (tj. sloupce odděleny čárkou nebo středníkem), formát JSON (vytvořený firmou Google a často používaný ve webových aplikacích) a formát používaný `MATLABem` (pro zadávání matice ve skriptech `MATLABu`).

Konverze z textové podoby do matice/vektoru nebo naopak může probíhat v každém kroku algoritmu nebo je spouštěna pomocí vstupů `LOAD` a `SAVE`. Přesný způsob je určen parametrem `mode` a je podrobně vysvětlen u popisu tohoto parametru. Pokud nastane chyba, je signalizována na výstupy `iE` a v logu. Po fatální chybě se konverze z/do matice přestane provádět. Resetování chyby se pro `mode = 1 .. 4` provede nastavením `LOAD = SAVE = off`, pro `mode = 5 .. 8` resetování fatální chyby nelze provést (musí se přepnout na `mode = 1 .. 4` a pak zpět).

Parametr `nmax` slouží k předalokování výstupního textu. Pokud je `nmax > 0`, naalokuje se při inicializaci úlohy zadané množství znaků a pokud je to nedostatečné, blok hlásí chybu. Pokud je `nmax = 0`, blok zvětšuje délku výstupního textu podle potřeby. Může se zdát nesmyslné zadávat omezení na velikost, když pak může blok selhávat, ale pokud se nezadá, může v extrémním případě dojít k použití celé dostupné paměti, která pak chybí dalším komponentám (např. operačnímu systému) a pak selže celý systém nekontrolovatelně.

## Vstupy

<code>uMV</code>	Vstupní reference na matici nebo vektor	Reference
<code>uStr</code>	Text pro konverzi na matici/vektor nebo jméno souboru	String
<code>LOAD</code>	Povolení zápisu hodnoty do matice/vektoru	Bool
<code>SAVE</code>	Povolení zápisu hodnoty do souboru/textu	Bool

## Parametry

<code>mode</code>	Režim spouštění konverze	⊙2	Long (I32)
	1 ..... aktivované úrovní, do/z souboru – data jsou konvertována ze souboru do matice při <code>LOAD=on</code> a z matice do souboru při <code>SAVE=on</code> ; pokud jsou aktivní oba signály, je to chyba a žádná akce se neprovede		
	2 ..... aktivované hranou, do/z souboru – data jsou konvertována ze souboru do matice při náběžné hraně na vstupu <code>LOAD</code> a z matice do souboru při náběžné hraně <code>SAVE</code> ; pokud jsou náběžné hrany na obou signálech, je to chyba a žádná akce se neprovede		
	3 ..... aktivované úrovní, do/z textu – data jsou konvertována ze vstupu <code>uStr</code> do matice při <code>LOAD=on</code> a z matice na výstup <code>yStr</code> při <code>SAVE=on</code> ; pokud jsou aktivní oba signály, je to chyba a žádná akce se neprovede		
	4 ..... aktivované hranou, do/z textu – data jsou konvertována ze vstupu <code>uStr</code> do matice při náběžné hraně na vstupu <code>LOAD</code> a z matice na výstup <code>yStr</code> při náběžné hraně <code>SAVE</code> ; pokud jsou náběžné hrany na obou signálech, je to chyba a žádná akce se neprovede		
	5 ..... trvale text do matice – data jsou konvertována ze vstupu <code>uStr</code> do matice v každém kroku algoritmu		
	6 ..... trvale matice do textu – data jsou konvertována z matice na výstup <code>yStr</code> v každém kroku algoritmu		
	7 ..... trvale soubor do matice – data jsou konvertována ze souboru do matice v každém kroku algoritmu		
	8 ..... trvale matice do souboru – data jsou konvertována z matice do souboru v každém kroku algoritmu		
<code>format</code>	Formát souboru/textu	⊙1	Long (I32)
	1 ..... CSV		
	2 ..... CSV(středník)		
	3 ..... JSON		
	4 ..... MATLAB		
<code>prec</code>	Počet platných cifer pro každou hodnotu	↓0 ↑20 ⊙6	Long (I32)
<code>TRN</code>	Příznak transpozice matice		Bool
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0	Long (I32)

## Výstupy

<code>yMV</code>	Výstupní reference na matici nebo vektor	Reference
<code>yStr</code>	Textová podoba matice/vektoru	String
<code>iE</code>	Kód chyby	Error



**MX\_VEC – \* Blok pro uložení dat vektoru**

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Parametry

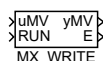
n	Počet prvků vektoru	↓1 ↑1000000000	⊙10	Long (I32)
etype	Typ prvků		⊙8	Long (I32)
	1 ..... Bool			
	2 ..... Byte (U8)			
	3 ..... Short (I16)			
	4 ..... Long (I32)			
	5 ..... Word (U16)			
	6 ..... DWord (U32)			
	7 ..... Float (F32)			
	8 ..... Double (F64)			
	-- .....			
	10 ..... Large (I64)			

## Výstup

yVec	Výstupní reference na vektor	Reference
------	------------------------------	-----------

## MX\_WRITE – \* Výpis matice/vektoru do konzole/systemého logu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uMV	Vstupní reference na matici nebo vektor	Reference
RUN	Povolení běhu algoritmu	Bool

### Parametry

Symbol	Symbolické jméno matice/vektoru pro výstup na konzoli nebo log	String
mchars	Počet znaků jednoho prvku	↓3 ↑25 ⊙8 Long (I32)
mdec	Počet desetinných míst jednoho prvku	↓0 ↑23 ⊙4 Long (I32)
mode	Závažnost výpisu	⊙3 Long (I32)
	0 ..... Žádný	
	1 ..... Žádný	
	2 ..... Podrobný	
	3 ..... Informace	
	4 ..... Varování	
	5 ..... Chyba	

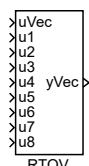
### Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

## RTOV – Vektorový multiplexer

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok slouží pro vkládání několika hodnot do vektoru (popř. matice - viz dále). Odkaz na pole se přivede na vstup `uVec`. Blok ve vektoru nastaví počet prvků určených parametrem vstup `n` počínaje prvkem s indexem vstup `offset` (první prvek vektoru má index 0). Pokud je vstup `vstup uVec` nepřipojen, hodnoty se přidávají do interního vektoru bloku s (alokovanou) velikostí vstup `nmax`. Na výstupu `yVec` je vždy odkaz/reference na vektor, do kterého se hodnoty nastavily.

Vektory (i matice) mají v systému REXYGEN vždy maximální (alokovanou) velikost a aktuální velikost. Hodnoty, které by se zapsaly na indexy větší než maximální velikost nebo menší než 0 se nezapišou. Hodnoty, které vedou na indexy mimo aktuálně platnou oblast (ale do alokované/maximální oblasti) se nastaví a navíc se nastaví i aktuální velikost tak, aby všechny nastavené hodnoty byly platné (tj. aktuální velikost je vždy nejméně poslední zapsaný index - vždy samozřejmě omezený na alokovanou velikost).

Pokud se na vstup `uVec` přivede matice, tak blok funguje také, ale hodnota parametru vstup `offset` je chápána jako linearizovaný index, přičemž matice je uložena po sloupcích a následující prvky jsou také následující ve sloupci. Pokud v případě matice dojde ke zvětšení oblasti platných dat (aktuální velikosti), může u matice dojít k nekonzistenci (počet prvků není násobkem počtu řádek), proto se v případě matice doporučuje vždy mít nastavenou potřebnou velikost předem.

Parametr `etype` určuje typ vstupních hodnot `u1 .. u8` a typ hodnot ve vnitřním vektoru. Pokud vektor (nebo matice) přivedená na vstup `uVec` má jiný typ, provede se konverze.

**POZOR:** Do verze 2.50.10 je vnitřní vektor řádka, v pozdějších verzích (od 2.51.0.9525) je vnitřní vektor sloupec. Řádkové vektory způsobovaly obtížně řešitelné komplikace v blocích, které pracují s maticemi (a vektor chápou jako matici s jedním sloupcem). Pokud se výstup používá v bloku, který přesně kontroluje rozměry matice, doporučuje se rozměry definovat blokem `MX_MAT` nebo `CNA` a takto definovanou matici přivést na vstup `uVec`.

## Vstupy

<code>uVec</code>	Vektorový signál	Reference
<code>u1</code>	Analogový vstupní signál	Double (F64)
<code>u2</code>	Analogový vstupní signál	Double (F64)
<code>u3</code>	Analogový vstupní signál	Double (F64)
<code>u4</code>	Analogový vstupní signál	Double (F64)
<code>u5</code>	Analogový vstupní signál	Double (F64)
<code>u6</code>	Analogový vstupní signál	Double (F64)
<code>u7</code>	Analogový vstupní signál	Double (F64)
<code>u8</code>	Analogový vstupní signál	Double (F64)

## Parametry

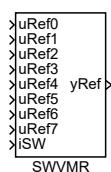
<code>nmax</code>	Alokovaná velikost vektoru	↓0 ⊙8	Long (I32)
<code>offset</code>	Index prvního vstupu ve vektoru	↓0	Long (I32)
<code>n</code>	Počet použitých vstupů	↓0 ↑8 ⊙8	Long (I32)
<code>etype</code>	Typ prvků	⊙8	Long (I32)
	1 ..... Bool		
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	7 ..... Float (F32)		
	8 ..... Double (F64)		
	-- .....		
	10 ..... Large (I64)		

## Výstup

<code>yVec</code>	Vektorový signál	Reference
-------------------	------------------	-----------

## SWVMR – \* Přepínač vektorového/maticového/odkazovacího signálu

Symbol bloku

Licence: [STANDARD](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uRef0	Vektorový signál	Reference
uRef1	Vektorový signál	Reference
uRef2	Vektorový signál	Reference
uRef3	Vektorový signál	Reference
uRef4	Vektorový signál	Reference
uRef5	Vektorový signál	Reference
uRef6	Vektorový signál	Reference
uRef7	Vektorový signál	Reference
iSW	Selektor aktivního signálu	Long (I32)

### Výstup

yRef	Vektorový signál	Reference
------	------------------	-----------

## VTOR – \* Vektorový demultiplexer

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

## Vstup

uVec	Vektorový signál	Reference
------	------------------	-----------

## Parametry

n	Počet použitých výstupů	↓0 ↑8 ⊙8	Long (I32)
offset	Index prvního výstupu	↓0	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 ..... Bool		
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	7 ..... Float (F32)		
	8 ..... Double (F64)		
	-- ..... 10 .... Large (I64)		

## Výstupy

y1	Analogový výstupní signál	Double (F64)
y2	Analogový výstupní signál	Double (F64)
y3	Analogový výstupní signál	Double (F64)
y4	Analogový výstupní signál	Double (F64)
y5	Analogový výstupní signál	Double (F64)
y6	Analogový výstupní signál	Double (F64)
y7	Analogový výstupní signál	Double (F64)
y8	Analogový výstupní signál	Double (F64)

## Kapitola 15

# OPTIM – Bloky pro optimalizaci

### Obsah

---

<a href="#">QCEDPOPT</a> – * <b>Optimalizace tzv. Economic Dispatch problému</b> . . .	<b>456</b>
<a href="#">QP_MPC2QP</a> – * <b>Převod úlohy prediktivního řízení na kvadratické programování</b> . . . . .	<b>458</b>
<a href="#">QP_OASES</a> – * <b>Kvadratické programování pomocí metody aktivní množiny</b> . . . . .	<b>460</b>
<a href="#">QP_UPDATE</a> – * <b>Aktualizace matic/vektorů kvadratického programování</b> . . . . .	<b>463</b>

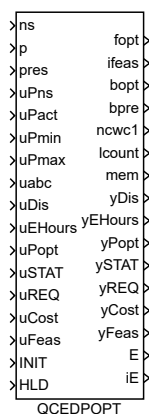
---

Knihovna OPTIM je přizpůsobena pro optimalizační algoritmy a procesy. Zahrnuje [QCEDPOPT](#) pro optimalizace tzv. Economic Dispatch problémů, poskytující pokročilé nástroje pro řešení složitých optimalizačních problémů. Knihovna také obsahuje bloky jako [QP\\_MPC2QP](#) a [QP\\_OASES](#) pro kvadratické programování, které jsou nezbytné ve scénářích Model Predictive Control (MPC). Navíc je k dispozici [QP\\_UPDATE](#) pro aktualizaci parametrů kvadratického programu. Tato knihovna je zvláště užitečná v systémech vyžadujících řešení optimalizace na vysoké úrovni, jako jsou pokročilé řídicí a rozhodovací algoritmy.

## QCEDPOPT – \* Optimalizace tzv. Economic Dispatch problému

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

ns	Horizont startování zdrojů	↓1 ↑10	Long (I32)
p	Výkon k rozdělení mezi zdroje	↓0.0 ↑1000000.0	Double (F64)
pres	Rezerva výkonu	↓0.0 ↑1000000.0	Double (F64)
uPns	Vstupní odkaz na vektor predikce rozdělovaného výkonu		Reference
uPact	Vstupní odkaz na vektor výkonů aktivních (běžících) zdrojů		Reference
uPmin	Vstupní odkaz na vektor minimálních výkonů zdrojů		Reference
uPmax	Vstupní odkaz na vektor maximálních výkonů zdrojů		Reference
uabc	Vstupní odkaz na matici cenových koeficientů		Reference
uDis	Vstupní odkaz na celočíselný vektor důvodů deaktivace zdrojů		Reference
uEHours	Vstupní odkaz na vektor motohodin		Reference
uPopt	Vstupní odkaz na alokovaný vektor pro optimální rozdělení výkonu		Reference
uSTAT	Vstupní odkaz na vektor typu bool indikující běžící stav zdrojů		Reference
uREQ	Vstupní odkaz na alokovaný vektor typu bool pro požadavky na běh zdrojů		Reference
uCost	Vstupní odkaz na nepovinný vektor hodnot kritéria optimality		Reference
uFeas	Vstupní odkaz na nepovinný vektor typu byte pro typy řešitelnosti		Reference



INIT	Je-li TRUE, pak reinitializuj optimalizaci, jinak optimalizuj v reálném čase	Bool
HLD	Pozastavení	Bool

## Parametr

modesh	Režim heuristiky pro startování	↓0 ↑3	Long (I32)
mchng	Maximálně dovolený počet změn běhu m za periodu		Long (I32)
		↓0 ↑10 ⊙3	
nmax	Maximální hodnota n, nmax ≤ 10	↓0 ↑10 ⊙10	Long (I32)
nsmax	Maximální hodnota ns, nsmax ≤ 10	↓0 ↑10 ⊙10	Long (I32)
logflags	Příznaky výpisů do systémového logu	↓0 ↑63 ⊙3	Long (I32)
ts	Perioda vzorkování. Pro ts = 0 je použita perioda vzorkování úlohy obsahující tento blok	↓0.0 ↑1000000.0	Double (F64)
tspre	Doba pro předstart výkonového zdroje	↓0.0 ↑1000000.0	Double (F64)
svNF1	Náhradní hodnota v yCost pro neřešitelné úlohy v prvním kroku optimalizace	⊙-1.0	Double (F64)
svNFns	Náhradní hodnota v yCost pro neřešitelné úlohy v celém horizontu ns	⊙-1.0	Double (F64)

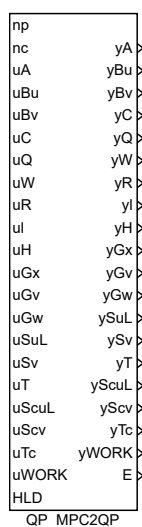
## Výstup

fopt	Optimální hodnota kritéria pro horizont ns	Double (F64)
ifeas	Typ řešitelnosti vybraného řešení	Long (I32)
bopt	Bitová kombinace běžících zdrojů pro fopt	Long (I32)
bpre	Bitová kombinace zdrojů, které mají být předstartovány	Long (I32)
ncwc1	Počet kombinací v nejhorším případě závislý na n zdrojích a parametru mchng	Long (I32)
lcount	Počet stavů projitých při prohledávání stavového prostoru	Large (I64)
mem	Velikost paměti v bajtech interně alokovaných pracovních polí	Long (I32)
yDis	Výstupní odkaz na celočíselný vektor důvodů deaktivace zdrojů	Reference
yEHours	Výstupní odkaz na vektor motohodin	Reference
yPopt	Výstupní odkaz na alokovaný vektor pro optimální rozdělení výkonu	Reference
ySTAT	Výstupní odkaz na vektor typu bool indikující běžící stav zdrojů	Reference
yREQ	Výstupní odkaz na alokovaný vektor typu bool pro požadavky na běh zdrojů	Reference
yCost	Výstupní odkaz na nepovinný vektor hodnot kritéria optimality	Reference
yFeas	Výstupní odkaz na nepovinný vektor typu byte pro typy řešitelnosti	Reference
E	Příznak chyby	Bool
iE	Kód chyby	Long (I32)

## QP\_MPC2QP – \* Převod úlohy prediktivního řízení na kvadratické programování

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

np	Horizont predikce	↓1 ↑1000000	Long (I32)
nc	Horizont řízení	↓1 ↑1000000	Long (I32)
uA	Vstupní odkaz na matici systému A		Reference
uBu	Vstupní odkaz na matici Bu vektoru řízení u		Reference
uBv	Vstupní odkaz na matici Bv vektoru poruchy v		Reference
uC	Vstupní odkaz na výstupní matici C		Reference
uQ	Vstupní odkaz na symetrickou matici Q v kritériu optimality		Reference
uW	Vstupní odkaz na vektor W v kritériu optimality		Reference
uR	Vstupní odkaz na symetrickou matici R v kritériu optimality		Reference
uI	Vstupní odkaz na celočíselný vektor indexů I		Reference
uH	Vstupní odkaz na Hessovu matici H		Reference
uGx	Vstupní odkaz na složku gradientního vektoru G odpovídající stavu x		Reference
uGv	Vstupní odkaz na složku gradientního vektoru G odpovídající poruše v		Reference

uGw	Vstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
uSuL	Vstupní odkaz na pracovní matici Su*L	Reference
uSv	Vstupní odkaz na pracovní matici Sv	Reference
uT	Vstupní odkaz na pracovní matici T	Reference
uScuL	Vstupní odkaz na pracovní matici Scu*L	Reference
uScv	Vstupní odkaz na pracovní matici Scv	Reference
uTc	Vstupní odkaz na pracovní matici Tc	Reference
uWORK	Vstupní odkaz na pracovní matici WORK	Reference
HLD	Pozastavení	Bool

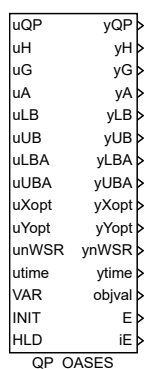
## Výstupy

yA	Výstupní odkaz na matici systému A	Reference
yBu	Výstupní odkaz na matici Bu vektoru řízení u	Reference
yBv	Výstupní odkaz na matici Bv vektoru poruchy v	Reference
yC	Výstupní odkaz na výstupní matici C	Reference
yQ	Výstupní odkaz na symetrickou matici Q v kritériu optimality	Reference
yW	Výstupní odkaz na vektor W v kritériu optimality	Reference
yR	Výstupní odkaz na symetrickou matici R v kritériu optimality	Reference
yI	Výstupní odkaz na celočíselný vektor indexů I	Reference
yH	Výstupní odkaz na Hessovu matici H	Reference
yGx	Výstupní odkaz na složku gradientního vektoru G odpovídající stavu x	Reference
yGv	Výstupní odkaz na složku gradientního vektoru G odpovídající poruše v	Reference
yGw	Výstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
ySuL	Výstupní odkaz na pracovní matici Su*L	Reference
ySv	Výstupní odkaz na pracovní matici Sv	Reference
yT	Výstupní odkaz na pracovní matici T	Reference
yScuL	Výstupní odkaz na pracovní matici Scu*L	Reference
yScv	Výstupní odkaz na pracovní matici Scv	Reference
yTc	Výstupní odkaz na pracovní matici Tc	Reference
yWORK	Výstupní odkaz na pracovní matici WORK	Reference
E	Příznak chyby	Bool

## QP\_OASES – \* Kvadratické programování pomocí metody aktivní množiny

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

uQP	Vstupní odkaz na úlohu kvadratického programování	Reference
uH	Vstupní odkaz na Hessovu matici H	Reference
uG	Vstupní odkaz na gradientní vektor G	Reference
uA	Vstupní odkaz na matici omezení A	Reference
uLB	Vstupní odkaz na vektor dolních mezí LB	Reference
uUB	Vstupní odkaz na vektor horních mezí UB	Reference
uLBA	Vstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
uUBA	Vstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
uXopt	Vstupní odkaz na vektor prvotního optimálního řešení	Reference
uYopt	Vstupní odkaz na vektor duálního optimálního řešení	Reference
unWSR	Maximální počet výpočtů pracovní množiny během inicializace	Long (I32)
utime	Maximální dovolený čas CPU v sekundách pro celou inicializaci	Double (F64)
VAR	Příznak časové variantnosti matic H a A	Bool
INIT	Při každém spuštění bloku volá funkci <code>init()</code> místo <code>hotstart()</code>	Bool
HLD	Pozastavení	Bool

## Parametry

<code>hessianType</code>	Typ Hessovy matice	Long (I32)
<code>printLevel</code>	Úroveň výpisů	Long (I32)
<code>enableRamping</code>	Povolit tzv. rampování	Bool
<code>enableFarBounds</code>	Povolit použití tzv. dalekých mezí	Bool
<code>enableFlippingBounds</code>	Povolit používání tzv. převrácených mezí (flipping bounds)	Bool
<code>enableRegularisation</code>	Povolit regularizaci semidefinitní Hessovy matice	Bool
<code>enableFullLITests</code>	Povolit použití tvrdších testů lineární nezávislosti	Bool
<code>enableNZCTests</code>	Povolit testy nenulového zakřivení	Bool
<code>enableDriftCorrection</code>	Korekce frekvence nebo driftu (0 = off)	Long (I32)
<code>enableCholeskyRefact</code>	Frekvence plné refaktorizace projektovaného Hessiánu (0 = off)	Long (I32)
<code>enableEqualities</code>	Rovnosti se vždy považují za aktivní omezení	Bool
<code>terminationTolerance</code>	Ukončovací tolerance	Double (F64)
<code>boundTolerance</code>	Pokud se horní a spodní meze liší méně než o tuto toleranci, jsou považovány za shodné, tj. za podmínku s rovností	Double (F64)
<code>boundRelaxation</code>	Počáteční relaxace mezí pro startovací homotopii a počáteční hodnota dalekých mezí	Double (F64)
<code>epsNum</code>	Tolerance čitatele pro poměrové testy	Double (F64)
<code>epsDen</code>	Tolerance jmenovatele pro poměrové testy	Double (F64)
<code>maxPrimalJump</code>	Maximálně povolený skok v primálních proměnných pro testy lineární nezávislosti	Double (F64)
<code>maxDualJump</code>	Maximálně povolený skok v duálních proměnných pro testy lineární nezávislosti	Double (F64)
<code>initialRamping</code>	Počáteční hodnota pro strategii rampování	Double (F64)
<code>finalRamping</code>	Koncová hodnota pro strategii rampování	Double (F64)
<code>initialFarBounds</code>	Počáteční velikost dalekých mezí	Double (F64)
<code>growFarBounds</code>	Faktor zvětšování dalekých mezí	Double (F64)
<code>initialStatusBounds</code>	Počáteční stav mezí v první iteraci	Long (I32)
<code>epsFlipping</code>	Tolerance kvadrátu prvku na Choleského diagonále, která aktivuje tzv. převrácené meze (flipping bounds)	Double (F64)
<code>numRegularisationSteps</code>	Maximální počet po sobě jdoucích regularizačních kroků	Long (I32)
<code>epsRegularisation</code>	Měřítkovací faktor identické matice použitý pro regularizaci Hessiánu	Double (F64)
<code>numRefinementSteps</code>	Maximální počet kroků pro iterační upřesnění	Long (I32)
<code>epsIterRef</code>	Tolerance předčasného ukončení pro iterativní upřesnění	Double (F64)
<code>epsLITests</code>	Tolerance pro testy lineární nezávislosti	Double (F64)

<code>epsNZCTests</code>	Tolerance pro testy nenulového zakřivení	Double (F64)
--------------------------	--	--------------

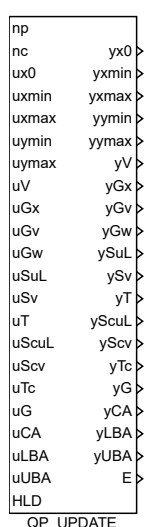
## Výstupy

<code>yQP</code>	Výstupní odkaz na úlohu kvadratického programování	Reference
<code>yH</code>	Výstupní odkaz na Hessovu matici H	Reference
<code>yG</code>	Výstupní odkaz na gradientní vektor G	Reference
<code>yA</code>	Výstupní odkaz na matici omezení A	Reference
<code>yLB</code>	Výstupní odkaz na vektor dolních mezí LB	Reference
<code>yUB</code>	Výstupní odkaz na vektor horních mezí UB	Reference
<code>yLBA</code>	Výstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
<code>yUBA</code>	Výstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
<code>yXopt</code>	Výstupní odkaz na vektor prvotního optimálního řešení	Reference
<code>yYopt</code>	Výstupní odkaz na vektor duálního optimálního řešení	Reference
<code>ynWSR</code>	Skutečný počet výpočtů pracovní množiny během inicializace	Long (I32)
<code>ytime</code>	Spotřebovaný čas CPU v sekundách pro celou inicializaci	Double (F64)
<code>objval</code>	Optimální hodnota účelové funkce	Double (F64)
<code>E</code>	Příznak chyby	Bool
<code>iE</code>	Kód chyby	Long (I32)

## QP\_UPDATE – \* Aktualizace matic/vektorů kvadratického programování

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

np	Horizont predikce	↓1 ↑1000000	Long (I32)
nc	Horizont řízení	↓1 ↑1000000	Long (I32)
ux0	Vstupní odkaz na vektor počátečních podmínek x0 vektoru stavu x		Reference
uxmin	Vstupní odkaz na vektor spodních mezí prvků stavového vektoru		Reference
uxmax	Vstupní odkaz na vektor horních mezí prvků stavového vektoru		Reference
uymin	Vstupní odkaz na vektor spodních mezí výstupních nerovností		Reference
uymax	Vstupní odkaz na vektor horních mezí výstupních nerovností		Reference
uV	Vstupní odkaz na vektor predikovaných poruch		Reference
uGx	Vstupní odkaz na složku gradientního vektoru G odpovídající stavu x		Reference
uGv	Vstupní odkaz na složku gradientního vektoru G odpovídající poruše v		Reference

uGw	Vstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
uSuL	Vstupní odkaz na pracovní matici Su*L	Reference
uSv	Vstupní odkaz na pracovní matici Sv	Reference
uT	Vstupní odkaz na pracovní matici T	Reference
uScuL	Vstupní odkaz na pracovní matici Scu*L	Reference
uScv	Vstupní odkaz na pracovní matici Scv	Reference
uTc	Vstupní odkaz na pracovní matici Tc	Reference
uG	Vstupní odkaz na gradientní vektor G	Reference
uCA	Vstupní odkaz na matici omezení kvadratického programování CA	Reference
uLBA	Vstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
uUBA	Vstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
HLD	Pozastavení	Bool

## Výstupy

yx0	Výstupní odkaz na vektor počátečních podmínek x0 vektoru stavu x	Reference
yxmin	Výstupní odkaz na vektor spodních mezí prvků stavového vektoru	Reference
yxmax	Výstupní odkaz na vektor horních mezí prvků stavového vektoru	Reference
yymin	Výstupní odkaz na vektor spodních mezí výstupních nerovností	Reference
yymax	Výstupní odkaz na vektor horních mezí výstupních nerovností	Reference
yV	Výstupní odkaz na vektor predikovaných poruch	Reference
yGx	Výstupní odkaz na složku gradientního vektoru G odpovídající stavu x	Reference
yGv	Výstupní odkaz na složku gradientního vektoru G odpovídající poruše v	Reference
yGw	Výstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
ySuL	Výstupní odkaz na pracovní matici Su*L	Reference
ySv	Výstupní odkaz na pracovní matici Sv	Reference
yT	Výstupní odkaz na pracovní matici T	Reference
yScuL	Výstupní odkaz na pracovní matici Scu*L	Reference
yScv	Výstupní odkaz na pracovní matici Scv	Reference
yTc	Výstupní odkaz na pracovní matici Tc	Reference
yG	Výstupní odkaz na gradientní vektor G	Reference
yCA	Výstupní odkaz na matici omezení kvadratického programování CA	Reference



yLBA	Výstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
yUBA	Výstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
E	Příznak chyby	Bool



## Kapitola 16

# SPEC – Speciální bloky

### Obsah

---

EPC – Blok pro spouštění externích programů . . . . .	468
HTTP – * Blok pro generování požadavků HTTP GET a POST (zastaralý) . . . . .	471
HTTP2 – * Blok pro generování HTTP požadavků . . . . .	473
RDC – Komunikační blok . . . . .	475
REXLANG – Volně programovatelný blok . . . . .	480
SMTP – * Blok pro odesílání e-mailových oznámení přes SMTP .	501
STEAM – Přepočet vlastností páry . . . . .	503
UART – * Komunikační blok pro UART . . . . .	505

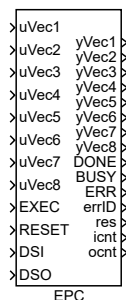
---

Knihovna SPEC zahrnuje rozmanitou sadu funkčních bloků navržených pro integraci široké škály funkcí do automatizace, řídicích systémů a komunikačních protokolů. Od usnadnění přesných termodynamických výpočtů s blokem **STEAM** po umožnění bezproblémové datové komunikace prostřednictvím **UART** a **SMTP** slouží knihovna jako komplexní sada nástrojů pro inženýry a vývojáře. Obsahuje specializované bloky pro spouštění externích programů (**EPC**), zpracování webových požadavků (**HTTP2**). Navíc nabízí unikátní vstupně-výstupní řešení (**RDC**) a všestranné programovací prostředí s **REXLANG**.

## EPC – Blok pro spouštění externích programů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Tento blok v okamžiku náběžné hrany (**off**→**on**) na vstupu **EXEC** spustí externí program, jehož název a parametry jsou uvedeny v parametru **cmd**. Zápis příkazu je naprosto shodný, jako by se psal na příkazovou řádku operačního systému.

Externímu programu lze předat hodnoty ze systému **REXYGEN** pomocí souborů. Formát těchto souborů určuje parametr **format**. V současnosti podporované formáty jsou všechny textové a velice jednoduché, takže je snadno načíst do téměř libovolného programu. Například do **MATLABu** se soubor načte příkazem

```
hodnoty=load('-ASCII', 'epc_uVec1');
```

do **SCILABu** příkazem

```
hodnoty=read('/tmp/epc_uVec1', -1, 32);
```

Název souboru, počet sloupců, jméno matice atd. je samozřejmě potřeba zvolit podle konkrétní aplikace. Hodnoty z externího programu zpět do systému **REXYGEN** se předávají analogickým způsobem (tj. opět pomocí souborů ve stejném formátu).

Blok rozlišuje dva režimy. V základním režimu je v okamžiku náběžné hrany na vstupu **EXEC** nejprve načtena aktuální hodnota na vstupech, uložena do souboru (vždy hodnoty z *i*-tého vstupního vektoru **uVec<i>** do *i*-tého souboru v parametru **ifns**). Ve vzorkovacím režimu jsou data ze vstupních vektorů ukládána do souborů v každé periodě algoritmu. V obou případech platí, že hodnoty vstupů z jednoho časového okamžiku jsou v jedné řádce souboru.

Analogicky jsou kopírována data z výstupních souborů na výstupy bloku (vždy jedna řádka z *i*-tého souboru v parametru **ofns** do *i*-tého výstupního vektoru **yVec<i>**).

Čísla vstupů, které pracují ve vzorkovacím režimu jsou uvedena v parametru **sl** (jednotlivá čísla se oddělují čárkou). Výstupy jsou vždy ve vzorkovacím režimu, přičemž pokud v souboru nejsou další data (řádky), je ponechána předchozí hodnota. Kopírování vstupů do souboru je možné zablokovat (pozastavit) vstupem **DSI**; kopírování dat ze souborů na výstupy bloku je možné zablokovat (pozastavit) vstupem **DSO**.

Vektorové vstupy a výstupy bloku umožňují jednoduše uložit do jednoho souboru více hodnot (v každém kroku). Pro převod více jednoduchých signálů na vektor slouží blok **RTOV**. Tyto bloky lze řetěžit, takže je možné vytvořit vektor téměř libovolné velikosti. Obdobně pro převod vektoru na jednoduché signály slouží blok **VTOR**, přičemž jeho vícenásobným použitím je možné získat hodnoty z libovolně velkého vektoru.

## Vstupy

<code>uVec1..uVec8</code>	Vstupní vektorové signály	Reference
<code>EXEC</code>	Náběžná hrana spouští externí program	Bool
<code>RESET</code>	Reset bloku (smaže vstupní i výstupní soubory a zastaví externí program)	Bool
<code>DSI</code>	Pozastavení vzorkování na vstupech	Bool
<code>DSO</code>	Pozastavení vzorkování na výstupech	Bool

## Výstupy

<code>yVec1..yVec8</code>	Výstupní vektorové signály	Reference
<code>DONE</code>	Příznak skončení externího programu	Bool
<code>BUSY</code>	Příznak běhu externího programu	Bool
<code>ERR</code>	Příznak chyby	Bool
<code>errID</code>	Kód chyby i ..... obecná chyba systému REXYGEN	Error
<code>res</code>	Návratový kód externího programu	Long (I32)
<code>icnt</code>	Aktuální číslo vzorku na vstupech	Long (I32)
<code>ocnt</code>	Aktuální číslo vzorku na výstupech	Long (I32)

## Parametry

<code>cmd</code>	Externí program	String
<code>ifns</code>	Vstupní soubory (oddělené středníkem) ⊙ <code>epc_uVec1;epc_uVec2</code>	String
<code>ofns</code>	Výstupní soubory (oddělené středníkem) ⊙ <code>epc_yVec1;epc_yVec2</code>	String
<code>sl</code>	Seznam čísel vzorkovacích vstupů. Zadává se ve tvaru např. 1,3..5,8. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101. ↓0 ↑255 ⊙85	Long (I32)
<code>ifm</code>	Maximální počet vzorků ve vstupním souboru	⊙10000 Long (I32)
<code>format</code>	Formát vstupních a výstupních souborů 1 ..... textový (pouze hodnoty oddělené mezerou) 2 ..... CSV (desetinná tečka a čárky) 3 ..... CSV (desetinná čárka a středníky)	⊙1 Long (I32)
<code>nmax</code>	Maximální délka výstupních vektorů	↓2 ↑1000000 ⊙100 Long (I32)

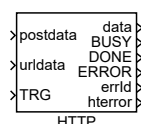
## Poznámky

- Spuštěný skript má stejnou prioritu, jako task, který ji spustil. Ta je (implicitně) hodně velká (v některých případech dokonce vyšší, než tasky zpracovávající interrupty v kernelu operačního systému). Pokud je toto nežádoucí (tj. zejména pokud externí skript trvá dlouho), je potřeba prioritu externího programu snížit. V Linuxu se to provede tak, že příkaz napíšeme ve tvaru `chrt -o 0 extprg.sh`, kde `extprg.sh` je skript/program, který chceme spustit.
- Z implementačních důvodů je počet výstupních signálů omezen a je určen parametrem `nmax`. Parametr umožňuje zadat i hodně velká čísla, ale pro některé platformy nemusí být k dispozici dostatek paměti. Volte proto vždy co nejmenší číslo, které (s malou rezervou) dostačuje aplikaci.
- Jména souborů je potřeba psát tak, jak to vyžaduje použitý operační systém na cílové platformě. Nicméně pro vyhnutí se nečekaným potížím je doporučeno používat v názvu souboru jen písmena anglické abecedy, číslice a podtržítka. Také pozor na velikost písmen (Linux ji rozlišuje). Dále je potřeba zvážit, zda zadávat soubory s absolutní cestou nebo relativně k aktuálnímu adresáři. Zejména při vývoji aplikace se může aktuální adresář různě měnit a externí aplikace soubory nenajde.
- Z implementačních důvodů blok vytváří ještě kopie souborů uvedených v parametrech `ifns` a `ofns`. Tyto kopie mají v názvu navíc znak podtržítka.
- Parametry `ifns` a `ofns` určují umístění souborů. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému REXYGEN na cílovém zařízení. Z důvodu výkonnosti je vhodné v tomto adresáři vytvořit symbolický link na souborový systém v RAM paměti. Na druhou stranu, pro dlouhé řady je výhodné mít soubor na disku, protože blok v případě výpadku řídicího systému po jeho opětovném spuštění naváže na předchozí data.
- Pro volání některých funkcí operačního systému lze použít i blok [OSCALL](#).

## HTTP – \* Blok pro generování požadavků HTTP GET a POST (zastaralý)

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

postdata	Data vložená do HTTP požadavku	String
urldata	Data připojená k URL adrese	String
TRG	Spuštění zvolené akce	Bool

### Parametry

url	URL adresa pro odeslání HTTP požadavku	String
method	Typ HTTP požadavku 1 ..... GET 2 ..... POST	⊙1 Long (I32)
user	Uživatelské jméno	String
password	Heslo	String
certificate	Certifikát pro autentifikaci	String
VERIFY	Povolení verifikace serveru (platnost certifikátu)	Bool
postmime	Typ kódování pro požadavek POST	⊙application/json String
acceptmime	Typ kódování pro požadavek GET	⊙application/json String
timeout	Povolená doba pro dokončení operace	⊙5.0 Double (F64)
BLOCKING	Čekání na dokončení operace	Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)
postmax	postmax	↓128 ↑65520 ⊙256 Long (I32)
datamax	Alokovaná paměť pro odpověď na příkaz HTTP	↓128 ↑10000000 ⊙1024 Long (I32)

### Výstupy

data	Data z odpovědi	String
BUSY	Odeslání HTTP požadavku	Bool

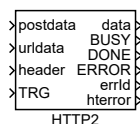
<code>DONE</code>	HTTP požadavek byl zpracován	<code>Bool</code>
<code>ERROR</code>	Příznak chyby	<code>Bool</code>
<code>errId</code>	Kód chyby	<code>Error</code>
<code>hterror</code>	HTTP odpověď	<code>Long (I32)</code>



## HTTP2 — \* Blok pro generování HTTP požadavků

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

postdata	Data vložená do HTTP požadavku	String
urldata	Data připojená k URL adrese	String
header	Uživatelské položky hlavičky	String
TRG	Spuštění zvolené akce	Bool

### Parametry

url	URL adresa pro odeslání HTTP požadavku		String
method	Typ HTTP požadavku	⊙1	Long (I32)
	1 ..... GET		
	2 ..... POST		
	3 ..... PUT		
	4 ..... DELETE		
	5 ..... HEAD		
	6 ..... TRACE		
	7 ..... PATCH		
	8 ..... OPTIONS		
	9 ..... CONNECT		
user	Uživatelské jméno		String
password	Heslo		String
certificate	Certifikát pro autentifikaci		String
VERIFY	Povolení verifikace serveru (platnost certifikátu)		Bool
postmime	Typ kódování pro požadavek POST	⊙application/json	String
acceptmime	Typ kódování pro požadavek GET	⊙application/json	String
timeout	Povolená doba pro dokončení operace	⊙5.0	Double (F64)
BLOCKING	Čekání na dokončení operace		Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
postmax	postmax	↓128 ↑65520 ⊙4096	Long (I32)

<code>datamax</code>	Alokovaná paměť pro odpověď na příkaz HTTP	Long (I32)
	↓128 ↑10000000 ⊕64000	

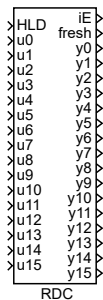
## Výstupy

<code>data</code>	Data z odpovědi	String
<code>BUSY</code>	Odesílání HTTP požadavku	Bool
<code>DONE</code>	HTTP požadavek byl zpracován	Bool
<code>ERROR</code>	Příznak chyby	Bool
<code>errId</code>	Kód chyby	Error
<code>hterror</code>	HTTP odpověď	Long (I32)

## RDC – Komunikační blok

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Tento blok je speciální vstupně-výstupní blok. Hodnoty se předávají mezi dvěma bloky se stejným číslem, ale na různých počítačích (popřípadě na stejném počítači mezi dvěma Simulinky nebo Simulinkem a systémem REXYGEN). Hodnoty se předávají UDP/IP protokolem. Tento protokol je stejně rozšířený jako známější TCP/IP (tj. funguje na všech lokálních sítích LAN i na linkách sítě Internet). Algoritmus v každém kroku provádí následující operace:

- Otestuje vstup HLD. Pokud je HLD = on, činnost bloku končí.
- Má-li parametr **period** kladnou hodnotu, zjistí rozdíl mezi systémovým časem a časem posledního vyslání paketu. Pokud je tato doba menší než hodnota **period**, činnost bloku končí. (Pokud je hodnota parametru **period** menší nebo rovna nule, testování rozdílu času se neprovádí.)
- Vytvoří paket, který obsahuje číslo bloku, tzv. číslo **invoke** (pořadové číslo paketu), hodnoty **u0** až **u15**. Všechny hodnoty se do paketu ukládají ve standardně užívaném pořadí (tzv. network byte order), takže aplikace může běžet na libovolném počítači/procesoru.
- Odešle paket na zadanou IP adresu a port.
- Zvětší o 1 číslo **invoke**.
- Otestuje, jestli přišel nějaký paket.
- Pokud ano, otestuje, zda je paket v pořádku (souhlasí velikost, číslo bloku, číslo **invoke**).
- Pokud je paket v pořádku, nastaví výstupy **y0** až **y15** na hodnoty z přijatého paketu.

- Nastaví výstup **iE** (pokud došlo k nějaké chybě) a výstup **fresh**.

Z uvedeného popisu je zřejmé, že dvojice bloků (se stejným číslem, ale každý na jiném počítači) periodicky přenáší 16 hodnot v každém směru. Vždy se přenesou  $u(i)$  z jednoho bloku na  $y(i)$  druhého bloku. Protože protokol UDP/IP (na rozdíl od TCP/IP) nemá mechanismus pro ošetření ztráty ani duplicity paketu, musí se to zajistit v algoritmu. K ošetření ztráty slouží mechanismus čísla **invoke**. To je stavová proměnná, která se zvětší o 1 při každém odeslaném paketu. Protože blok si pamatuje **invoke** číslo minulého přijatého paketu, pozná, k čemu došlo, a podle toho reaguje – pakety s číslem **invoke** menším než číslo **invoke** posledního přijatého paketu odmítá. Protože se však po ukončení a opětovném spuštění programu číslo **invoke** vynuluje, algoritmus přesto přijme paket s číslem menším než číslo posledního paketu, pokud je rozdíl velký (větší než 10). Z implementačních důvodů musí mít všechny bloky v jedné aplikaci stejný **local port** a v jedné aplikaci může být nejvýše 64 bloků **RDC**. Pokud by na jednom počítači běžely dva programy, které používají blok **RDC**, musí být parametr **local port** v každé aplikaci jiný.

## Vstupy

<b>HLD</b>	Vstup pozastavující činnost bloku. Pokud je <b>HLD = on</b> , blok <b>Boo1</b> nevysílá ani nepřijímá žádné pakety.
<b>u0..u15</b>	Hodnoty, které se předávají/zapisují na hodnoty <b>y0</b> až <b>y15</b> spolupracujícího bloku <b>RDC</b> <b>Double (F64)</b>

## Výstupy

<b>iE</b>	Zobrazuje kód poslední chyby. Použitá čísla jsou v následující tabulce: <b>Long (I32)</b>
	0 ..... Bez chyby
	<i>Trvalé chyby, vznikají v inicializační části bloku, systém je nedokáže sám opravit (&lt; 0)</i>
	-1 .... překročen maximální počet bloků (z interních důvodů je omezen počet bloků v jednom programu na 64)
	-2 .... blok má jiný lokální port (z interních důvodů musí mít všechny bloky v jednom programu (jedné úloze) stejný parametr <b>lport</b> )
	-3 .... nelze otevřít socket (protokol UDP/IP je nedostupný)
	-4 .... nelze přiřadit lokální port (port je pravděpodobně obsazen jinou službou nebo programem)
	-5 .... nelze nastavit tzv. neblokující mód socketu (blok <b>RDC</b> tento mód využívá a v případě chyby nemůže správně fungovat)
	-10 ... chyba v inicializaci socketové knihovny
	-11 ... chyba v inicializaci socketové knihovny
	-12 ... chyba v inicializaci socketové knihovny

*Přechodné chyby, mohou vzniknout ve kterémkoliv průchodu kódu, systém je dokáže sám opravit (> 0)*

1 .....	proběhla inicializace bloku, ale ještě nebyl přijat žádný platný paket s hodnotami	
2 .....	přijat chybný paket (chybná délka – buď došlo k chybě při přenosu a data jsou ztracena nebo může jít o konflikt s jinou službou/programem)	
4 .....	chyba při příjmu paketu (chybu hlásí socketová knihovna)	
8 .....	chyba při odeslání paketu (chybu hlásí socketová knihovna)	
fresh	Udává počet sekund od přijetí posledního paketu. Má význam pro detekci chyby protilehlého bloku.	Double (F64)
y0..y15	Signál přijatý ze vzdáleného bloku RDC – hodnoty naposledy přijatého paketu	Double (F64)

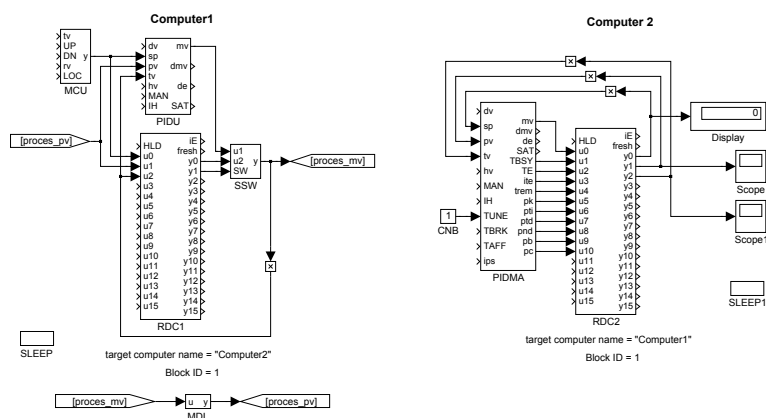
## Parametry

target	Zde se napíše jméno nebo IP adresa počítače, kde běží spolupracující blok RDC. Může to být i broadcast adresa a pak spolupracující blok může být na libovolném počítači v síti, ale při malých periodách (orientačně kratších než 50ms) může docházet k zahlcení sítě. Pro typickou lokální síť (tj. IP adresa 192.168.1.x, maska 255.255.255.0) je broadcast adresa 192.168.1.255 .	String
rport	Vzdálený port (tzv. remote port). Je to vlastně upřesňující adresa nebo též adresa služby protokolu UDP/IP. Pokud nenastane kolize s jinými programy, které používají protokol UDP/IP, je vhodné tento parametr neměnit. ☉1288	Word (U16)
lport	Místní port (tzv. local port), význam podobný jako u parametru rport. Remote port platí pro počítač, kam je paket posílán, local port platí pro počítač, ze kterého je paket posílán. ☉1288	Word (U16)
id	Identifikátor bloku. Toto číslo se posílá v paketu a bloky ve druhém počítači podle něj poznají, pro který blok RDC jsou data určena. Principiálně jej přijmou všechny bloky, ale jen blok, který má stejné číslo id, jej akceptuje a nastaví výstupy na hodnoty z paketu. ↓1 ↑32767 ☉1	Long (I32)
period	Perioda v sekundách, určující nejkratší dobu, po které se vysílají a případně čtou došlé pakety. V případě hodnoty period ≤ 0, se vysílají (a případně i čtou) pakety při každém spuštění bloku. Nastavení kladné hodnoty je výhodné zejména ve spojitých modelech simulovaných systémem Simulink (při použití solveru typu Variable step).	Double (F64)

## Příklad

Následující obrázky představují možné použití bloku RDC. Příklad představuje „vzdálený autotuner“. Jeden počítač (označený **Computer1**) představuje standardní PID regulátor, který řídí technologický proces. Jeho signály **pv**, **sp**, **mv** jsou vedeny na vstupy bloku RDC

a přenášeny na druhý počítač (označený **Computer2**). Na tomto počítači je autotuner (viz popis bloku **PIDMA**), který po náběžné hraně na vstupu **TUNE** provede identifikační experiment a vypočte parametry  $K$ ,  $T_i$ ,  $T_d$  vhodného regulátoru (výstup  $p_k$ ,  $p_{ti}$ ,  $p_{td}$  bloku **PIDMA**). Aby toto mohl udělat, musí se přenášet hodnota  $mv$  autotuneru na akční veličinu technologického procesu. Proto je výstup  $mv$  (hodnota akční veličiny) a **TBSY** (slouží k přepínání mezi  $mv$  PID regulátoru a autotuneru). Všimněme si ještě, že hodnoty  $p_k$ ,  $p_{ti}$ ,  $p_{td}$  jsou vyvedeny na vstupy bloku **RDC2**, takže se hodnoty přenesou na odpovídající výstupy bloku **RDC1**, kde by je bylo možné rovnou použít. Příklad je záměrně jednoduchý, aby byl dobře vidět princip bloku **RDC** a nikoliv složitost algoritmu, který lze v Simulinku vytvořit. Pro pochopení funkce si stačí uvědomit, že funkce uvedeného schématu je stejná, jako když bloky **RDC1** a **RDC2** vypustíme, zbytek obou výkresů sloučíme do jednoho a spojíme to, co původně vedlo na vstup  $u_0$  bloku **RDC1**, s tím, co původně vedlo na výstup  $y_0$  bloku **RDC2**, atd. pro  $u_1, y_1, \dots$



## OPC server pro blok RDC

Existuje OPC server, kterým se lze připojit k bloku **RDC**.

V popisu bloku **RDC** (viz výše) je uvedeno, že dva bloky **RDC** si vzájemně vyměňují hodnoty  $u$  a  $y$ . Jeden z této dvojice bloků může být emulován popisovaným OPC serverem. Jediný parametr, který se zadává je číslo portu. Je to `lport` bloku (resp. všech bloků), které OPC server emuluje. Hodnota se zadává jako parametr `target name` v textové podobě. Implicitní hodnota tohoto parametru je stejná, jako pro blok **RDC** (tj. 1288), takže ji obvykle není nutno měnit.

Pokud je přesto potřeba číslo portu změnit, tak hodnotu je možné zadat buď přímo systémovým programem Windows `regedit` (klíč je

`SOFTWARE\REX Controls\REX_<version>\RdcOPCSvr\TargetName`

- hodnotu je možné zadat do buď do sekce `LocalMachine` nebo `CurrentUser`, někdy je hodnota ve speciální podsekci `VirtualStore` a na 64-bitových počítačích je ještě podsekce `Wow6432Node`) nebo pomocí programu `RexOPCcfg.exe` (je součástí instalace systému **REXYGEN**, ale není na něj odkaz ve startmenu Windows - je potřeba v položce `Key` změnit text `RexOPCSvr` na `RdcOPCSvr` a požadované číslo portu zadat do pole `Target`

name).

Server emuluje bloky všech identifikačních čísel na tomto portu. Protože takových bloků je velké množství, jsou při procházení platných signálů (tzv. browse) zobrazeny jen bloky, od kterých server dostal již nějaká data. OPC klient však může číst i zapisovat hodnoty i od ostatních bloků (čtené hodnoty jsou samozřejmě nesmyslné, ale operace čtení neselže, což je v některých případech důležité).

V adresním prostoru OPC serveru jsou položky ve tvaru

RDC<ID>.<pin>,

kde

<ID> je číslo remote/emulovaného bloku

<pin> je název signálu nabývající jedné z hodnot:

- y0 až y15 – výstupy vzdáleného bloku (tj. vstupy u0 až u15 emulovaného bloku), které lze pouze zapisovat
- u0 až u15 – vstupy vzdáleného bloku (tj. výstupy y0 až y15 emulovaného bloku), které lze pouze číst
- fresh – doba uplynulá od přijetí posledního paketu v sekundách (tj. výstup fresh emulovaného bloku)

## REXLANG – Volně programovatelný blok

Symbol bloku

Licence: [REXLANG](#)

```

XHLD      IE >
XRESET    y0 >
Xu0       y1 >
Xu1       y2 >
Xu2       y3 >
Xu3       y4 >
Xu4       y5 >
Xu5       y6 >
Xu6       y7 >
Xu7       y8 >
Xu8       y9 >
Xu9       y10 >
Xu10      y11 >
Xu11      y12 >
Xu12      y13 >
Xu13      y14 >
Xu14      y15 >
XREXLANG

```

### Popis funkce

Standardní funkční bloky systému REXYGEN řeší většinu běžných požadavků na řízení. Existují však situace, kdy je nutný nebo efektivnější uživatelsky definovaný algoritmus. Blok REXLANG je určen pro tyto případy. Umožňuje implementaci uživatelsky definovaného algoritmu pomocí skriptovacího jazyka, který se blíží jazyku C (nebo Java).

### Skriptovací jazyk

Skriptovací jazyk použitý v REXLANG se podobá jazyku C, ale existují významné rozdíly a omezení:

- Podporované datové typy zahrnují `double`, `long` a `string`. Typy jako `int`, `short` a `bool` jsou interně konvertovány na `long`. Typ `float` je konvertován na `double`. Typ `typedef` není dostupný.
- Ukazatele a struktury nejsou implementovány. Pole definovat lze a indexace je možná pomocí operátoru `[ ]`. Vstupy, výstupy a parametry bloku nemohou být pole.
- Operátor `' , '` není implementován.
- Preprocesor podporuje příkazy jako `#include`, `#define`, `#ifdef .. [#else .. ] #endif`, and `#ifndef .. [#else .. ] #endif`. Nicméně `#pragma` a `#if .. [#else .. ] #endif` nejsou podporovány.
- Standardní knihovny ANSI C nejsou implementovány, ale mnoho matematických funkcí z `math.h` a další funkce jsou dostupné (viz níže).
- Klíčová slova `input`, `output` a `parameter` se používají pro odkazování na vstupy, výstupy a parametry bloku REXLANG. Jsou implementovány systémové funkce pro řízení běhu a diagnostiku.



- Funkce `main()` se provádí periodicky během běhu. K dispozici jsou i další funkce jako `init()` (prováděná jednou při spuštění), `exit()` (prováděná při zastavení řídicího algoritmu) a `parchange()` (prováděná při změnách parametrů).
- Ve funkcích a procedurách bez parametrů musí být v deklaraci explicitně uvedeno `void`.
- Přetěžování identifikátorů není povoleno, tj. klíčová slova a vestavěné funkce nemohou sdílet názvy s identifikátory. Lokální a globální proměnné musí mít různé názvy.
- Nelze inicializovat pole, ať už globální nebo lokální.
- Uživatelsky definované návratové hodnoty z `main()`, `init()` a `exit()` jsou zapsány do výstupu `iE`. Hodnoty  $< -99$  zastaví provádění algoritmu (vyžaduje vstup `RESET` pro další provádění). Návratové hodnoty jsou kategorizovány následovně:
 

<code>iE &gt;= 0</code>	indikuje, že vykonávání je bez chyb.
<code>0 &gt; iE &gt;= -99</code>	znamená varování bez ovlivnění provádění funkčního bloku.
<code>iE &lt; -99</code>	naznačuje chybu, která zastaví provádění funkčního bloku.

## Syntaxe skriptovacího jazyka

Syntaxe skriptovacího jazyka je založena na jazyku C, ale s některými úpravami:

- `<typ> input(<číslo vstupu>) <název proměnné>;` pro vstupní proměnné.
- `<typ> output(<číslo výstupu>) <název proměnné>;` pro výstupní proměnné.
- `<typ> parameter(<číslo parametru>) <název proměnné>;` pro parametrické proměnné.

Proměnné `input` a `parameter` jsou pouze pro čtení, zatímco proměnné `output` jsou pouze pro zápis.

Příklad:

```
double input(1) vstup; /* deklarace proměnné vstup typu double, která
                        představuje hodnotu vstupu bloku u1 */
long output(2) vystup; /* deklarace proměnné vystup typu long, která
                        představuje hodnotu výstupu bloku y2 */

vstup = 3;                //nedovolený příkaz - do vstupu nelze přiřazovat
sum = vystup + 1;        //nedovolený příkaz - z výstupu nelze číst hodnotu

if (vstup>1) vystup= 3 + vstup; //správné použití
```

## Dostupné funkce

Skriptovací jazyk obsahuje širokou škálu funkcí, včetně matematických výpočtů, operací

s vektory, manipulace s řetězci a systémových příkazů. Funkce jsou kategorizovány a podrobně popsány následovně:

- **Matematické funkce** (v souladu s knihovnou `math.h` ANSI C):

Tato kategorie zahrnuje funkce jako `atan`, `sin`, `cos`, `exp`, `log`, `sqrt`, `tan`, `asin`, `acos`, `fabs`, `fmod`, `sinh`, `cosh`, `tanh`, `pow`, `atan2`, `ceil`, `floor` a `abs`. Všimněte si, že `abs` je speciálně navržen pro celočíselné hodnoty, zatímco ostatní operují s proměnnými typu `double`. Pro výpočet absolutní hodnoty desetinného čísla slouží funkce `fabs`.

- **Vektorové funkce** (nejsou v ANSI C) :

Tato sada zahrnuje specializované funkce pro manipulaci s vektory:

`double max([n,] val1, ..., valn)`

Vrací maximální hodnotu mezi zadanými prvky. První parametr, označující počet prvků, je volitelný.

`double max(n, vec)`

Hledá maximální hodnotu ve vektoru `vec`.

`ddouble min([n,] val1, ..., valn)`

Podobně jako `max`, ale vrací minimální hodnotu.

`double min(n, vec)`

Hledá minimální hodnotu ve vektoru `vec`.

`double poly([n,] x, an, ..., a1, a0)`

Vypočítá hodnotu polynomu  $y = anx^n + \dots + a_1x + a_0$ . První parametr je volitelný.

`double poly(n, x, vec)`

Vypočítá hodnotu polynomu  $y = \text{vec}[n]x^n + \dots + \text{vec}[1]x + \text{vec}[0]$ .

`double scal(n, vec1, vec2)`

Vypočítá skalární součin dvou vektorů:  $y = \text{vec1}[0]\text{vec2}[0] + \dots + \text{vec1}[n-1]\text{vec2}[n-1]$ .

`double scal(n, vec1, vec2, skip1, skip2)`

Varianta `scal`, která umožňuje přeskočit prvky:  $y = \text{vec1}[0]\text{vec2}[0] + \text{vec1}[\text{skip1}]\text{vec2}[\text{skip2}] + \dots + \text{vec1}[(n-1)\text{skip1}]\text{vec2}[(n-1)\text{skip2}]$ . Tato varianta je vhodná pro násobení matic, které jsou uloženy jako vektory (řádek po řádku nebo sloupec po sloupci).

`double conv(n, vec1, vec2)`

Vypočítá konvolutorní součin dvou vektorů:  $y = \text{vec1}[0]\text{vec2}[n-1] + \text{vec1}[1]\text{vec2}[n-2] + \dots + \text{vec1}[n-1]\text{vec2}[0]$ .

`double sum(n, vec)`

Sečte prvky vektoru:  $y = \text{vec}[0] + \text{vec}[1] + \dots + \text{vec}[n-1]$ .

`double sum([n,] val1, ..., valn)`

Sečte zadané hodnoty. Počet prvků je volitelný.

`[]array([n,] an-1, ..., a1, a0)`

Vytvoří pole/vektor se zadanými prvky. Parametr `n` je volitelný. Typ vrácené hodnoty je automaticky vybrán tak, aby odpovídal typu parametrů (všechny musí být stejného typu).

`[]subarray(idx, vec)`

Vrátí pole `vec` začínající od indexu `idx`. Typ vrácené hodnoty je automaticky vybrán podle pole `vec`.

`copyarray(count, vecSource, idxSource, vecTarget, idxTarget)`

Kopíruje `count` prvků pole `vecSource`, začínající od indexu `idxSource`, do pole `vecTarget`, začínající od indexu `idxTarget`. Obě pole musí být stejného typu.

`void fillarray(vector, value, count)`

Kopíruje hodnotu `value` do `count` prvků pole `vector` (vždy od indexu 0).

Poznámka: Funkce `max`, `min`, `poly`, `scal`, `conv` a `sum` jsou přetížené, což znamená, že mají několik variant založených na parametrech. Parametry jsou přísně kontrolovány podle typu, vyžadující přetypování pro typy jiné než `double`. Například:

```
double res = max(dVal, (double)iVal, 1.0, (double)2);
```

přetypuje `iVal` na `double`. Pokud parametr není typu `double`, je hlášena chyba s textem "no function of this prototype was found".

- **Funkce pro práci s řetězci** (Tato sekce pokrývá funkce analogické těm, které se nachází v knihovně `string.h` v ANSI C a nabízí řadu operací pro manipulaci a analýzu řetězců:)

`string strstr(str, index, len)`

Extrahuje podřetězec z `str`, začínající na indexu `idx` a pokrývající `len` znaků.

`long strlen(str)`

Vrací délku řetězce `str`, měřeno v znacích.

`long strstrfind(str, substr[, offset])`

Najde první výskyt `substr` v `str` a vrací jeho pozici. Prohledávání začíná od znaku s indexem `offset` (pokud není zadán, tak od začátku). Parametr `substr` může být i znak.

`long strstrfind(str, substr)`

Identifikuje poslední výskyt `substr` v `str` a poskytuje jeho index.

`strreplace(str, pattern, substr)`

Nahrazuje všechny instance `pattern` v `str` řetězcem `substr`. Tato úprava je provedena přímo v `str`.

`strupr(str)`

Převede všechny znaky v `str` na velká písmena.

`strlwr(str)`

Převede `str` na malá písmena.

`strtrim(str)`

Odstraní počáteční a koncové bílé znaky z `str`.

`long str2long(str[, default])`

Převede `str` na celé číslo. Pokud konverze selže, je vrácen volitelný druhý parametr (výchozí hodnota) nebo 0, pokud není poskytnut.

- double** `str2double(str[, default])`  
 Převede `str` na desetinné číslo. Podobně jako `str2long`, vrací volitelnou výchozí hodnotu nebo 0 v případě selhání.
- string** `long2str(num[, radix])`  
 Převede celé číslo `num` na řetězec s volitelným parametrem `radix`, který určuje základ (výchozí je 10). Výstupní řetězec neudává číselný systém (bez prefixů jako 0x pro hexadecimální).
- string** `double2str(num)`  
 Převede desetinné číslo `num` na jeho řetězcovou reprezentaci.
- strcpy(dest, src)**  
 Zkopíruje obsah `src` do `dest`. Pro kompatibilitu s ANSI C dosahuje stejného výsledku `dest = src`.
- strcat(dest, src)**  
 Připojí `src` na konec `dest`. Stejně jako v ANSI C, `dest = dest + src` provádí stejnou operaci.
- strcmp(str1, str2)**  
 Porovná dva řetězce `str1` a `str2`. Pro stejný účel lze použít konstrukci `str1 == str2`, což je kompatibilní s ANSI C.
- float2buf(buf, x[, endian])**  
 Převede reálné číslo `x` do pole `buf` čtyř prvků, z nichž každý reprezentuje jeden byte čísla uloženého ve formátu IEEE 754 single precision (známé jako float). Funkce je užitečná pro naplnění komunikačních bufferů. Volitelný 3. parametr má následující význam:  
 0 nativní endian procesoru (výchozí),  
 1 little endian,  
 2 big endian.
- double2buf(buf, x[, endian])**  
 Podobně jako `float2buf`, ale pro dvojitou přesnost, ukládá osm prvků (typ double).
- double** `buf2float(buf[, endian])`  
 Inverzní funkce k `float2buf`.
- double** `buf2double(buf[, endian])`  
 Inverzní funkce k `double2buf`.
- long** `RegExp(str, regexp, capture[])`  
 Porovnává `str` s regulárním výrazem `regexp`, ukládá zachycené skupiny do `capture`. Vrací počet zachycených skupin nebo záporný kód chyby. Syntaxe regulárního výrazu zahrnuje standardní konstrukce jako:

(?i)	Musí být na začátku regulárního výrazu. Umožňuje porovnání bez ohledu na velikost písmen.
^	Porovnává začátek řetězce
\$	Porovnává konec řetězce
()	Skupinování a zachycování podřetězců
\s	Porovnává bílý znak
\S	Porovnává ne-bílý znak
\d	Porovnává desetinnou číslici
\n	Porovnává znak nového řádku
\r	Porovnává znak pro posun řádku
\f	Porovnává znak vertikální tabulace
\v	Porovnává znak horizontální tabulace
\t	Porovnává znak horizontální tabulace
\b	Porovnává znak zpětného posunu
+	Porovnává jeden nebo vícekrát (nenasytně)
+?	Porovnává jeden nebo vícekrát (nenenasytně)
*	Porovnává nula nebo vícekrát (nenasytně)
*?	Porovnává nula nebo vícekrát (nenenasytně)
?	Porovnává nula nebo jednou (nenenasytně)
x y	Porovnává x nebo y (operátor alternace)
\meta	Porovnává jeden z meta znaků: ^\$().[]*+? \
\xHH	Porovnává bajt s hexadecimální hodnotou 0xHH, např. \x4a.
[...]	Porovnává jakýkoliv znak ze sady. Podporovány jsou rozsahy jako [a-z].
[^...]	Porovnává jakýkoliv znak kromě těch ze sady.

Příklad:

```
RegExp("48,string1,string2","^(\\d+),([^\,]+)","capture);
```

```
Výsledek: capture=["48,string1","48","string1"]
```

```
long ParseJson(json, cnt, names[], values[])
```

Tato funkce zpracovává `json` řetězec, extrahuje hodnoty specifikovaných objektů. Pole `names` by mělo obsahovat názvy požadovaných vlastností (přístup k podpoložkám pomocí `.` a indexů pole pomocí `[]`, například `"cars[1].model"`). Odpovídající hodnoty jsou poté umístěny do pole `values`. Parametr `cnt` určuje počet objektů k analýze, který by měl odpovídat délce obou polí `names` a `values`. Funkce vrací celkový počet úspěšně analyzovaných hodnot nebo zápornou hodnotu, pokud dojde během analýzy k chybě.

Poznámka: Řetězcové proměnné jsou deklarovány stejně jako v ANSI C (`char <název proměnné>[<max znaků>];`). Pro argumenty funkcí použijte `char <název proměnné>[]` nebo `string <název proměnné>`.

- **Systémové funkce** (nejsou v ANSI C)

```
Archive(arc, type, id, lvl_cnt, value)
```

Tato funkce archivuje hodnotu do archivačního podsystému systému. `arc`

slouží jako bitová maska pro určení cílových archivů (například pro archivy 3 a 5 nastavte `arc = 20`, což je binárně 10100 nebo 20 v desítkové soustavě). Číslování archivů začíná od 1, s maximem 15 archivů (archiv 0 je vyhrazen pro interní systémové logy). Parametr `type` definuje datový typ s možnostmi:

- 1 Bool
- 2 Byte (U8)
- 3 Short (I16)
- 4 Long (I32)
- 5 Word (U16)
- 6 DWord (U32)
- 7 Float (F32)
- 8 Double (F64)
- 9 Time
- 10 Large (I64)
- 11 Error
- 12 String
- 17 Bool Group
- 18 Byte Group (U8)
- 19 Short Group (I16)
- 20 Long Group (I32)
- 21 Word Group (U16)
- 22 DWord Group (U32)
- 23 Float Group (F32)
- 24 Double Group (F64)
- 25 Time Group
- 26 Large Group (I64)
- 27 Error Group

`id` představuje unikátní ID archivní položky, `lvl_cnt` označuje úroveň plochy nebo počet prvků pro skupinové typy a `value` jsou data k archivaci.

`Trace(id, val)`

Zobrazuje hodnoty `id` a `val`. Funkce je určena zejména pro účely ladění. `id` je uživatelem definovaná konstanta v rozsahu od 0 do 9999 pro snadnou identifikaci zprávy. `val` může být jakýkoliv datový typ, včetně řetězců. Výstup se objevuje v systémovém logu REXYGEN.. Pro zobrazení těchto ladicích zpráv v systémovém logu je nutné je povolit. Jděte do menu *Target*→*Diagnostic messages* a zaškrtněte políčko *Information* v boxu *Function block messages*. Logování musí být také povoleno pro daný blok zaškrtnutím políčka *Enable logging* v záložce *Runtime* v dialogu parametrů bloku. Ve výchozím stavu je toto povoleno po umístění nového bloku z knihovny. Teprve poté jsou zprávy zobrazeny v *Systémovém logu*.

`TraceError(id, val)` `TraceWarning(id, val)` `TraceVerbose(id, val)`

Tyto příkazy mají podobný význam jako příkaz `Trace`, avšak výpis se objeví v jiné skupině systémového logu (Error, Warning, Verbose). Výpisy

úrovně *Error* se do logu zapisují vždy, nezávisle na zaškrtnutí checkboxu *Enable logging* daného bloku. Zprávy úrovně *Warning* a *Verbose* je potřeba nejprve povolit, stejně jako v případě příkazu *Trace*.

#### **Suspend(sec)**

Přeruší provádění kódu skriptu, pokud od jeho puštění (v dané periodě) uplynulo více času (v sekundách), než je uvedeno. Při dalším spuštění bloku se pokračuje za tímto příkazem. Při *Suspend(0)* dojde k přerušení vždy.

#### **double GetPeriod()**

Vrací vlastní periodu spouštění daného bloku ve vteřinách.

#### **double CurrentTime()**

Vrací aktuální čas (v interním formátu). Používá se ve spojení s funkcí *ElapsedTime()*.

#### **double ElapsedTime(new\_time, old\_time)**

Vrací uplynulý čas v sekundách (desetinné číslo), tj. rozdíl mezi časy určenými parametry *new\_time* a *old\_time*, získaným z předchozího volání funkce *CurrentTime()*.

#### **double Random()**

Vrací pseudonáhodné číslo z intervalu  $(0, 1)$ . Před voláním funkce *init()* se automaticky inicializuje generátor pseudonáhodného čísla, takže sekvence je vždy stejná.

#### **long QGet(var)**

Vrací kvalitu proměnné *var* (tak jak s ní pracuje systém REXYGEN, viz bloky *QFC*, *QFD*, *VIN*, *VOUT*). Funkci je možno použít jen pro vstupy, parametry a výstupy - pro vnitřní proměnné vrací vždy 0.

#### **void QSet(var, value)**

Nastaví kvalitu proměnné *var* (tak jak s ní pracuje systém REXYGEN) na hodnotu *val*. Funkci je možno použít jen pro výstupy - pro ostatní se nic nenastaví.

#### **long QPropag([n, ]val1, ..., valn)**

Vrací kvalitu, která vznikne sloučením kvalit *val1, ..., valn*. Základní pravidlo pro slučování je, že výsledná kvalita je nejhorší ze vstupních. Pokud nastavíme kvalitu výstupu bloku s použitím této funkce, tak že do parametrů dáme kvalitu všech vstupů bloku, které výstup ovlivňují, dostaneme stejné chování jako u ostatních bloků systému REXYGEN.

#### **double LoadValue(fileid, idx)**

Čte hodnotu ze souboru. Podporuje jak binární soubory (s hodnotami *double*) tak textové soubory (hodnoty na samostatných řádcích). Soubor je identifikován *fileid*, a *idx* udává index (pro binární soubory) nebo číslo řádku (pro textové soubory). V současnosti jsou podporovány následující hodnoty:

- 0 soubor na disku identifikovaný parametrem `p0`,
- 1 soubor na disku identifikovaný názvem bloku `REXLANG` a příponou `.dat`,
- 2 soubor na disku identifikovaný parametrem `srcname`, ale přípona je změněna na `.dat`,
- 3 soubor `rexlang.dat` v aktuálním adresáři,
- 4-7 stejné jako 0-3, ale formát je textový soubor. Každý řádek obsahuje jedno číslo. Index `idx` je číslo řádku a začíná na nule. Hodnota `idx=-1` znamená další řádek (např. sekvenční zápis).

`void SaveValue(fileid, idx, value)`

Ukládá hodnotu do souboru. Parametry jsou podobné jako u `LoadValue`.

`void GetSystemTime(time)`

Vrací systémový čas v UTC (upravitelný prostřednictvím nastavení operačního systému). Parametr `time` musí být pole alespoň 8 položek typu `long`. Funkce vyplní pole následujícími hodnotami v daném pořadí: rok, měsíc, den (v měsíci), den v týdnu, hodiny, minuty, sekundy, milisekundy. Na některých platformách má hodnota milisekund omezenou přesnost nebo není vůbec k dispozici (funkce vrací 0 ms).

`void Sleep(seconds)`

Zastaví provádění algoritmu bloku (a celého tasku) na definovanou dobu. Používejte tento blok s extrémní opatrností a pouze v případě, že neexistuje jiná možnost, jak dosáhnout požadovaného chování algoritmu. Doba usnutí by neměla překročit 900 milisekund. Nejkratší interval je asi 0,01s, přesná hodnota závisí na cílové platformě.

`long GetExtInt(ItemID)`

Vrací hodnotu vstupu/výstupu/parametru libovolného bloku v algoritmu `REXYGEN`. Tato externí datová položka je odkazována parametrem `ItemID`. Struktura řetězcového parametru `ItemID` je stejná jako například v parametru `sc` bloku `GETPI`. Pokud nelze hodnotu získat (např. neplatné nebo neexistující `ItemID`, konflikt datového typu atd.), blok `REXLANG` ohlásí chybu a musí být resetován.

`long GetExtLong(ItemID)`

Viz `GetExtInt(ItemID)`.

`double GetExtReal(ItemID)`

Stejný význam jako předchozí příkazy, ale pro desetinné číslo.

`double GetExtDouble(ItemID)`

Viz `GetExtReal(ItemID)`.

`string GetExtString(ItemID)`

Stejný význam jako předchozí příkazy, ale pro řetězec/text.

`void SetExt(ItemID, value)`

Nastaví hodnotu vstupu/výstupu/parametru libovolného bloku v `REX`u určeného parametrem `ItemID`. Tento parametr je textový a má stejný význam/strukturu jako parametr `sc` bloku `SETPI`. Nastavuje se hodnota



parametru `value`, přičemž typ nastavené hodnoty (`long/double/string`) je určen typem parametru `value`. Pokud hodnotu nelze nastavit (např. neexistující `ItemID` nebo neodpovídá datový typ) blok `REXLANG` skončí chybou a je potřeba ho resetovat.

`int BrowseExt(ItemID, first_subitem_index, max_count, subitems, kinds)`

Funkce prochází adresový prostor úlohy. Pokud je `ItemID` identifikátorem bloku (cestou k bloku), pole řetězců `subitems` bude obsahovat názvy všech vstupů, výstupů, parametrů a vnitřních stavů. Funkce vrací počet podpoložek nebo záporný chybový kód. Hodnoty `kinds` jsou: `executive = 0`, `module = 1`, `driver = 2`, `archive = 3`, `level = 4`, `task = 5`, `quicktask = 6`, `subsystem = 7`, `block = 8`, `input = 9`, `output = 10`, `internal state = 11`, `parameter or state array = 12`, `special = 13`.

`long CallExt(ItemID)`

Spustí jeden krok jakéhokoli bloku v algoritmu `REXYGEN`, identifikovaného pomocí `ItemID`. Struktura textového parametru `ItemID` je stejná jako například u parametru `sc` bloku `GETPI`. Doporučuje se volat pouze pozastavené bloky<sup>1</sup>, které by měly být ve stejném úkolu jako blok `REXLANG`. Funkce vrací výsledkový kód volaného bloku (viz chybové kódy `REXYGEN`).

`long GetInArrRows(input)`

Vrací počet řádek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`long GetInArrCols(input)`

Vrací počet sloupců pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`long GetInArrMax(input)`

Vrací maximální (alokovanou) velikost pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`double GetInArrDouble(input, row, col)`

Vrací prvek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`long GetInArrLong(input, row, col)`

Stejně jako `GetInArrDouble(...)`, ale hodnota je typu `long`.

`Void SetInArrValue(input, row, col, value)`

Nastaví prvek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`Void SetInArrDim(input, row, col)`

Nastaví rozměr pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

`long memrd32(hMem, offset)`

Čtení fyzické paměti. Handle se získá pomocí funkce

`Open(72, "/dev/mem", <fyzická adresa>, <velikost oblasti>)`.

---

<sup>1</sup>nastavte zaškrtnuté políčko `Halt` na stránce vlastností `Runtime` v dialogu nastavení parametrů bloku

`long memwr32(hMem, offset, value)`  
 Zápis do fyzické paměti. Handle se získá funkce  
`OpenMemory("/dev/mem", <physical address>, <area size>)`.

- **Komunikační funkce** (nejsou v ANSI C)

Tento soubor funkcí usnadňuje komunikaci přes různé kanály včetně TCP/IP, UDP/IP, sériových linek (RS-232 nebo RS-485), SPI sběrnice a I2C sběrnice. Níže je stručný seznam těchto funkcí. Pro podrobné informace viz příkladové projekty v systému REXYGEN.

`long OpenFile(string filename)`  
 Tato funkce otevře soubor a vrátí identifikační číslo (handle) souboru. Pokud funkce vrátí zápornou hodnotu, otevření souboru nebylo úspěšné.

`long OpenCom(string comname, long baudrate, long parity)`  
 Otevírá sériovou linku a vrátí její handle. Od verze 3.0 REXYGEN lze jako `comname` specifikovat virtuální porty. Pro podrobné informace o virtuálních portech viz popis bloku [UART](#). Pokud je vrácena záporná hodnota, otevření selhalo. Možnosti nastavení parity jsou 0 pro žádnou, 1 pro lichou a 2 pro sudou.

`long OpenUDP(string localname, long lclPort, string remotename, long remPort)`  
 Otevírá UDP socket a vrátí jeho handle. Pokud funkce vrátí zápornou hodnotu, otevření socketu nebylo úspěšné. Funkce může otevřít buď IPv4 nebo IPv6 socket na základě `remotename`, `localname` nebo nastavení operačního systému, pokud je použito DNS jméno. Volitelná nastavení zahrnují prázdný `localname` (jakékoli rozhraní), prázdný `remotename` nebo 0 pro `remPort` (nepoužívá se) a 0 pro `lclPort` (přiděleno UDP/IP stackem).

`long OpenTCPSvr(string localname, long lclPort)`  
 Tato funkce otevírá TCP socket v režimu serveru (poslechu). Vrací handle socketu a záporná návratová hodnota znamená neúspěšné otevření. Funkce může otevřít buď IPv4 nebo IPv6 socket v závislosti na `remotename`, `localname` nebo nastavení operačního systému, pokud je použito DNS jméno. Můžete nastavit prázdný `localname` jako jakékoli rozhraní.

`long OpenTCPcli(string remotename, long remPort)`  
 Otevírá TCP socket v režimu klienta a vrátí jeho handle. Záporná návratová hodnota znamená selhání otevření socketu. Funkce otevírá buď IPv4 nebo IPv6 socket na základě `remotename`, `localname` nebo nastavení operačního systému, pokud je použito DNS jméno. Všimněte si, že tato funkce nečeká na navázání spojení, což může trvat několik milisekund v místní síti nebo několik sekund pro vzdálené lokace. Pokud jsou volány `Write()` nebo `Read()` před navázáním spojení, je vrácen chybový kód -307 (chyba otevření souboru).

`long OpenI2C(string devicename)`  
 Otevírá I2C sběrnici a vrátí její handle. Pokud funkce vrátí zápornou hodnotu, otevření nebylo úspěšné.

- long OpenSPI(string devicename)**  
 Otevírá SPI sběrnici a vrátí její handle. Záporná návratová hodnota znamená neúspěšné otevření.
- long OpenDevice(string filename)**  
 Podobně jako `OpenFile()`, ale funkce `Write()` a `Read()` jsou neblokující. Pokud nelze data číst nebo zapisovat, funkce okamžitě vrátí chybový kód -1.
- long OpenMemory(string devicename, long baseaddr, long size)**  
 Mapuje fyzickou paměť a vrátí přidružený handle. Pokud je vrácena záporná hodnota, operace nebyla úspěšná.
- long OpenSHM(string devicename, long deviceid, long size, long flags)**  
 Mapuje sdílenou paměť (pouze v Linuxu, využívá `ftok()` a `shmget()`) a vrací přidružený handle. První a druhý parametr slouží k identifikaci paměťové oblasti (musí být shodné pro všechny kooperující entity). Parametr `size` udává velikost sdílené paměťové oblasti v bajtech. Parametr `flags` reprezentuje standardní linuxové příznaky a oprávnění (pokud je nastaveno na 0, což je výchozí hodnota, jsou nastavena následující práva: vytvořit oblast, pokud neexistuje, a umožnit všem čtení i zápis).
- void Close(long handle)**  
 Zavře socket, sériovou linku, soubor nebo jakékoli zařízení otevřené pomocí funkcí `Open...`
- void SetOptions(long handle, long params[])**  
 Nastavuje parametry socketu nebo sériové linky. Velikost pole musí být alespoň:
- 22 pro sériovou linku (na Windows parametry pro `SetCommState()` a `SetCommTimeouts()` v následujícím pořadí: `BaudRate`, `fParity`, `Parity`, `StopBits`, `ByteSize`, `fDtrControl`, `fRtsControl`, `fAbortOnError`, `fBinary`, `fErrorChar`, `fNull`, `fDsrSensitivity`, `fInX`, `fOutX`, `fOutxCtsFlow`, `fOutxDsrFlow`, `fTXContinueOnXoff`, `ReadIntervalTimeout`, `ReadTotalTimeoutConstant`, `ReadTotalTimeoutMultiplier`, `WriteTotalTimeoutConstant`, `WriteTotalTimeoutMultiplier`; linux používá odlišnou funkci, ale význam parametrů je stejný, pokud je to možné),
  - 2 pro soubor (1. položka je mód: 1=vyhledání od začátku, 2=vyhledání od aktuální pozice, 3=vyhledání od konce, 4=nastavit konec souboru, 2. položka je posun pro vyhledání),
  - 3 pro SPI (1. položka je režim SPI, 2. položka je počet bitů na slovo, 3. položka je maximální rychlost v Hz),
  - 5 pro I2C (1. položka je adresa zařízení, 2. položka je příznak 10bitové adresy, 3. položka je příznak kontroly chyb v balíčku, 4. položka je počet pokusů, 5. položka je časový limit),
- ostatní typy handle nejsou podporovány

`void GetOptions(long handle, long params[])`

Načte parametry socketu nebo sériové linky a uloží je do pole `params`. Velikost pole musí odpovídat požadavkům daného zařízení (viz `SetOptions`).

`long Accept(long hListen)`

Přijme spojení na poslouchajícím socketu a vrátí handle komunikačního socketu nebo chybový kód.

`long Read(long handle, long buffer[], long count[, offset])`

Přijímá data ze sériové linky nebo socketu a vrátí počet přečtených bajtů nebo chybový kód. Parametr `count` definuje maximální počet bajtů k přečtení. Každý bajt přichozích dat je vložen do pole `buffer` typu `long` ve stejném pořadí. Funkce má ještě jeden (nepovinný) parametr `offset`, který lze použít při čtení dat z paměti, kdy je handle vytvořen pomocí `OpenSHM()` nebo `OpenMemory()`.

Lze také použít zápis

`long Read(long handle, string data[], long count)` (tj. používá se řetězec místo pole dat; jeden bajt ve vstupním souboru odpovídá jednomu znaku; nelze použít pro binární soubory).

Chybové kódy jsou:

- 1 je nutné počkat na dokončení operace (funkce je "neblokující")
- 309 čtení selhalo; chybový kód operačního systému se zobrazí v logu (pokud je povoleno logování bloků funkcí)
- 307 soubor/socket není otevřen

`long Write(long handle, long buffer[], long count[, offset])`

Odesílá data přes sériovou linku nebo socket. Parametr `count` definuje počet bajtů k odeslání. Počet odeslaných bajtů nebo chybový kód je vrácen. Každý bajt odchozích dat je přečten z pole `buffer` typu `long` ve stejném pořadí. Funkce má ještě jeden (nepovinný) parametr `offset`, který lze použít pro zápis dat do paměti, kdy je handle vytvořen pomocí `OpenSHM()` nebo `OpenMemory()`.

Lze také použít formu

`long Write(long handle, string data)` (tj. používá se řetězec místo pole dat; jeden bajt ve výstupním souboru odpovídá jednomu znaku; nelze použít pro binární soubory).

Chybové kódy jsou:

- 1 je nutné počkat na dokončení operace (funkce je "neblokující")
- 310 zápis selhal; chybový kód operačního systému se zobrazí v logu (pokud je povoleno logování bloků funkcí)
- 307 soubor/socket není otevřen

`long ReadLine(long handle, string data)`

Čte řádek z (textového) souboru, sériové linky nebo socketu a ukládá znaky do `data` až do jejich alokované velikosti. Funkce vrátí skutečnou velikost řádku nebo chybový kód.

- `long DeleteFile(string filename)`  
Smaže soubor. Vrací 0 pokud je soubor smazán; záporné číslo znamená chybu.
- `long RenameFile(string filename, string newfilename)`  
Přejmenuje soubor. Vrací 0 pokud je soubor přejmenován; záporné číslo znamená chybu.
- `bool ExistFile(string filename)`  
Ověří, zda soubor nebo zařízení existuje (lze otevřít pro čtení), vrátí true nebo false.
- `long I2C(long handle, long addr, long bufW[], long cntW, long bufR[], long cntR)`  
Zajišťuje komunikaci přes sběrnici I2C, zejména na Linuxových systémech s možností I2C (např. Raspberry Pi). Funkce odesílá a přijímá data do/z slave zařízení pomocí `addr`. Parametr `handle` je vrácen funkcí `OpenI2C`, jejíž parametr určuje název zařízení (podle operačního systému). Parametr `bufW` je buffer (pole) pro data, která se odesílají, `cntW` je počet bajtů k odeslání, `bufR` je buffer (pole) pro příchozí data a `cntR` je počet bajtů k přijetí. Funkce vrací 0 nebo chybový kód.
- `long SPI(long handle, 0, long bufW[], long cntW, long bufR[], long cntR)`  
Provede transakci přes sběrnici SPI, zejména na Linuxových systémech se sběrnici SPI. Parametr `handle` je vrácen funkcí `OpenSPI`, jejíž parametr určuje název zařízení (podle operačního systému). Druhý parametr je vždy 0 (rezervováno pro interní použití). Parametr `bufW` je buffer (pole) pro data, která se odesílají, `cntW` je počet bajtů k odeslání, `bufR` je buffer (pole) pro příchozí data a `cntR` je počet bajtů k přijetí. Všimněte si, že komunikace přes sběrnici SPI je full-duplex, proto výsledná délka transakce SPI je dána maximem parametrů `cntW` a `cntR`, nikoli jejich součtem. Funkce vrací 0 nebo chybový kód.
- `long Seek(long handle, long mode[], long offset)`  
Nastaví pozici pro příkazy Read/Write. Parametr `mode` znamená:  
  - 1 posun od začátku souboru,
  - 2 posun od aktuální pozice,
  - 3 posun od konce souboru.
- `long Recv(long handle, long buffer[], long count)`  
*Zastaralá funkce. Použijte Read.*
- `long Send(long handle, long buffer[], long count)`  
*Zastaralá funkce. Použijte Write.*
- `long crc16(data, length, init, poly, flags, offset)`  
Vypočítá 16bitový cyklický redundantní kód (CRC), běžně používaný jako kontrolní součet nebo hash v různých komunikačních protokolech.  
`data` Pole bajtů (reprezentováno polem typu long) nebo řetězec, pro který se vypočítává hash.  
`length` Počet bajtů ve vstupním poli nebo textu. Použijte -1 pro zpracování celého řetězce.

**init** Počáteční vektor pro výpočet CRC.

**poly** Kontrolní polynom použitý ve výpočtu.

**flags** Konfigurační příznaky:

- 1: Obrací pořadí bitů jak ve vstupních bajtech, tak ve výsledném CRC.
- 2: Výsledné CRC je XORováno s 0xFFFF.
- 4: Pokud je **data** pole typu long, jsou zpracovány všechny 4 bajty v prvku pole (LSB první).
- 8: Podobně jako příznak 4, ale zpracovává MSB první.

**offset** Index prvního zpracovávaného bajtu v datovém poli (obvykle 0).

Poznámka: Existují analogické funkce pro výpočet 32bitového a 8bitového CRC: `long crc32(data, length, init, poly, flags, offset)` a `long crc8(data, length, init, poly, flags, offset)`. Počáteční vektor, kontrolní polynom a příznaky pro různé protokoly lze nalézt na <https://crccalc.com/>

Příklady:

- MODBUS: `crc16("123456789", -1, 0xFFFF, 0x8005, 1, 0)`
- DECT-X: `crc16("123456789", -1, 0, 0x0589, 0, 0)`

**Další poznámka:** Funkce `crc8(...)` a `crc32(...)` také existují, podporují výpočty 8bitových a 32bitových CRC se stejnou strukturou parametrů.

## Ladění kódu, debugging

Pro ladění kódu je k dispozici příkaz **Trace**, viz výše. Dále lze použít výstupy bloku, které se noužívají pro vlastní algoritmus a zapisovat do nich hodnoty různých mezivýpočtů. V závislosti na povaze algoritmu může být vhodné tyto ladící hodnoty připojit do trendu. Pokud je potřeba sledovat hodnot více, je možné do tasku přidat blok CNA (připojený na TRNDV nebo VTOR) a do hodnot v jeho poli nastavovat opět různé mezivýsledky pomocí funkce **SetExt**.

## Poznámky

- Datové typy vstupů `u0..u15`, výstupů `y0..y15` a parametrů `p0..p15` jsou určeny během kompilace zdrojového kódu.
- Chybové kódy `< -99` vyžadují vstup **RESET** pro restartování bloku **REXLANG** po řešení příčiny chyby.
- **POZOR!!!** Ve funkci `init()` je sice možné číst vstupy, ale protože ostatní bloky obvykle nenastavují v `init` fázi výstupy, bude tam vždy 0. Nastavovat výstupy lze, ale obvykle se to nedělá.

- Skript může obsahovat cestu v parametru `srcname`; pokud ne, očekává se soubor v adresáři projektu nebo ve specifikovaných adresářích (viz parametr `LibraryPath` bloku `PARAM`).
- Parametr `srcname` je možné udávat s absolutní cestou. V opačném případě se soubor hledá na aktuálním adresáři a specifikovaných adresářích (viz parametr `LibraryPath` bloku `PARAM`).
- Parametry vektorových funkcí jsou primárně typu `double`, s výjimkou parametru `n`, který je typu `long`. Funkce s jedním vektorovým parametrem existují ve třech variantách:

```
double function(val1, ..., valn)
```

Vektor je definován jako sekvence hodnot typu `double`.

```
double function(n, val1, ..., valn)
```

Vektor je definován jako v prvním případě, pouze první parametr definuje počet hodnot – velikost vektoru. Tato varianta je kompatibilní s kompilátorem jazyka C. Parametr `n`<sup>2</sup> musí být číslo, nikoli tzv. `const` proměnná, a musí odpovídat počtu následujících prvků definujících vektor.

```
double function(n, vec)
```

Parametr `n` je libovolný výraz typu `long` a definuje počet prvků, které funkce zohledňuje.

- Je důležité si pamatovat, že pole v skriptovacím jazyku se chovají podobně jako pole v jazyce C: indexace začíná od 0 a neexistuje automatická kontrola hranic. Například pokud deklarujete `double vec[10], x;`, pole `vec` bude mít prvky indexované od 0 do 9. Přístup k `vec[10]` nevyvolá chybu syntaxe ani runtime, ale vrácená hodnota je nedefinovaná, protože leží mimo hranice pole. Navíc přiřazení hodnoty k `vec[11]` (např. `vec[11] = x;`) může být obzvláště nebezpečné, protože může nechtěně přepsat sousední paměťová místa. To by mohlo vést k nepředvídatelnému chování nebo způsobit pád programu.
- Během kompilačního procesu, pokud existují chyby syntaxe, kompilátor hlásí **parser error** spolu s číslem řádku, kde došlo k chybě. Tyto hlášení konkrétně ukazují na problémy se syntaxí. Pokud však syntaxe vypadá správně a chyba je stále hlášena, je vhodné zkontrolovat konflikty zahrnující identifikátory, klíčová slova nebo názvy funkcí, jelikož tyto mohou také způsobit chyby neokamžitě zřejmé jako chyby syntaxe.
- Všechny skoky jsou přeloženy jako relativní, tj. odpovídající kód je omezen na 32767 instrukcí (ve formátu přenosném pro různé platformy).

---

<sup>2</sup>Volitelný parametr `n` vektorových funkcí musí být specifikován, pokud je vyžadována kompatibilita s kompilátorem jazyka C/C++. V takovém případě musí být implementovány také všechny nestandardní funkce a funkce s proměnným počtem parametrů musí znát počet parametrů.

- Všechny platné proměnné a výsledky dočasných výpočtů jsou uloženy v zásobníku, který zahrnuje:
  - Globální a lokální `static` proměnné jsou trvale umístěny na začátku zásobníku.
  - Návrátové adresy pro volání funkcí.
  - Parametry funkcí.
  - Proměnné lokální pro funkce.
  - Návrátová hodnota funkce.
  - Dočasné výsledky výpočtů. Například ve výrazu `a = b + c`; jsou nejprve `b` a `c` vloženy na zásobník. Jejich součet je pak vypočítán, operandy jsou odstraněny ze zásobníku a výsledek je vložen na zásobník.

Jednoduché proměnné, jako jsou `long` nebo `double`, zabírají jeden slot zásobníku. U polí záleží na celkové velikosti, bez ohledu na typ prvků.

- Když jsou pole předávána funkcím, jsou referencována, nikoli kopírována. To znamená, že pro referenci je použit pouze jeden slot zásobníku a funkce přímo pracuje s původním polem.
- Pokud je alokovaná velikost zásobníku nedostatečná (méně než prostor potřebný pro globální proměnné plus 10), automaticky se zdvojnásobí, s dalšími 100 sloty pro výpočetní potřeby, parametry funkcí a lokální proměnné, zejména pokud je definováno málo globálních proměnných.
- Při základní úrovni ladění jsou během provádění skriptu prováděny různé kontroly. Ty zahrnují inicializaci čtených hodnot a kontrolu hranic indexů polí. Navíc je na začátek a konec každého deklarovaného pole vloženo několik neinicializovaných hodnot pro kontrolu hranic a do souboru `*.ill` jsou přidány instrukce `NOP` s čísly řádků zdrojového souboru.
- Při úplné úrovni ladění je povolena další kontrola neplatných přístupů k datovým rozsahům (jako jsou přetečení zásobníku).
- V tomto kontextu se 'instrukcí' rozumí procesorově nezávislý mnemonický kód. Tyto kódy jsou uloženy v souboru `*.ill`.
- Funkce `OpenCom()` nastaví binární neblokující režim bez timeoutů, 8 datových bitů, 1 stopbit, bez parity, 19200Bd. Volitelně lze upravit `baudrate` a `parity` parametr ve funkci `OpenCom()`.
- Přístup k textovým souborům je výrazně pomalejší než k binárním souborům. Textové soubory však mají výhodu, že jsou viditelné a upravitelné bez specializovaného software.



- Blok automaticky nevyvolá funkci `parchange()`. Tato funkce musí být manuálně volána v rámci funkce `init()`, pokud je to potřeba.
- Funkce `OpenFile()` otevírá soubory v datovém adresáři systému REXYGEN (tj. v Linuxu implicitně v `\rex\data`, na Windows `C:\ProgramData\REX Controls\REX_<verze>\RexCore`). Jsou dovoleny podadresáře, ale není dovoleno `..\`. Linky se následují.

## Ladění kódu

Použijte výše uvedený příkaz `Trace`.

## Vstupy

HLD	Pozastavení – kód bloku se nevykonává, je-li hodnota rovna <code>on</code> .	Bool
RESET	Resetování chyby při náběžné hraně; blok se znovu inicializuje (vynulují se všechny globální proměnné a zavolá se funkce <code>Init()</code> )	Bool
u0..u15	Vstupní signály, jejichž hodnoty jsou přístupné ve skriptu	Any

## Výstupy

iE	Kód chyby při běhu skriptu. Pro chybové <code>&lt; -99</code> je provádění algoritmu bloku zastaveno do reinicializace vstupem <code>RESET</code> nebo novým spuštěním exekutivy <ul style="list-style-type: none"> <li><code>0</code> ..... vše v pořádku, proběhla celá funkce <code>main()</code> popř. <code>init()</code></li> <li><code>-1</code> ..... provádění programu skončilo příkazem <code>Suspend()</code>, tj. vypršel čas pro výpočet; výpočet bude při dalším spuštění bloku pokračovat tam, kde skončil</li> <li><code>&lt; -1</code> .. chybový kód <code>xxx</code> systému REXYGEN, více viz příloha C</li> <li><code>&gt; 0</code> ... uživatelské návratové hodnoty, běh algoritmu beze změny</li> </ul>	Error
y0..y15	Výstupní signály, jejichž hodnoty jsou definovány ve skriptu	Any

## Parametry

srcname	Jméno souboru se skriptem	⊙srcfile.c	String
---------	---------------------------	------------	--------

<b>srctype</b>	Typ zdrojového souboru 1: C-like Textový soubor s výše popisovanou syntaxí obdobnou jazyku C. 2: STL Textový soubor se syntaxí dle IEC61131-3. Norma je implementována se stejnými omezeními jako C-like skript (tj. žádné struktury, z typů jen INT a REAL a STRING, vstupy bloku jsou globální VAR_INPUT, výstupy bloku jsou globální VAR_OUTPUT, parametry bloku jsou globální VAR_PARAMETER, standardní funkce dle specifikace, systémové a komunikační funkce jako v C-like) 3: RLB Soubor v binárním formátu, který vzniká při překladu z formátu STL i C-like. Tento formát použijeme, pokud chceme předat fungující blok někomu jinému a nechceme mu dát zdrojové soubory. 4: ILL Textový soubor, ale zapisují se mnemonické kódy instrukcí, do kterých je překládán formát STL. Dalo by se to přirovnat k assembleru. V současnosti není tento formát podporován.	⊙1 Long (I32)
<b>stack</b>	Velikost zásobníku pro všechny výpočty a proměnné. Zadává se jako počet proměnných, které se mají do zásobníku vejít. Výchozí hodnota 0 znamená, že velikost zásobníku se zvolí automaticky, což ve většině případů vyhovuje.	Long (I32)
<b>debug</b>	Úroveň/množství ladicích kontrol a informací; větší číslo znamená více kontrol a tím pomalejší běh algoritmu; volbu bez kontroly se doporučuje nepoužívat (může vést až k pádu aplikace na cílové platformě při nesprávně napsaném kódu).	Long (I32)
		⊙3
	1 ..... bez kontroly 2 ..... základní kontrola 3 ..... úplná kontrola	
<b>strs</b>	Velikost paměti (zásobníku) pro všechny texty. Zadává se jako počet byte/znaků, které se mají do zásobníku vejít. Výchozí hodnota 0 znamená, že velikost zásobníku se zvolí automaticky, což ve většině případů vyhovuje.	Long (I32)
<b>p0..p15</b>	Parametry, jejichž hodnoty jsou přístupné ze skriptu	Any

### Příklad C-like

Následující příklad ukazuje jednoduchý kód pro sečtení dvou vstupních signálů a také sečtení dvou uživatelsky definovaných parametrů.

```
double input(0) input_u0;
double input(2) input_u2;

double parameter(0) param_p0;
double parameter(1) param_p1;
```

```

double output(0) output_y0;
double output(1) output_y1;

double my_value;

long init(void)
{
    my_value = 3.14;
    return 0;
}

long main(void)
{
    output_y0 = input_u0 + input_u2;
    output_y1 = param_p0 + param_p1 + my_value;
    return 0;
}

long exit(void)
{
    return 0;
}

```

## Příklad STL

A zde je stejný příklad ve strukturovaném textu.

```

VAR_INPUT
    input_u0:REAL;
    input_u1:REAL;
    input_u2:REAL;
END_VAR

VAR_OUTPUT
    output_y0:REAL;
    output_y1:REAL;
END_VAR

VAR_PARAMETER
    param_p0:REAL;
    param_p1:REAL;
END_VAR

VAR
    my_value: REAL;

```

```
END_VAR
```

```
FUNCTION init : INT;  
    my_value := 3.14;  
    init := 0;  
END_FUNCTION
```

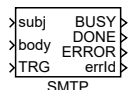
```
FUNCTION main : INT;  
    output_y0 := input_u0 + input_u2;  
    output_y1 := param_p0 + param_p1 + my_value;  
    main := 0;  
END_FUNCTION
```

```
FUNCTION exit : INT;  
    exit := 0;  
END_FUNCTION
```

## SMTP – \* Blok pro odesílání e-mailových oznámení přes SMTP

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

### Vstupy

subj	Předmět e-mailu	String
body	Tělo e-mailu	String
TRG	Spuštění zvolené akce	Bool

### Parametry

server	Adresa SMTP serveru	String
to	E-mail příjemce	String
from	E-mail odesílatele	String
tls	Metoda šifrování	⊙1 Long (I32)
	1 ..... None	
	2 ..... StartTLS	
	3 ..... TLS	
user	Uživatelské jméno	String
password	Heslo	String
domain	domain	String
auth	Metoda autentifikace	⊙1 Long (I32)
	1 ..... Login	
	2 ..... Plain	
certificate	Certifikát pro autentifikaci	String
VERIFY	Povolení verifikace serveru (platnost certifikátu)	Bool
timeout	Povolená doba pro dokončení operace	Double (F64)
BLOCKING	Čekání na dokončení operace	Bool

### Výstupy

BUSY	Odesílání e-mailu	Bool
DONE	E-mail byl odeslán	Bool
ERROR	Příznak chyby	Bool

errId	Kód chyby	Error
-------	-----------	-------

## STEAM – Přepočet vlastností páry

Symbol bloku

Licence: **STANDARD**



### Popis funkce

Block **STEAM** vypočítává různé termodinamické parametry vody a páry (například entalpii, entropii, teplotu a tlak sytosti, hustotu, viskozitu), přičemž vstupem jsou jiné termodinamické parametry (obvykle teplota a tlak, ale výpočty jsou obousměrné, takže existuje funkce určující teplotu z entalpie a podobně). Výpočet probíhá dle standardu IAPWS IF-97 (detaily lze nalézt na <http://www.iapws.org/relguide/IF97-Rev.pdf>). Jednotky v jakých výpočet probíhá lze pro teplotu zvolit parametrem **tunit** a pro tlak parametrem **tunit**; energie je v kilojoulech, tj. entalpie je v kJ/kg, tepelná kapacita v kJ/kg/K atd. (tak jak je definována v IF-97), ostatní veličiny v SI jednotkách (např. hustota v kg/m<sup>3</sup>). Vypočítávaná funkce má název ve tvaru <výstupní veličina>\_<1.vstupní veličina><2. vstupní veličina>, přičemž veličiny jsou:

- T Teplota
- p Tlak
- h Entalpie [kJ/kg]
- v Měrný objem [m<sup>3</sup>/kg]
- rho Hustota [kg/m<sup>3</sup>]
- s Měrná entropie
- u Měrná vnitřní energie [kJ/kg]
- Cp Měrná tepelná kapacita při konstantním tlaku [kJ/kg/K]
- Cv Měrná tepelná kapacita při konstantním objemu [kJ/kg/K]
- w Rychlost zvuku [m/s]
- my Viskozita
- tc Tepelná vodivost
- st Povrchové napětí
- x Hmotnostní podíl páry

- vx Objemový podíl páry

Výstupní veličina může mít modifikátor:

- sat saturační hodnota, tj. pro stav na přechodu mezi vodou a párou
- V pára (plyn) pro stav na přechodu mezi vodou a párou
- L voda (kapalina) pro stav na přechodu mezi vodou a párou

Příklady:

- h\_pT výstup je entalpie pro stav zadaný tlakem (1. vstup) a teplotou (2. vstup); např. pro tlak 100 kPa a teplotu 120 C je entalpie 2716 kJ/kg
- Tsat\_p výstup je teplota rovnovážného stavu mezi vodou a párou (tj. teplota varu) pro zadaný tlak (1. vstup); např. při tlaku 100kPa je teplota varu 100 C
- hL\_p výstup je entalpie za situace, že médium je voda na mezi sytosti a má zadaný tlak (1. vstup); např. pro 100 kPa (a teplotu 100 C, aby byl stav na mezi sytosti) může médium obsahovat libovolný poměr vody a páry (podle poměru bude různá entalpie), funkce (výstup bloku) bude entalpie pro situaci, že médium je voda (bey páry), tj. 417 kJ/kg

## Vstupy

u1	Hodnota 1. vstupní veličiny	Double (F64)
u2	Hodnota 2. vstupní veličiny (pokud ji funkce vyžaduje)	Double (F64)

## Parametry

func	Vypočítávaná funkce	⊙1 Long (I32)
punit	Jednotka tlaku	⊙1 Long (I32)
	1 ..... MPa	
	2 ..... bar	
	3 ..... kPa	
tunit	Jednotka teploty	⊙1 Long (I32)
	1 ..... K	
	2 ..... °C	

## Výstupy

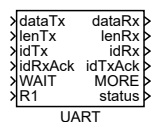
y	Hodnota výstupní veličiny	Double (F64)
E	Příznak chyba (výstupní hodnota je neplatná)	Bool



## UART – \* Komunikační blok pro UART

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>dataTx</code>	Reference vektoru s vysílanými daty		Reference
<code>lenTx</code>	Délka vysílaných dat (0 = celý vektor)	↓0	Long (I32)
<code>idTx</code>	ID vysílaného packetu	↓0	Long (I32)
<code>idRxAck</code>	ID posledního zpracovaného přijatého packetu	↓0	Long (I32)
<code>WAIT</code>	Příznak pozastavení vysílání (data jsou vložena do mezibufferu)		Bool
<code>R1</code>	Reset bloku		Bool
<code>dataRx</code>	Reference vektoru s přijatými daty		Reference
<code>lenRx</code>	Délka přijatých dat	↓0	Long (I32)
<code>idRx</code>	ID přijatého packetu	↓0	Long (I32)
<code>idTxAck</code>	ID posledního zpracovaného odeslaného packetu	↓0	Long (I32)
<code>MORE</code>	Příznak dalších dat v přijímacím mezibufferu		Bool
<code>status</code>	Indikátor vnitřního stavu		Long (I32)
	0 ..... Žádná chyba		
	-1 ..... Nepovedlo se otevřít port		
	1 ..... Přetečení vysílacího buferu		
	2 ..... Chyba odesílání dat		
	256 ... Chyba přijímání dat		

### Parametr

<code>port</code>	Název komunikačního zařízení		String
<code>baudrate</code>	Bitová rychlost [bit/s] (0 = nenastavovat)	↓0 ↑4000000	Long (I32)
<code>parity</code>	Parita		Long (I32)
	0 ..... Nenastaveno		
	1 ..... Bez parity		
	2 ..... Lichá parita		
	3 ..... Sudá parita		

<b>databits</b>	Počet datových bitů (0 = nenastavovat)	↓0 ↑3	Long (I32)
<b>stopbits</b>	Počet stop bitů (0 = nenastavovat)	↓0 ↑2	Long (I32)
<b>maxId</b>	Maximální hodnota ID packetu	↓2 ↑10000000 ⊕4	Long (I32)
<b>maxLen</b>	Maximální délka přijatých dat	↓1 ↑10000000 ⊕64	Long (I32)
<b>nmax</b>	Rezervovaná paměť pro pole	↓8 ↑10000000 ⊕256	Long (I32)

## Kapitola 17

# LANG – Speciální bloky

### Obsah

---

PYTHON – Volně programovatelný blok v jazyce Python . . . . .	508
---	-----

---

## PYTHON – Volně programovatelný blok v jazyce Python

Symbol bloku

Licence: [REXLANG](#)

>HLD	iE	>
>RESET	iRes	>
>u0	y0	>
>u1	y1	>
>u2	y2	>
>u3	y3	>
>u4	y4	>
>u5	y5	>
>u6	y6	>
>u7	y7	>
>u8	y8	>
>u9	y9	>
>u10	y10	>
>u11	y11	>
>u12	y12	>
>u13	y13	>
>u14	y14	>
>u15	y15	>
PYTHON		

### Popis funkce

V některých případech se může stát, že je do řídicího algoritmu nutné implementovat funkci, kterou nelze efektivně vytvořit z dostupné množiny bloků. Pro takový účel byl vyvinut blok **REXLANG**, který je vhodný zejména pro aplikace se striktními požadavky na běh programu v reálném čase. V opačném případě lze jako alternativu využít blok **PYTHON**.

Blok **PYTHON** implementuje algoritmus definovaný skriptovacím jazykem Python. V porovnání s blokem **REXLANG** nabízí snazší vývoj algoritmu a umožňuje rozšířit škálu funkcí poskytovaných systémem **REXYGEN** prostřednictvím balíčků a knihoven dostupných v ekosystému jazyka Python.

**Upozornění: blok PYTHON je určen pro prototypování a experimentální aplikace, proto prosím zvažte velmi pečlivě jeho použití ve vaší aplikaci. PYTHON je experimentální blok a vždy bude. Existují situace, které mohou vést k neočekávanému chování nebo dokonce zamrznutí exekutivy. Balíčky mohou být špatně napsány nebo mohou mít špatně implementovanou reinicializaci a jejich použití může vést až k pádu. Pro tento blok poskytujeme pouze velmi omezenou podporu.**

### Skriptovací jazyk

Skriptovacím jazykem bloku je standardní Python 3 (see [6]). Každý blok se odkazuje na skript napsaný v \*.py zdrojovém souboru. Zdrojový kód může volitelně obsahovat funkce s danými jmény, které jsou spouštěny systémem **REXYGEN**.

Funkce **main()** je spouštěna periodicky za běhu systému **REXYGEN**. Krom funkce **main()** jsou spouštěny funkce **init()** při startu řídicího algoritmu a po resetu bloku, funkce **exit()** při skončení řídicího algoritmu a před resetem bloku a funkce **parchange()** při změně hodnot parametrů bloku.

## Scripty na cílovém zařízení

Standardní interpret umožňuje načítání modulů/skriptů z různých lokací na cílovém zařízení. Blok PYTHON se může odkazovat na jakýkoliv skript dostupný standardnímu interpretu a navíc umožňuje načítání skriptů z adresáře `/rex/scripts/python`. Uživatelské skripty mohou být nahrávány přímo do této složky a nebo lze nastavit parametr `embedded` na hodnotu `on`. To způsobí, že referovaný skript je vložen do exekutivy při kompilaci a je následně dočasně extrahován do složky `/rex/scripts/python/embedded` při inicializaci bloku po nahrání a spuštění exekutivy na cílovém zařízení.

## API pro výměnu dat

Pro výměnu dat mezi interpretrem jazyka Python a systémem REXYGEN byl vyvinut modul `PyRexExt` v podobě nativního rozšíření interpretu. Modul obsahuje objekt `REX`, který obsluhuje veškeré operace výměny dat. Pro inicializaci API lze použít následující úryvek kódu na začátku skriptu.

```
from PyRexExt import REX
```

## I/O objekty

`REX.u0` - `REX.u15`

– objekty reprezentující *vstupy* bloku v prostředí jazyka Python

`REX.p0` - `REX.p15`

– objekty reprezentující *parametry* bloku v prostředí jazyka Python

`REX.y0` - `REX.y15`

– objekty reprezentující *výstupy* bloku v prostředí jazyka Python

## Čtení a zápis hodnot

Všechny I/O objekty obsahují atribut `v`. Čtení atributu `v` provádí konverzi z datových typů systému REXYGEN na datové typy jazyka Python. Vrácená hodnota pak může být uložena do proměnné a použita v algoritmu bloku. Pole systému REXYGEN jsou konvertovány na seznam hodnot v případě jednodimenzionálního pole, nebo na seznam seznamů v případě multidimenzionálního pole.

Příklad čtení hodnoty vstupu bloku:

```
x = REX.u0.v
```

Zápis atributu `v` naopak provádí konverzi z datových typů jazyka Python na datové typy systému REXYGEN a zápis hodnoty na příslušný výstup/parametr bloku.

Příklad zápisu hodnoty na výstup bloku:

```
REX.y0.v = 5
```

## Pole

Objekty vstupů a výstupů obsahují atribut `size`. Tento atribut je jen pro čtení a vrací počet typu `tuple` s počtem řádek a počtem sloupců.

S poli lze manipulovat prostřednictvím atributu `v`, ale tento přístup není příliš efektivní kvůli konverzi mezi poli a seznamy jazyk Python. Proto objekty vstupů a výstupů podporují operátor indexace `[]`, který provádí konverzi pouze dané položky `v` poli.

Příklad čtení vstupu bloku pro jednodimenzionální pole:

```
x = REX.u0[0]
```

Příklad zápisu hodnoty do multidimenzionálního pole výstupu bloku:

```
REX.u0[1, 3] = 5
```

## Externí položky

Objekt `REX` obsahuje metodu `Item`, která vrací handle na požadovanou externí položku systému `REXYGEN` specifikovanou přípojným řetězcem předaným v prvním parametru metody.

Příklad vytvoření externí položky a nastavení její hodnoty:

```
cns = REX.Item("myproject_task.CNS:scv")
cns.v = "abc"
```

## Trasování

Objekt `REX` obsahuje metody `Trace`, `TraceError`, `TraceWarning`, `TraceVerbose` and `TraceInfo`, které mohou být použity pro zápis zpráv do logu systému `REXYGEN`. Každá zpráva má k sobě připojený `stacktrace`.

Příklad logování zprávy:

```
REX.Trace("abc")
```

## Dodatečné vlastnosti

`REX.RexDataPath` – `RexDataPath` je konstantní řetězec obsahující cestu k datovému adresáři systému `REX` na dané platformě. Tato vlastnost může být užitečné při psaní platformově nezávislého kódu manipulujícímu se souborovým systémem pomocí absolutních cest.

## Vstupy

<code>HLD</code>	Hold – Algoritmus bloku není spuštěn při nastavení vstupu na <code>on</code> .	<code>Bool</code>
<code>RESET</code>	Náběžná hrana resetuje blok. Blok je opět inicializován, takže všechny globální proměnné jsou uvolněny a je zavolána funkce <code>init()</code> .	<code>Bool</code>
<code>u0..u15</code>	Vstupní signály, které jsou přístupné ve skriptu.	<code>Any</code>

## Výstupy

<code>iE</code>	Chybový kód. 0 ..... Žádná chyba nenastala, celá funkce <code>main()</code> byla úspěšně spuštěna. (včetně funkce <code>init()</code> ). xxx ... Chybový kód systému REXYGEN, viz Appendix C	Error
<code>iRes</code>	Výstupní kód exekuce bloku.	Long (I32)
<code>y0..y15</code>	Výstupní signály, které jsou dostupné ve skriptu.	Any

## Parametry

<code>srcname</code>	Jméno zdrojového souboru	<code>⊙program.py</code>	String
<code>embedded</code>	Vestavění skriptu	<code>⊙on</code>	Bool
<code>p0..p15</code>	Parametry bloku, které jsou dostupné ve skriptu.		Any

## Definování datových typů

Pro správnou výměnu dat mezi systémem REXYGEN a prostředím jazyka Python musí být striktně definované datové typy vstupních signálů `u0..u15`, výstupních signálů `y0..y15` a parametrů `p0..p15`.

Z toho důvodu musí být vytvořen konfigurační soubor pro každý skript se stejným názvem a s přidáním příponou `.cfg` (e.g. `program.py.cfg`). Pokud tento soubor chybí, je vytvořen při kompilaci projektu s tím, že všechny datové vstupy jsou nastaveny na defaultní hodnotu `double`. Nepředpokládá se, že by tento soubor byl editován přímo. Ke konfiguraci datových typů lze použít editor REXYGEN studia specifický pro blok PYTHON. Pro vstupy výstupy a parametry jsou dostupné datové typy `boolean`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, `int64`, `float`, `double`, `string` a pro vstupy a výstupy jsou navíc dostupné datové typy `array`, `numpy` a `image`.

Pro tyto datové typy `numpy` a `image` musí být na cílovém zařízení nainstalován balíček `numpy`. Pro vstupy typu `numpy` je předpokládáno, že vstupní signál je typu pole, které je následně překonvertováno na nativní reprezentaci objektu `numpy`. Pro vstupy typu `image` je předpokládáno, že vstupní signál je datového typu `image` z modulu `RexVision`, který je rovněž překonvertován na nativní reprezentaci typu `numpy` a může tedy být přímo použitý s `OpenCV` balíčkem jazyka Python.

Pro výstupy datového typu `numpy` je předpokládáno, že budou nastaveny ve skriptu pomocí objektu typu `numpy`, který je následně překonvertován na běžné pole. Pro výstupy datového typu `image` je předpokládáno, že budou nastaveny ve skriptu pomocí objektu typu `numpy`, který je následně překonvertován na objekt typu `image` definovaný modulem `RexVision`.

### Příklad definice datových typů

Následující příklad ukazuje zkrácenou verzi souboru ve formátu JSON, který popisuje datové typy vstupů, výstupů a parametrů bloku.

```
{
  "types": {
    "in": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ],
    "param": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ],
    "out": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ]
  }
}
```



## Příklad skriptu v jazyce Python

Následující příklad ukazuje jednoduchý zdrojový kód v jazyce Python, který sčítá dva vstupní signály a dva uživatelem definované parametry.

```
from PyRexExt import REX

def main():
    REX.y0.v = REX.u0.v + REX.u1.v
    REX.y1.v = REX.p0.v + REX.p1.v
    return
```

## Instalace - Debian

Prostředí jazyka Python by mělo být korektně nastaveno po instalaci debian balíčku PythonBlk\_T. Pro instalaci s volitelnými balíčky `numpy` a `OpenCV` je třeba spustit následující příkazy z terminálu.

```
sudo apt install rex-pythonblk
sudo apt install python3-numpy python3-opencv
```

## Instalace - Windows

Pro instalaci správné verze prostředí Python je doporučeno stáhnout a nainstalovat 64-bitovou verzi z [oficiálního repozitáře](https://www.python.org/ftp/python/3.9.6/) (<https://www.python.org/ftp/python/3.9.6/>). Při instalaci je vhodné se ujistit, že je zvolena možnost instalace programu `pip` a že cesta k binárním souborům interpretru bude přidána do systémové proměnné `PATH`.

Pro instalaci volitelných balíčků `numpy` a `OpenCV` je třeba spustit následující příkazy z příkazové řádky.

```
pip install numpy
pip install opencv-python
```

## Omezení

Vzhledem k omezením zakořeněným v implementaci standardního interpretru jazyka Python není doporučeno používat více instancí bloku `PYTHON` na rozdílných úrovních exekutivy. Takové použití může vést k nepředvídatelnému chování a nestabilitě programu `RexCore`.



## Kapitola 18

# DSP – Zpracování speciálních signálů

### Obsah

---

<b>BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu</b> . . . . .	<b>516</b>
<b>BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typu</b>	<b>518</b>
<b>BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu</b> . . . . .	<b>520</b>
<b>BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu</b> . . . . .	<b>521</b>
<b>BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu</b> . . . . .	<b>522</b>
<b>FFT – Rychlá Fourierova transformace</b> . . . . .	<b>523</b>
<b>MOSS – Přesný senzor pohybu</b> . . . . .	<b>525</b>
<b>PSD – Výkonová spektrální hustota</b> . . . . .	<b>527</b>

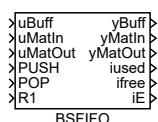
---

Knihovna DSP je přizpůsobena pro pokročilé zpracování digitálních signálů. Obsahuje bloky jako **FFT** pro operace rychlé Fourierovy transformace a **PSD** pro analýzu spektrální hustoty výkonu. Knihovna také nabízí **BSFIFO**, **BSGET**, **BSGETV**, **BSSET** a **BSSETV** pro ukládání do bufferu a jeho načítání, což umožňuje efektivní manipulaci s daty při úlohách zpracování signálů. Kromě toho **MOSS** poskytuje možnosti výběru signálů na základě více cílů. Tato kolekce bloků je nezbytná pro sofistikovanou analýzu a manipulaci se signály v digitálních systémech.

## BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Tento blok postupně přidává a vybírá data do velkého bufferu (přivedeného na vstup `uBuff`). Základní jednotka v bufferu je sloupec. Všechny matice (tj. matice nebo vektory přivedené na vstupy `uBuff`, `uMatIn`, `uMatOut`) musí mít stejnou velikost sloupce v bajtech. Data jsou organizována buď jako fronta (pokud `REV=off`) nebo jako zásobník (pokud `REV=on`). Chování bloku závisí na vstupech tímto způsobem:

- Pokud je `PUSH=on`, vkládá se do bufferu obsah matice `uMatIn` (všechny definované sloupce).
- Pokud je `POP=on`, vyjímá se z bufferu počet sloupců určených parametrem `col` a tato data se vloží do matice `uMatOut` (musí mít dostatečnou velikost).
- Pokud je `R1=on`, data se znovu načtou (hlavně počet platných sloupců) do bufferu bloku. Vlastní data se předávají odkazem a jsou proto sdílená. Tento signál je prioritní a blokuje signály `PUSH`, `POP`.

Chybové stavy (např. neodpovídající si rozměry matic, nedostatečné místo v některé matici, nedostatek dat v bufferu) jsou indikovány na výstupu `iE` a zprávou v `SystemLog`.

### Vstupy

<code>uBuff</code>	Binární struktura s daty (pole bajtů)	Reference
<code>uMatIn</code>	Reference matice se vstupními daty (pro <code>PUSH</code> )	Reference
<code>uMatOut</code>	Reference matice pro uložení výstupních dat (pro <code>POP</code> )	Reference
<code>PUSH</code>	Povolení vkládání do bufferu	Bool
<code>POP</code>	Povolení vyčítání z bufferu	Bool
<code>R1</code>	Reset bufferu (znovu načtení hlaviček z <code>uBuff</code> )	Bool

### Parametry

<code>OW</code>	Režim přepisu nejstarších položek	Bool
<code>REV</code>	Vyčítání posledních vložených položek nejdříve	Bool
<code>col</code>	Počet vyčítaných sloupců (pro <code>POP</code> )	⊙1 Long (I32)

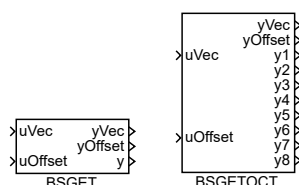
## Výstupy

yBuff	Kopie uBuff pro řetězení	Reference
yMatIn	Kopie uMatIn pro řetězení	Reference
yMatOut	Kopie uMatOut pro řetězení	Reference
iused	Počet použitých bajtů v bufferu	Long (I32)
ifree	Počet volných bajtů v bufferu	Long (I32)
iE	Kód chyby	Error

## BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typu

Symboly bloků

Licence: [ADVANCED](#)



### Popis funkce

Tato skupina bloků slouží pro získávání hodnot z binární struktury (bajtového pole). Pro zápis do binární struktury lze použít bloky [BSSET](#) a [BSSETOCT](#).

Pokud binární struktury přijdou po komunikaci, je možné je zpracovat přímo v bloku zprostředkovávajícím komunikaci. Typicky se jedná o programovatelný blok [REXLANG](#) nebo [PYTHON](#). Pomocí struktur je však možné předávat data i v rámci aplikace [REXYGEN](#). Binární struktura se přivede ve formě pole (vektoru) bajtů na vstup `uVec`. Vstup `uOffset` udává posunutí (v bajtech) požadované hodnoty od začátku struktury. Typ hodnoty udává parametr `type`.

Výstup `yOffset` je začátek následujícího prvku ve struktuře. To je výhodné pro řetězení: pokud struktura obsahuje několik prvků za sebou, je možné zapojit vstup `uOffset` na výstup `yOffset` předchozího bloku a není nutné posunutí dopočítávat.

Bloky se liší jen v tom, že [BSGET](#) získává jednu hodnotu. Blok [BSGETOCT](#) je schopný získat až 8 hodnot (počet určuje parametr `m`).

### Vstupy

<code>uVec</code>	Struktura s daty (pole bajtů)	Reference
<code>uOffset</code>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)

### Výstupy

<code>yVec</code>	Kopie <code>uVec</code> pro řetězení	Reference
<code>y</code>	Požadovaná hodnota ze struktury (typ hodnoty je definován parametrem)	Any
<code>yOffset</code>	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	Long (I32)

### Parametry

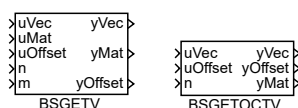
<code>m</code>	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	Long (I32)
----------------	--	----------	------------

<b>BE</b>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)	<b>Bool</b>
<b>type</b>	Datový typ hodnoty	↓2 ↑10 ⊙2 <b>Long (I32)</b>

## BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu

Symboly bloků

Licence: [ADVANCED](#)



### Popis funkce

Tato skupina bloků slouží pro získávání hodnot z binární struktury (bajtového pole). Pro zápis do binární struktury lze použít bloky [BSSETV](#) a [BSSETOCTV](#).

Význam většiny parametrů je stejný jako u bloku [BSGET](#), ale tyto bloky načítají několik hodnot stejného typu a ukládají je do pole (matice). Matice má vždy  $m$  řádek a  $n$  sloupců. U bloku [BSGETV](#) jsou všechny prvky stejného typu (určuje parametr `type`) a data se vyplňují do matice přivedené na vstup `uMat`. Blok [BSGETOCTV](#) načítá až 8 vektorů. Každý řádek matice může být jiného typu. Blok alokuje matici sám. Matice je na výstupu `yMat`.

### Vstupy

Parametr	Popis	Typ
<code>uVec</code>	Struktura s daty (pole bajtů)	Reference
<code>uMat</code>	Matice pro uložení hodnot	Reference
<code>uOffset</code>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)
<code>n</code>	Počet sloupců výstupní matice	Long (I32)
<code>m</code>	Počet řádek výstupní matice	Long (I32)

### Výstupy

Parametr	Popis	Typ
<code>yVec</code>	Kopie <code>uVec</code> pro řetězení	Reference
<code>yMat</code>	Kopie <code>uMat</code> pro řetězení	Reference
<code>yOffset</code>	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	Long (I32)

### Parametry

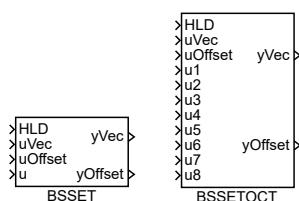
Parametr	Popis	Typ
<code>m</code>	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8 Long (I32)
<code>BE</code>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)	Bool
<code>nmax</code>	Alokovaná velikost výstupní matice (celkový počet prvků) <code>yMat</code>	↓1 ⊙32 Long (I32)
<code>type</code>	Datový typ hodnoty	↓2 ↑10 ⊙2 Long (I32)



## BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu

Symboly bloků

Licence: [ADVANCED](#)



### Popis funkce

Tato skupina bloků slouží pro nastavování hodnot do binární struktury (bajtového pole). Funkce je inverzní bloku [BSGET](#) a [BSGETOCT](#), tj. všechny signály mají stejný význam, jen data se kopírují opačným směrem - ze vstupu `u` do binární struktury reprezentované polem bajtů připojeným ke vstupu `uVec`. Blok modifikuje binární strukturu jen pokud je `HLD=off`.

### Vstupy

<code>HLD</code>	Potlačení zápisu	Bool
<code>uVec</code>	Struktura s daty (pole bajtů)	Reference
<code>uOffset</code>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)
<code>u</code>	Zapisovaná hodnota do struktury (typ hodnoty definován parametrem)	Any

### Výstupy

<code>yVec</code>	Kopie <code>uVec</code> pro řetězení	Reference
<code>yOffset</code>	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	Long (I32)

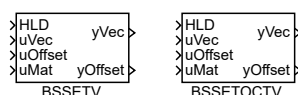
### Parametry

<code>m</code>	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	Long (I32)
<code>BE</code>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		Bool
<code>type</code>	Datový typ hodnoty	↓2 ↑10 ⊙2	Long (I32)

## BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu

Symboly bloků

Licence: [ADVANCED](#)



### Popis funkce

Tato skupina bloků slouží pro nastavení matice hodnot do binární struktury (bajtového pole). Funkce je inverzní bloku [BSGETV](#) a [BSGETOCTV](#), tj. všechny signály mají stejný význam, jen data se kopírují opačným směrem - ze vstupu `uMat` do binární struktury reprezentované polem bajtů připojeným ke vstupu `uVec`. Blok modifikuje binární strukturu jen pokud je `HLD=off`.

Na rozdíl od bloku [BSGETV](#) se počet řádek a sloupců nezadává, ale určuje se ze skutečné velikosti matice připojené na vstup `uMat`.

### Vstupy

<code>HLD</code>	Potlačení zápisu	Bool
<code>uVec</code>	Struktura pro vyplnění daty z matice (pole bajtů)	Reference
<code>uOffset</code>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)
<code>uMat</code>	Matice s daty	Reference

### Výstupy

<code>yVec</code>	Kopie <code>uVec</code> pro řetězení	Reference
<code>yOffset</code>	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	Long (I32)

### Parametry

<code>m</code>	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	Long (I32)
<code>BE</code>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		Bool
<code>type</code>	Datový typ hodnoty	↓2 ↑10 ⊙2	Long (I32)

## FFT – Rychlá Fourierova transformace

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok FFT (Fast Fourier Transform) slouží pro výpočet rychlé Fourierovy transformace pomocí balíku PocketFFT [7]. PocketFFT implementovaný v C99 vychází z balíku FFTPack (ve Fortranu) [8], který byl vyvinut na základě kapitoly z [9].

Vstup `uc` odkazuje na vstupní data (vektor nebo matice) tohoto bloku. Pokud vstup `uc` odkazuje na sloupcový vektor (počet sloupců  $m = 1$ ) s počtem prvků  $n$ , pak bude FFT počítána z jednoho signálu s  $n$  vzorky. Pokud vstup `uc` odkazuje na matici s  $n$  řádky a  $m$  sloupci, pak bude FFT počítána  $m$  krát (jednou pro každý sloupec).

Vstupní data jsou zpracovávána podle parametru `mode` (viz níže), který určuje, zda bude FFT počítána pro reálná nebo komplexní data a přímou nebo zpětnou transformaci. Přesněji, parametr `mode` určuje, jaká z funkcí `rfft_forward()`, `rfft_backward()`, `cffft_forward()` nebo `cffft_backward()` z balíku PocketFFT bude volána.

Vstup `uf` odkazuje na výstupní data, která mají stejný počet  $n$  řádků a  $m$  sloupců jako data, na něž odkazuje `uc`. Pokud je vstup `uf` připojen na předalokovaný vektor/matice pak na výstupní data FFT algoritmu odkazuje výstup `yf`. Pokud není vstup `uf` připojen, počítá se FFT na místě a výstupní data jsou ukládána v poli, na něž se odkazuje výstup `yc`, a tedy data odkazovaná vstupem `uc` jsou přepisována.

Odkaz `uc` je kopírován na `yc`, odkaz `uf` je kopírován na `yf`.

Vstup `HLD` umožňuje dočasně zastavit výpočet FFT.

### Vstupy

<code>uc</code>	Vstupní odkaz na vstupní/výstupní data	Reference
<code>uf</code>	Vstupní odkaz na nepovinná výstupní data	Reference
<code>HLD</code>	Hodnota <code>HLD=on</code> zastavuje výpočet FFT	Bool

### Parametry

<code>mode</code>	Výpočetní mód FFT	↓1 ↑4 ⊙1 Long (I32)
1	..... Reálná dopředná FFT	
2	..... Reálná zpětná FFT	
3	..... Komplexní dopředná FFT	
4	..... Komplexní zpětná FFT	

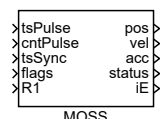
## Output

yc	Výstupní odkaz na vstupní/výstupní data	Reference
yf	Výstupní odkaz na nepovinná výstupní data	Reference
E	Příznak chyby	↓0 ↑1000000000 Long (I32)

## MOSS – Přesný senzor pohybu

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok implementuje pokročilý filtr pro inkrementální (kvadrurní) senzory polohy. Blok vyžaduje speciální hardware, neboť pro správnou činnost je nutné znát nejen aktuální hodnotu z čidla, ale také časovou značku posledního pulsu, směr pohybu při posledním pulzu, časovou značku referenčního okamžiku (ze stejného zdroje jako časová značka pulzu). Výstupem bloku je nejen filtrovaná poloha, ale také rychlost a zrychlení. Pro správnou činnost je potřeba vhodně zvolit parametr **alpha**. Menší hodnota snižuje šum, ale zvětšuje zpoždění signálu.

Poznámka 1: Zdánlivě nelze polohu určit přesněji než kvantizační chyba (+-1 pulz) měření, ale znalost rychlosti umožňuje odhadovat polohu přesněji. Dále se může zdát, že nelze určit rychlost přesněji než jako podíl počtu pulzů a rozdílu časových značek, ale vložím obou signálů do grafu je vidět, že MOSS signál zlepšuje. A určovat zrychlení diferencí vede na naprosto nepoužitelné hodnoty.

Poznámka 2: Interně blok realizuje Kalmanův filtr pro systém se 2 integrátory (tj. vstup zrychlení, výstup poloha). Filtr je diskretizován v každé periodě znova, přičemž diskretizační perioda je aktuální rozdíl časových značek. Pro odvození filtru je potřeba znát vstupní a výstupní poruchy. Pokud uvažujeme bílý (gausovský) šum, stačí znát jejich poměr a to je parametr **alpha**.

### Vstupy

<b>tsPulse</b>	Časová značka posledního pulsu	DWord (U32)
<b>cntPulse</b>	Poslední puls (stav čítače)	DWord (U32)
<b>tsSync</b>	Časová značka synchronizačního pulsu. Čas synchronizačního pulsu je okamžik, ke kterému je určován výstup (obvykle okamžik spuštění tasku systému REXYGEN)	DWord (U32)
<b>flags</b>	Vstupní příznaky (1: POS, 2: NEG, 4: RUN)	DWord (U32)
<b>R1</b>		DWord (U32)

### Parametry

<b>freq</b>	Frekvence zdroje časové značky [Hz]	↓0.0 ⊕100000000.0	Double (F64)
-------------	-------------------------------------	-------------------	--------------

<b>stall</b>	Čas detekce zastavení [s]. Pokud nepřijde pulz více než <b>stall</b> sekund, je to vyhodnoceno jako zastavení (tj. výstupní rychlost i zrychlení je 0) ↓0.0 ⊖0.08	Double (F64)
<b>alpha</b>	Návrhový parametr kalmanova filtru. Menší hodnota vede na menší šum na výstupu, ale signál je více zpožděn a zprůměrován. Musí se nastavit kompromisně podle aplikace. ↓0.0 ↑200.0 ⊖26.0	Double (F64)
<b>maxpos</b>	Parametr pro optimalizaci zaokrouhlovacích chyb. Pokud je poloha velké číslo, dochází při výpočtu k významným zaokrouhlovacím chybám. Aby se tomu zamezilo, tak pokud je poloha větší než <b>maxpos</b> odečte se celočíselný násobek tohoto parametru, s takto sníženou hodnotou se provede výpočet a na konec se odečtená hodnota zase přičte. Parametr by se neměl měnit. ↓0.0 ⊖1e+10	Double (F64)
<b>mindivert</b>	Minimální doba pro omezení odchylky [s]. Když pulz nepřijde dlouhou dobu, výstupní poloha "driftuje". Aby se tomu zamezilo, tak pokud pulz nepřijde déle než <b>mindivert</b> je kontrolována výstupní poloha a pokud se liší více než +-1 od změřené, použije se náhradní (změřená) poloha. ↓0.0 ⊖0.003	Double (F64)

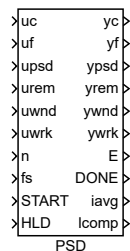
## Výstupy

<b>pos</b>	Filtrovaná poloha	Double (F64)
<b>vel</b>	Filtrovaná rychlost	Double (F64)
<b>acc</b>	Filtrované zrychlení	Double (F64)
<b>status</b>	Výstupní stavové příznaky (1: POS, 2: NEG, 4: RUN, 8: INIT, 16: PULSE, 32: STALLED, 64: DIVERT)	Long (I32)
<b>iE</b>		Error

## PSD – Výkonová spektrální hustota

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Funkční blok PSD (Power Spectral Density) slouží pro výpočet výkonové spektrální hustoty (PSD, periodogramu) pomocí balíku PocketFFT [7]. PocketFFT implementovaný v C99 vychází z balíku FFTPack (ve Fortranu) [8], který byl vyvinut na základě kapitoly z [9].

PSD se nejčastěji počítá ze segmentů délky  $N$  vzorků vstupního signálu. Data těchto segmentů jsou vynásobena váhovou funkcí (window function), též nazývanou oknem (např. Hammingovo okno). Z takto vážených dat je vypočtena rychlá Fourierova transformace (FFT), a z ní pak spektrální výkonová hustota (PSD) každého segmentu. Nakonec se zprůměruje `navg` těchto výsledků (viz parametr `navg`). Vstupní signál je rozdělen do segmentů následujících po sobě nebo se segmenty vždy o polovinu překrývají (Welchova metoda [10], [11]). Výsledná PSD může být vypočtena v decibelech (dB), pokud je parametr `DB = on`.

Vstup `uc` odkazuje na vstupní data (vektor nebo matice) tohoto bloku. Pokud vstup `uc` odkazuje na sloupcový vektor (počet sloupců  $m = 1$ ) s počtem prvků  $n$ , pak bude PSD počítána z jednoho signálu s  $n$  vzorky. Pokud vstup `uc` odkazuje na matici s  $n$  řádky a  $m$  sloupci, pak bude PSD počítána  $m$  krát (jednou pro každý sloupec). Pokud je vstup  $n > 0$ , pak bude počet vzorků segmentu  $N = n$ , jinak bude  $N$  nastaveno na počet sloupců pole, na něž odkazuje vstup `uc`.

Vstup `uf` odkazuje na interní vektor dimenze  $\geq N$  pro výpočet FFT. Vstup `upsd` odkazuje na výslednou PSD s `npsd` řádky a  $m$  sloupci. Počet frekvencí PSD `npsd = N/2+1` pro sudé  $N$  nebo `npsd = (N + 1)/2` pro liché  $N$ . Vstup `uwrk` odkazuje na pracovní pole se stejnými rozměry jako pole odkazované vstupem `upsd`. Pokud vstupní data obsahují víc prvků než může být zpracováno v jednom spuštění bloku PSD, jsou zbývající data zkopírována do pole odkazovaného vstupem `urem` naalokovaného s  $N$  řádky a  $m$  sloupci.

Zpracování vstupních dat je určeno parametry `mode` a `mtrig` (viz níže). Parametr `mode` určuje způsob výpočtu PSD, podle toho, zda vstupní data odkazovaná vstupem `uc` jsou reálná nebo komplexní a zda bude použito překrývání segmentů o polovinu. Vstupní data vzorkovaná s frekvencí připojenou na vstup `fs` mohou být získávána v jiné

úloze (např. s podstatně kratší periodou vzorkování) než v té, ve které je počítána PSD. Způsob spouštění výpočtu PSD je určen parametrem `mtrig` (trigger mode). Výpočet PSD může být spouštěn každou periodu běhu bloku PSD, v případě, že ve vstupním poli je dostatečný počet vzorků (alespoň rovný hodnotě  $N$ ), nebo náběžnou hranou vstupu `START`.

Na vstup `uwnd` je připojen vektor dimenze  $N$ , do kterého se uloží hodnoty váhové funkce okna, zvoleného parametrem `iwin`. Váhové funkce okna mohou mít (volitelně) celočíselný parametr `lwnd` a/nebo reálný parametr `rwnd`. Některá okna lze používat ve dvou variantách:

**Symetrické** (pro `lwnd=0`). Je vypočteno okno délky  $L = N$  vzorků, první a poslední vzorek je shodný. Tato varianta je vhodná pro návrh filtrů s konečnou impulzní odezvou (Finite Impulse Response, FIR).

**Periodické** (pro `lwnd=1`). Je vypočteno okno délky  $L = N + 1$  vzorků, přičemž se uloží prvních  $N$  vzorků. Periodická verze se přednostně používá pro spektrální analýzu, protože diskretní Fourierova transformace předpokládá periodický průběh vstupního vektoru.

Váhové funkce oken jsou definovány následujícími výrazy:

`iwin=1` **Bartlettovo-Hahnovo okno** (Bartlett-Hahn Window)

$$w(n) = 0.62 - 0.48 \left| \left( \frac{n}{N-1} - 0.5 \right) \right| + 0.38 \cos \left[ 2\pi \left( \frac{n}{N-1} - 0.5 \right) \right], \quad 0 \leq n \leq N-1.$$

`iwin=2` **Bartlettovo okno** (Bartlett Window)

$$w(n) = \begin{cases} \frac{2n}{N}, & 0 \leq n \leq \frac{N}{2}, \\ 2 - \frac{2n}{N}, & \frac{N}{2} \leq n \leq N. \end{cases}$$

`iwin=3` **Blackmanovo okno** (Blackman Window)

$$w(n) = 0.42 - 0.5 \cos \left( \frac{2\pi n}{L-1} \right) + 0.08 \cos \left( \frac{4\pi n}{L-1} \right), \quad 0 \leq n \leq M-1,$$

kde  $M$  je  $N/2$  pro sudé  $N$  a  $(N+1)/2$  pro liché  $N$ .

`iwin=4` **Blackmanovo-Harrisovo okno** (Blackman-Harris Window)

$$w(n) = a_0 - a_1 \cos \left( \frac{2\pi n}{L-1} \right) + a_2 \cos \left( \frac{4\pi n}{L-1} \right) - a_3 \cos \left( \frac{6\pi n}{L-1} \right), \quad 0 \leq n \leq N-1$$

kde  $a_0 = 0.35875$ ,  $a_1 = 0.48829$ ,  $a_2 = 0.14128$  a  $a_3 = 0.01168$ .



iwin=5 **Bohmanovo okno** (Bohman Window)

$$w(x) = (1 - |x|) \cos(\pi |x|) + \frac{1}{\pi} \sin(\pi |x|), \quad -1 \leq x \leq 1$$

iwin=6 **Čebyševovo okno** (Chebyshev Window)

Použitím Čebyševových polynomů  $n$ -tého řádu

$$T_n(x) \triangleq \begin{cases} \cos [n \cos^{-1}(x)], & |x| \leq 1, \\ \cosh [n \cosh^{-1}(x)], & |x| > 1, \end{cases}$$

je Fourierova transformace Čebyševova okna

$$W(k) = \frac{T_{N-1}[x_0 \cos(\pi k/N)]}{T_{N-1}(x_0)}$$

kde

$$x_0 = \cosh \left[ \frac{\cosh^{-1}(10^{r/20})}{N-1} \right].$$

iwin=7 **Okno s plochým vrcholem** (Flat Top Window)

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{L-1}\right) + a_2 \cos\left(\frac{4\pi n}{L-1}\right) - a_3 \cos\left(\frac{6\pi n}{L-1}\right) + a_4 \cos\left(\frac{8\pi n}{L-1}\right),$$

kde  $0 \leq n \leq N-1$  and  $a_0 = 0.21557895$ ,  $a_1 = 0.41663158$ ,  $a_2 = 0.277263158$ ,  $a_3 = 0.083578947$  a  $a_4 = 0.006947368$ .

iwin=8 **Gaussovo okno** (Gauss Window)

$$w(n) = e^{-\frac{1}{2} \left[ \frac{n}{(N-1)/2} \right]^2} = e^{-n^2/(2\sigma^2)}$$

kde  $-(N-1)/2 \leq n \leq (N-1)/2$  a  $\sigma = (N-1)/(2\alpha)$  je směrodatná odchylka Gaussovy funkce hustoty pravděpodobnosti.

iwin=9 **Hammingovo okno** (Hamming Window)

$$w(n) = 0.54 - 0.46 \cos\left(2\pi \frac{n}{L-1}\right), \quad 0 \leq n \leq N-1,$$

iwin=10 **Hannovo okno** (Hann Window)

$$w(n) = 0.5 \left[ 1 - \cos\left(2\pi \frac{n}{L-1}\right) \right], \quad 0 \leq n \leq N-1,$$

**iwin=12 Nuttallova varianta Blackmanova-Harrisova okna** (Nuttall's Blackman-Harris Window)

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{L-1}\right) + a_2 \cos\left(\frac{4\pi n}{L-1}\right) - a_3 \cos\left(\frac{6\pi n}{L-1}\right), \quad 0 \leq n \leq N-1$$

kde  $a_0 = 0.3635819$ ,  $a_1 = 0.4891775$ ,  $a_2 = 0.1365995$  a  $a_3 = 0.0106411$ .

**iwin=13 Parzenovo okno** (Parzen Window)

$$w(n) = \begin{cases} 1 - 6\left(\frac{|n|}{N/2}\right)^2 + 6\left(\frac{|n|}{N/2}\right)^3, & 0 \leq |n| \leq (N-1)/4, \\ 2\left(1 - \frac{|n|}{N/2}\right)^3, & (N-1)/4 < |n| \leq (N-1)/2. \end{cases}$$

**iwin=14 Obdélníkové okno** (Rectangular Window)

$$w(n) = 1, \quad 0 \leq n \leq N-1.$$

Tento blok neprovádí násobení všech datových vzorků hodnotou 1, a proto nemusí být vstup `uwnd` připojen (vektor okna se nepoužívá).

**iwin=16 Trojúhelníkové okno** (Triangular Window)

Pro liché  $N$ :

$$w(n) = \begin{cases} \frac{2n+2}{N+1}, & 0 \leq n \leq (N-1)/2, \\ 2 - \frac{2n+2}{N+1}, & (N-1)/2 < n \leq (N-1). \end{cases}$$

Pro sudé  $N$ :

$$w(n) = \begin{cases} \frac{2n+1}{N}, & 0 \leq n \leq N/2 - 1, \\ 2 - \frac{2n+1}{N}, & N/2 \leq n \leq (N-1). \end{cases}$$

**iwin=17 Tukeyovo okno** (Tukey Window)

$$w(x) = \begin{cases} \frac{1}{2} \left\{ 1 + \cos\left(\frac{2\pi}{r} [x - r/2]\right) \right\}, & 0 \leq x < \frac{r}{2}, \\ 1, & \frac{r}{2} \leq x < 1 - \frac{r}{2}, \\ \frac{1}{2} \left\{ 1 + \cos\left(\frac{2\pi}{r} [x - 1 + r/2]\right) \right\}, & 1 - \frac{r}{2} \leq x < 1, \end{cases}$$

kde  $r$  je parametrem `rwnd`, s implicitní hodnotou 0.5.

**iwin=18 Exponenciální okno** (Exponential Window)

$$w(x) = e^{-\frac{|x|}{\tau}}, \quad -1 \leq x \leq 1,$$

kde  $\tau$  je parametr okna (nastavený v `rwnd`).

**iwin=19 Welchovo okno** (Welch Window)

$$w(x) = 1 - x^2, \quad -1 \leq x \leq 1.$$

**iwin=20 Externě definované okno**

Pokud není uživatel spokojen s žádným s předdefinovaných oken, může si zadat své vlastní okno pomocí `N` vzorků ve vektoru odkazovaném vstupem `uwnd` předtím, než je spuštěn blok PSD.

Vstup `HLD` umožňuje dočasně zastavit výpočet PSD.

**Vstupy**

<code>uc</code>	Vstupní odkaz na vstupní data	Reference
<code>uf</code>	Vstupní odkaz na interní vektor pro FFT	Reference
<code>upsd</code>	Vstupní odkaz na výstupní vektor/matici spektrální výkonové hustoty	Reference
<code>urem</code>	Vstupní odkaz na zbývající (nezpracovaná) data	Reference
<code>uwnd</code>	Vstupní odkaz na vektor váhové (window) funkce	Reference
<code>uwrk</code>	Vstupní odkaz na pracovní vektor/matici	Reference
<code>n</code>	Počet vzorků signálu v segmentu FFT	Long (I32)
<code>fs</code>	Frekvence vzorkování v [Hz]	Double (F64)
<code>START</code>	Start výpočtu (náběžná hrana)	Bool
<code>HLD</code>	Hodnota <code>HLD=on</code> zastavuje výpočet PSD	Bool

**Parametry**

<code>mode</code>	Výpočetní mód spektrální výkonové hustoty	↓1 ↑4 ⊙1	Long (I32)
	1 ..... Reálná		
	2 ..... Reálná s překryvem		
	3 ..... Komplexní		
	4 ..... Komplexní s překryvem		
<code>mtrig</code>	Spouštěcí mód výpočtu	↓1 ↑3 ⊙1	Long (I32)
	1 ..... Každá perioda		
	2 ..... Dostatek vzorků		
	3 ..... Náběžná hrana <code>START</code>		

<code>iwnd</code>	Váhová funkce (window)	↓1 ↑9 ⊙1	Long (I32)
	1 ..... Bartlett-Hann	11 .... —	
	2 ..... Bartlett	12 .... Nuttall	
	3 ..... Blackman	13 .... Parzen	
	4 ..... Blackman-Harris	14 .... Obdélníkové	
	5 ..... Bohman	15 .... —	
	6 ..... Čebyšev	16 .... Trojúhelníkové	
	7 ..... Flat top	17 .... Tukey	
	8 ..... Gaussian	18 .... Exponenciální	
	9 ..... Hamming	19 .... Welch	
	10 .... Hann	20 .... Externí	
<code>lwnd</code>	Parametr <code>lwnd</code> některých váhových funkcí	↓1 ↑9 ⊙1	Long (I32)
<code>rwnd</code>	Parametr <code>rwnd</code> některých váhových funkcí	↓1 ↑9 ⊙1	Long (I32)
<code>navg</code>	Počet segmentů FFT pro průměrování	↓1 ⊙10	Long (I32)
<code>DB</code>	Vypočtená spektrální výkonová hustota je konvertována do decibelů [dB]		Bool

## Output

<code>yc</code>	Výstupní odkaz na vstupní data	Reference
<code>yf</code>	Výstupní odkaz na interní vektor pro FFT	Reference
<code>ypsd</code>	Výstupní odkaz na výstupní vektor/matici spektrální výkonové hustoty	Reference
<code>yrem</code>	Výstupní odkaz na zbývající (nezpracovaná) data	Reference
<code>ywnd</code>	Výstupní odkaz na externí váhovou (window) funkci	Reference
<code>ywrk</code>	Výstupní odkaz na pracovní vektor/matici	Reference
<code>E</code>	Příznak chyby	Bool
<code>DONE</code>	Příznak dokončení výpočtu průměrováním	Bool
<code>iavg</code>	Aktuální index segmentu pro průměrování	Long (I32)
<code>lcomp</code>	Počet úspěšných výpočtů spektrální výkonové hustoty	Large (I64)

## Kapitola 19

# MQTTDrv – Komunikace přes MQTT protokol

### Obsah

---

<a href="#">MqttPublish – Odeslání zprávy protokolem MQTT</a>	534
<a href="#">MqttSubscribe – Odběr zpráv z MQTT topic</a>	536

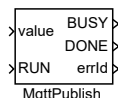
---

Knihovna MQTTDrv je určena pro komunikaci IoT (Internet of Things) pomocí protokolu MQTT (Message Queuing Telemetry Transport). Skládá se ze dvou primárních bloků: [MqttPublish](#) a [MqttSubscribe](#). Blok [MqttPublish](#) se používá pro odesílání zpráv brokerovi MQTT, umožňující publikování dat do MQTT témat. Naopak blok [MqttSubscribe](#) je určen pro přihlášení k odběru témat a příjem zpráv od brokera. Tato knihovna usnadňuje efektivní a efektivní datovou komunikaci v aplikacích IoT a využívá lehký a široce používaný protokol MQTT pro výměnu zpráv.

## MqttPublish – Odeslání zprávy protokolem MQTT

Symbol bloku

Licence: [MQTT](#)



### Popis funkce

Tento funkční blok je závislý na MQTT ovladači. Je doporučeno si před použitím přečíst manuál MQTTDrv ovladače [12].

Blok `MqttPublish` slouží pro odesílání zpráv na zařízení typu *MQTT Broker* prostřednictvím připojení, které zajišťuje MQTTDrv ovladač.

Prvním parametrem jménem *topic* se určuje do jakého tématického celku budou zprávy produkované blokem zařazeny. Protokol MQTT doručuje aplikační zprávy dle zvolené úrovně kvality služby (*Quality of Service – QoS*). Požadovaná úroveň lze nastavit parametrem `QoS`. Více informací naleznete ve specifikaci MQTT protokolu [13] (pouze anglicky).

Pokud je parametr `RETAIN` nahozen, odchozí pakety zprávy budou označeny příznakem `RETAIN`. Více informací naleznete ve specifikaci MQTT protokolu [13] (pouze anglicky).

Parametr `defBuffSize` může být použit pro optimalizaci práce s pamětí bloku. Hodnota parametru představuje velikost staticky alokované paměti pro vnitřní buffer odchozích zpráv. Pokud je velikost bufferu nadbytečně velká, blok si alokuje paměť, která je zbytečně blokována. Na druhou stranu pokud je hodnota parametru příliš malá, algoritmus bloku musí často dynamicky alokovat paměť, což je časově náročná operace.

Odesílaná zpráva je konstruována ze vstupního signálu `value`. Blok předpokládá, že vstup `value` bude datového typu *string*. Pokud je vstup jiného typu, dojde k automatické konverzi. Pro odeslání zprávy v aktuální periodě stačí nastavit vstup `RUN` na `on`. Výstup `BUSY` má hodnotu `on` pokud je blok zaneprázdněn nevyřízeným požadavkem pro odeslání zprávy nebo vyčkává na odpověď od zařízení typu *Broker*. Pokud je požadavek pro odeslání úspěšně vyřízen v aktuální periodě, dojde k nastavení výstupu `DONE` na `on`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<code>value</code>	Vstupní signál	String
<code>RUN</code>	Povolení běhu algoritmu	Bool

## Parametr

<code>topic</code>	MQTT topic		String
<code>QoS</code>	Quality of Service	⊙1	Long (I32)
	1 ..... QoS0 (Maximálně jednou)		
	2 ..... QoS1 (Alespoň jednou)		
	3 ..... QoS2 (Právě jednou)		
<code>RETAIN</code>	Zachovat poslední zprávu	⊙on	Bool
<code>defBuffSize</code>	Výchozí velikost bufferu	↓1 ⊙2048	Long (I32)

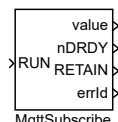
## Výstup

<code>BUSY</code>	Příznak probíhající operace	Bool
<code>DONE</code>	Příznak dokončení transakce	Bool
<code>errId</code>	Kód chyby	Error

## MqttSubscribe – Odběr zpráv z MQTT topic

Symbol bloku

Licence: [MQTT](#)



### Popis funkce

Tento funkční blok je závislý na MQTT ovladači. Je doporučeno si před použitím přečíst manuál MQTTDrv ovladače [12].

Účelem bloku MqttSubscribe je přihlášení se k odebírání zpráv ze zařízení typu *MQTT Broker* a jejich vyčítání prostřednictvím spojení, které zajišťuje ovladač MQTTDrv.

Parametr `topic` určuje tématický celek, k jehož odběru se blok přihlašuje. Protokol MQTT doručuje aplikační zprávy dle zvolené úrovně kvality služby (*Quality of Service* – *QoS*). Požadovaná úroveň lze nastavit parametrem `qoS`. Více informací naleznete ve specifikaci MQTT protokolu [13] (pouze anglicky).

Nastavením parametru `type` lze určit, jaký je očekávaný typ čtených zpráv. Blok se snaží konvertovat příchozí zprávy do zvoleného datového typu a výsledek nastavuje na výstupní signál `value` v případě úspěchu a nebo nastavuje výstup `errId` na příslušný chybový kód.

Parametr `mode` má dvě možné hodnoty: `Last value` a `Buffered values`. Pokud je parametr nastaven na hodnotu `Last value`, blok na výstup vždy vystaví pouze poslední zprávu i v případě, že bylo v poslední periodě přijato více zpráv. Pokud je ale parametr nastaven na hodnotu `Buffered values`, blok si zprávy ukládá do vnitřního bufferu a na výstup předává jednu zprávu za druhou v následujících ticích tasku.

Parametr `defBuffSize` může být použit pro optimalizaci práce s pamětí bloku. Hodnota parametru představuje velikost staticky alokované paměti pro vnitřní buffer odchozích zpráv. Pokud je velikost bufferu nadbytečně velká, blok si alokuje paměť, která je zbytečně blokována. Na druhou stranu pokud je hodnota parametru příliš malá, algoritmus bloku musí často dynamicky alokovat paměť, což je časově náročná operace.

Akce přihlášení se k odběru je provedena na základě vzestupné hrany (`off`→`on`) a odhlášení odběru na základě sestupné hrany (`on`→`off`) vstupu `RUN`.

Výstup `nDRDY` určuje počet přijatých zpráv, které jsou dostupné ve vnitřním bufferu. Pokud je parametr `mode` nastaven na hodnotu `Last value` výstup může mít pouze hodnotu 0 nebo 1.

Výstup `RETAIN` je nastaven na `on`, pokud přijatý paket měl nastaven příznak `RETAIN`. Více informací naleznete ve specifikaci MQTT protokolu [13] (pouze anglicky)..



Přihlašování se k odběrům s využitím zástupných znaků typu *wildcards* není podporováno.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>RUN</code>	Povolení běhu algoritmu	<code>Bool</code>
------------------	-------------------------	-------------------

## Parametr

<code>topic</code>	MQTT topic	<code>String</code>
<code>QoS</code>	Quality of Service	⊙1 <code>Long (I32)</code>
	1 ..... QoS0 (Maximálně jednou)	
	2 ..... QoS1 (Alespoň jednou)	
	3 ..... QoS2 (Právě jednou)	
<code>type</code>	Očekávaný typ příchozích dat	⊙1 <code>Long (I32)</code>
	1 ..... string	
	2 ..... double	
	3 ..... long	
	4 ..... bool	
	5 ..... byte vector/blob	
<code>mode</code>	Způsob bufferování příchozích zpráv	⊙1 <code>Long (I32)</code>
	1 ..... poslední hodnota	
	2 ..... buffrovat hodnoty	
<code>defBuffSize</code>	Výchozí velikost bufferu	↓1 ⊙2048 <code>Long (I32)</code>

## Výstup

<code>value</code>	Výstupní signál	<code>Any</code>
<code>nRDY</code>	Počet přijatých zpráv	↓0 ↑10 <code>Long (I32)</code>
<code>RETAIN</code>	Zachovat poslední zprávu	⊙on <code>Bool</code>
<code>errId</code>	Kód chyby	<code>Error</code>



## Kapitola 20

# MC\_SINGLE – Řízení pohybu v jedné ose

### Obsah

---

MCP_AccelerationProfile – * Generování trajektorie (zrychlení) .	542
MCP_Halt – * Zastavení pohybu (přerušitelné) . . . . .	544
MCP_HaltSuperimposed – * Zastavení pohybu (přídavné a přerušitelné) . . . . .	545
MCP_Home – * Nalezení výchozí polohy . . . . .	546
MCP_MoveAbsolute – * Pohyb do pozice (absolutní souřadnice) . .	548
MCP_MoveAdditive – * Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	550
MCP_MoveContinuousAbsolute – * Pohyb do pozice (absolutní souřadnice) . . . . .	552
MCP_MoveContinuousRelative – * Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	554
MCP_MoveRelative – * Kartézské transformace . . . . .	556
MCP_MoveSuperimposed – * Pohyb do pozice (přídavný pohyb) . .	558
MCP_MoveVelocity – * Pohyb konstantní rychlostí . . . . .	559
MCP_PositionProfile – * Generování trajektorie (poloha) . . . . .	561
MCP_SetOverride – * Nastavení násobivých faktorů na ose . . . . .	563
MCP_Stop – * Zastavení pohybu . . . . .	564
MCP_TorqueControl – * Řízení síly/momentu . . . . .	565
MCP_VelocityProfile – * Generování trajektorie (rychlost) . . . . .	567
MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení) . . . . .	569
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné) . . . . .	573
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné) . . . . .	575

MC_Home, MCP_Home – Nalezení výchozí polohy . . . . .	576
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	578
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	581
MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	584
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	588
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění) . . . . .	592
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb) . . . . .	595
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí	598
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha) . . . . .	601
MC_Power – Aktivace osy . . . . .	605
MC_ReadActualPosition – Skutečná poloha osy . . . . .	606
MC_ReadAxisError – Chyba osy . . . . .	607
MC_ReadBoolParameter – Čtení parametru (logická hodnota) . . .	608
MC_ReadParameter – Čtení parametru (číselná hodnota) . . . . .	609
MC_ReadStatus – Stav osy . . . . .	611
MC_Reset – Nulování chyb osy . . . . .	613
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose . . . . .	614
MC_Stop, MCP_Stop – Zastavení pohybu . . . . .	616
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu . . .	618
MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost) . . . . .	621
MC_WriteBoolParameter – Nastavení parametru (logická hodnota)	625
MC_WriteParameter – Nastavení parametru (číselná hodnota) . . .	626
RM_Axis – Osa pro řízení pohybu . . . . .	627
RM_AxisOut – Výstupní blok osy . . . . .	634
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení) . . . . .	635
RM_HomeOffset – * Nastavení počáteční polohy pomocí offsetu . .	640
RM_Track – Sledování a krokování . . . . .	641

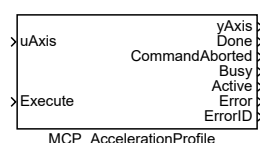
Knihovna MC\_SINGLE je určena pro řízení pohybu v jednoosých systémech. Obsahuje bloky jako MC\_MoveAbsolute, MC\_MoveRelative a MC\_MoveVelocity pro přesné polohování a ovládání rychlosti. Knihovna obsahuje MC\_Home pro operace navádění a MC\_Power pro řízení stavu napájení osy. Pokročilé funkce poskytují MC\_AccelerationProfile,

`MC_PositionProfile` a `MC_VelocityProfile` pro přizpůsobení profilů pohybu. Nabízí také možnosti monitorování a úpravy parametrů prostřednictvím `MC_ReadActualPosition`, `MC_ReadAxisError`, `MC_ReadParameter` a `MC_WriteParameter`. Knihovna navíc obsahuje bloky jako `MC_Halt`, `MC_Reset` a `MC_Stop` pro nouzové a kontrolní operace. Tato knihovna je nezbytná pro aplikace vyžadující přesný a řízený pohyb v jednoosých konfiguracích.

## MCP\_AccelerationProfile – \* Generování trajektorie (zrychlení)

Symbol bloku

Licence: MOTION CONTROL



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

### Parametr

<b>alg</b>	Typ interpolace	⊙1	<b>Long (I32)</b>
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
	5 ..... tabulka čas/hodnota (+okraj)		
	6 ..... hodnoty ve stejném intervalu (+okraj)		
	7 ..... aproximace polynomem 3.řádu (B-spline)		
	8 ..... aproximace polynomem 5.řádu (B-spline)		
	9 ..... vše lineární		
<b>nmax</b>	Počet segmentů profilu	⊙3	<b>Long (I32)</b>
<b>TimeScale</b>	Konstanta násobení pro přepočet časové osy profilu	⊙1.0	<b>Double (F64)</b>
<b>AccelerationScale</b>	Konstanta násobení pro přepočet hodnotové osy profilu	⊙1.0	<b>Double (F64)</b>
<b>Offset</b>	Aditivní konstanta pro přepočet hodnotové osy profilu		<b>Double (F64)</b>

<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 . . . . . aborting		
	2 . . . . . buffered		
	3 . . . . . blending low		
	4 . . . . . blending high		
	5 . . . . . blending previous		
	6 . . . . . blending next		
<b>times</b>	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 30]	Double (F64)
<b>values</b>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 50]	Double (F64)

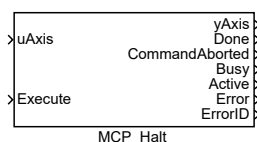
## Výstup

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i . . . . . kód chyby REXYGENu	

## MCP\_Halt – \* Zastavení pohybu (přerušitelné)

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

## Parametr

<code>Deceleration</code>	Maximální povolené zpomalení [ $\text{unit}/\text{s}^2$ ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [ $\text{unit}/\text{s}^3$ ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	

## Výstup

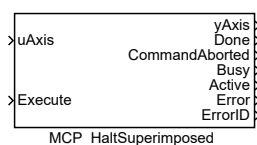
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	



## MCP\_HaltSuperimposed – \* Zastavení pohybu (přídavné a přerušitelné)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

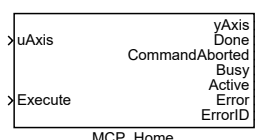
<code>Deceleration</code>	Maximální povolené zpomalení [ $\text{unit}/\text{s}^2$ ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [ $\text{unit}/\text{s}^3$ ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	

### Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

MCP\_Home – \* **Nalezení výchozí polohy**

Symbol bloku

Licence: [MOTION CONTROL](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

## Parametr

<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>TorqueLimit</code>	Maximální povolený moment/síla	Double (F64)
<code>TimeLimit</code>	Maximální povolený čas pro celý algoritmus bloku [s]	Double (F64)
<code>DistanceLimit</code>	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	Double (F64)
<code>LagLimit</code>	Maximální povolený skluz pro celý algoritmus bloku [unit]	Double (F64)
<code>Direction</code>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙3 Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<code>Position_</code>	Požadovaná poloha [unit]	Double (F64)
<code>HomingMode</code>	Algoritmus hledání výchozí pozice	⊙1 Long (I32)
	1 ..... nulový spínač	
	2 ..... koncové spínače	
	3 ..... nulový pulz	
	4 ..... uživatelská hodnota	
	5 ..... absolutní snímač polohy	
	6 ..... mechanický doraz	
	7 ..... reserva1	
	8 ..... reserva2	

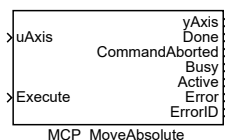
## Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . kód chyby <code>REXYGENu</code>	Error

## MCP\_MoveAbsolute – \* Pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

## Parametr

<b>Position_</b>	Požadovaná poloha [unit]	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0 Double (F64)
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1 Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

## Výstup

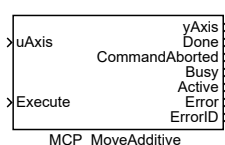
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
--------------	--	-----------

<code>Done</code>	Příznak dokončení algoritmu	<code>Bool</code>
<code>CommandAborted</code>	Příznak přerušení funkce bloku	<code>Bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>Bool</code>
<code>Active</code>	Příznak, že blok řídí osu	<code>Bool</code>
<code>Error</code>	Příznak chyby	<code>Bool</code>
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby REXYGENu	<code>Error</code>

## MCP\_MoveAdditive – \* Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

<b>Distance</b>	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0 Double (F64)
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	

### Výstup

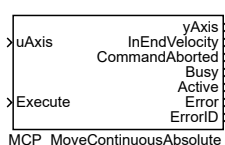
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool

<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_MoveContinuousAbsolute – \* Pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

### Parametr

<b>Position_</b>	Požadovaná poloha [unit]		<b>Double (F64)</b>
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	↓0.0	<b>Double (F64)</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0	<b>Double (F64)</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0	<b>Double (F64)</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0	<b>Double (F64)</b>
<b>BufferMode</b>	Režim převzetí osy	⊙1	<b>Long (I32)</b>
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1	<b>Long (I32)</b>
	1 ..... kladný		
	2 ..... nejkratší		
	3 ..... záporný		
	4 ..... aktuální		
<b>EndVelocity</b>	Koncová rychlost		<b>Double (F64)</b>

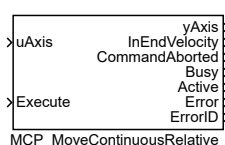


## Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>InEndVelocity</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . kód chyby <code>REXYGENu</code>	Error

## MCP\_MoveContinuousRelative – \* Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbol bloku

Licence: [MOTION CONTROL](#)

### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

<b>Distance</b>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0 Double (F64)
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<b>EndVelocity</b>	Koncová rychlost	Double (F64)

### Výstup

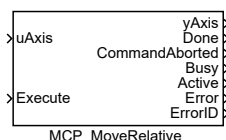
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>InEndVelocity</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool

<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

MCP\_MoveRelative – \* **Kartézské transformace**

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

## Parametr

Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0 Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	

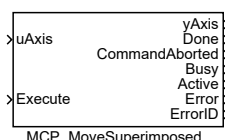
## Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool

<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_MoveSuperimposed – \* Pohyb do pozice (přídavný pohyb)

Symbol bloku

Licence: [MOTION CONTROL](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

## Parametr

<b>Distance</b>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	<b>Double (F64)</b>
<b>VelocityDiff</b>	Maximální povolená rychlost [unit/s]	↓0.0 <b>Double (F64)</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	↓0.0 <b>Double (F64)</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	↓0.0 <b>Double (F64)</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	↓0.0 <b>Double (F64)</b>

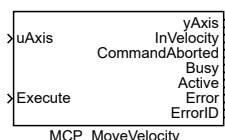
## Výstup

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>Bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_MoveVelocity – \* Pohyb konstantní rychlostí

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1 Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	

### Výstup

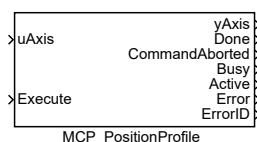
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
--------------	--	-----------

InVelocity	Příznak dosažení požadované rychlosti	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... kód chyby REXYGENu	Error



MCP\_PositionProfile – \* **Generování trajektorie (poloha)**

Symbol bloku

Licence: [MOTION CONTROL](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

## Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

## Parametr

<b>alg</b>	Typ interpolace	⊙2	Long (I32)
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
	5 ..... tabulka čas/hodnota (+okraj)		
	6 ..... hodnoty ve stejném intervalu (+okraj)		
	7 ..... aproximace polynomem 3.řádu (B-spline)		
	8 ..... aproximace polynomem 5.řádu (B-spline)		
	9 ..... vše lineární		
<b>nmax</b>	Počet segmentů profilu	⊙3	Long (I32)
<b>TimeScale</b>	Konstanta násobení pro přepočtení časové osy profilu	⊙1.0	Double (F64)
<b>PositionScale</b>	Konstanta násobení pro přepočtení hodnotové osy profilu	⊙1.0	Double (F64)
<b>Offset</b>	Aditivní konstanta pro přepočtení hodnotové osy profilu		Double (F64)
<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		

<code>BeginVelocity</code>	rychlost na počátku pohybu	Double (F64)
<code>EndVelocity</code>	rychlost na konci pohybu	Double (F64)
<code>times</code>	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 30] Double (F64)
<code>values</code>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 50] Double (F64)

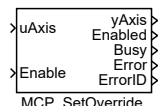
## Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby <code>REXYGENu</code>	Error

## MCP\_SetOverride – \* Nastavení násobivých faktorů na ose

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>Bool</b>

### Parametr

<b>diff</b>	Minimální hodnota změny násobivého faktoru	↓0.0 ↑1.0	<b>Double (F64)</b>
<b>VelFactor</b>	Faktor násobení pro rychlost	⊙1.0	<b>Double (F64)</b>
<b>AccFactor</b>	Faktor násobení pro zrychlení	⊙1.0	<b>Double (F64)</b>
<b>JerkFactor</b>	Faktor násobení pro změnu zrychlení	⊙1.0	<b>Double (F64)</b>

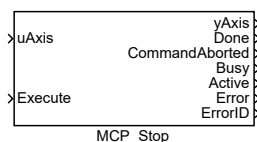
### Výstup

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>Enabled</b>	Povolení funkce bloku (aktivace výstupů)	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_Stop – \* Zastavení pohybu

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

## Parametr

<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

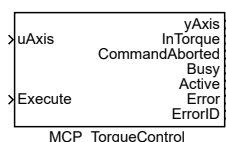
## Výstup

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	Error

## MCP\_TorqueControl – \* Řízení síly/momentu

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

<b>kma</b>	Poměr mezi silou/momentem a zrychlením	Double (F64)
<b>Torque</b>	Maximální povolený moment/síla	Double (F64)
<b>TorqueRamp</b>	Maximální povolená změna momentu/síly	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1 Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	

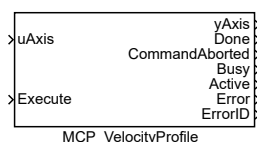
## Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>InTorque</code>	Příznak dosažení požadovaného momentu/síly	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby REXYGEnu	Error

## MCP\_VelocityProfile – \* Generování trajektorie (rychlost)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

<b>alg</b>	Typ interpolace	⊙1	Long (I32)
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
	5 ..... tabulka čas/hodnota (+okraj)		
	6 ..... hodnoty ve stejném intervalu (+okraj)		
	7 ..... aproximace polynomem 3.řádu (B-spline)		
	8 ..... aproximace polynomem 5.řádu (B-spline)		
	9 ..... vše lineární		
<b>nmax</b>	Počet segmentů profilu	⊙3	Long (I32)
<b>TimeScale</b>	Konstanta násobení pro přepočtení časové osy profilu	⊙1.0	Double (F64)
<b>VelocityScale</b>	Konstanta násobení pro přepočtení hodnotové osy profilu	⊙1.0	Double (F64)
<b>Offset</b>	Aditivní konstanta pro přepočtení hodnotové osy profilu		Double (F64)
<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		

<b>BeginAcceleration</b>	zrychlení na počátku pohybu	Double (F64)
<b>EndAcceleration</b>	zrychlení na konci pohybu	Double (F64)
<b>times</b>	Posloupnost hraničních časů jednotlivých segmentů ⊙[0 15 25 30]	Double (F64)
<b>values</b>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...) ⊙[0 100 100 50]	Double (F64)

## Výstup

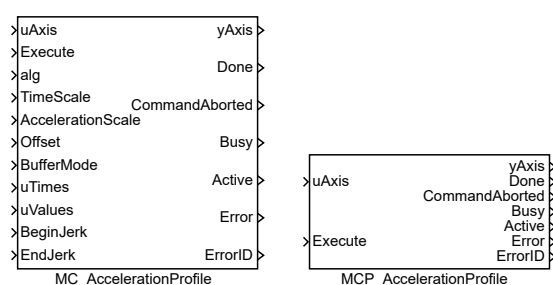
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGEnu	Error



## MC\_AccelerationProfile, MCP\_AccelerationProfile – Generování trajektorie (zrychlení)

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_AccelerationProfile a MCP\_AccelerationProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

### Popis funkce

Blok `MC_AccelerationProfile` generuje takovou trajektorii, aby zrychlení byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a zrychlení. Mezi jednotlivými časy se hodnota zrychlení interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru `times`, příslušné hodnoty zrychlení do pole/parametru `values`. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli `times`. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli `values` ve vzestupném pořadí (tj. pole `values` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli `times` jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Vstup/Parametr `uValues` musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 2: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak změnit. `AccelerationScale` a `Offset` je samozřejmě pro zrychlení. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 3: Dialog je jednotný pro `MC_AccelerationProfile`, `MC_VelocityProfile`, `MC_PositionProfile`. Některé režimy (parametr `alg` nedávají pro `AccelerationProfile` smysl a nefungují (aproximace B-spline nefunguje, režimy 1,2,5,6 používají lineární interpolaci)

Poznámka 4: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací `PLCopen`).

## Vstupy

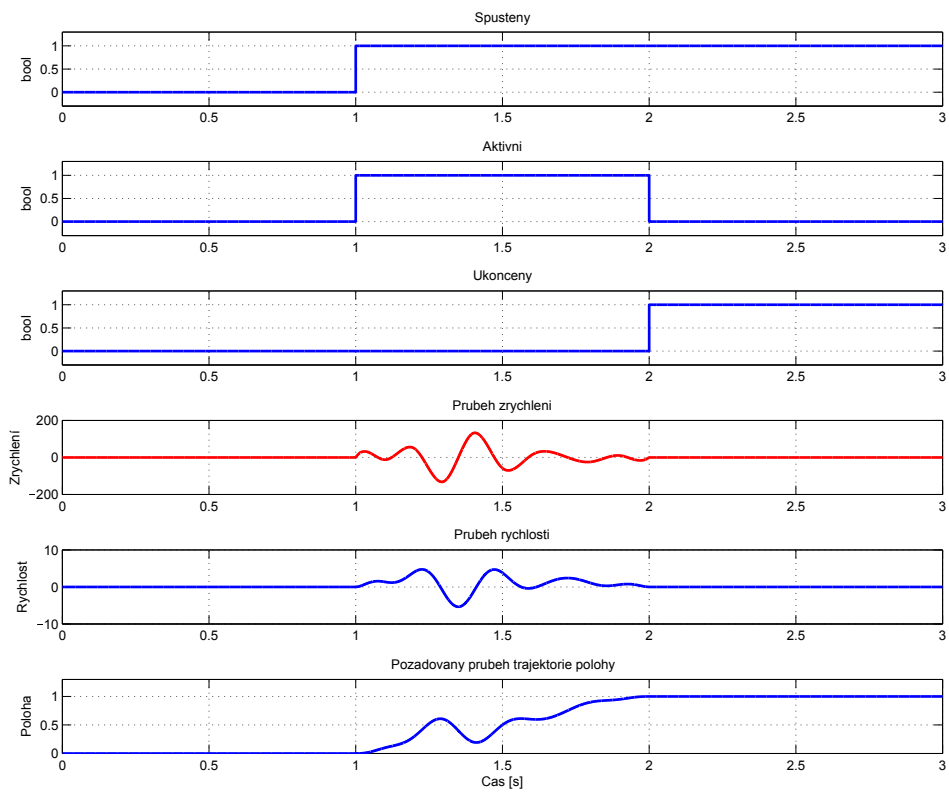
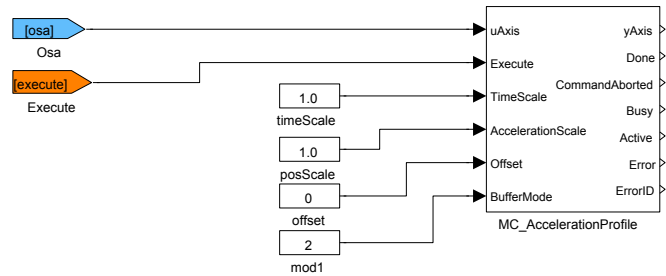
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>alg</code>	Typ interpolace	⊙2 Long (I32)
	1 ..... tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 ..... interpolace polynomy 5. řádu	
	4 ..... polynomy 5. řádu s ekvidistantními intervaly	
	5 ..... tabulka čas/hodnota + počáteční a koncová derivace	
	6 ..... hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 ..... hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 ..... hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 ..... tabulka čas/hodnota (lineární interpolace)	
<code>nmax</code>	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
<code>TimeScale</code>	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
<code>AccelerationScale</code>	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
<code>Offset</code>	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
<code>uTimes</code>	vektor s časovými hodnotami	Reference

<b>uValues</b>	vektor s hodnotami zrychlení nebo koeficienty polynomů	<b>Reference</b>
1 .....	Aborting (nový blok se spustí okamžitě)	
2 .....	Buffered (nový blok se spustí po dokončení předchozího)	
3 .....	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4 .....	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5 .....	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6 .....	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>Reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>Bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace	<b>Error</b>
i .....	obecná chyba systému REXYGEN	

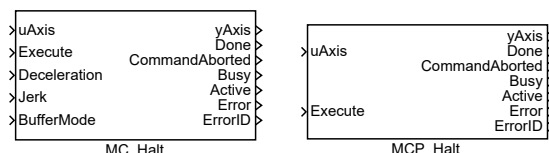
## Příklad



## MC\_Halt, MCP\_Halt – Zastavení pohybu (přerušitelné)

### Symboly bloků

Licence: MOTION CONTROL



### Popis funkce

*Bloky MC\_Halt a MCP\_Halt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_Halt** zahajuje řízené zastavení pohybu. Osa se přesune do stavu **DiscreteMotion**, dokud není rychlost nulová. Společně s nastavením výstupu "Done" je stav změněn na "Standstill".

Poznámka 1: Blok **MC\_Halt** se používá k zastavení osy za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší **MC\_Halt** a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. **MC\_Halt** je aktivován. Před dosažením stavu "Standstill" je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

### Výstupy

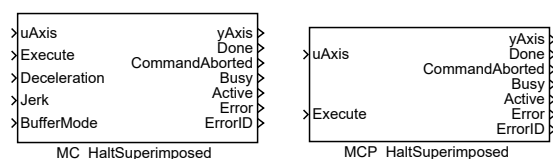
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool

<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace	<b>Error</b>
	i . . . . . obecná chyba systému REXYGEN	

## MC\_HaltSuperimposed, MCP\_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)

Symbole bloků

Licence: [MOTION CONTROL](#)



Popis funkce

*Bloky MC\_HaltSuperimposed a MCP\_HaltSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_HaltSuperimposed zahajuje řízené zastavení přídavného pohybu, který generuje například blok [MC\\_MoveSuperimposed](#).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

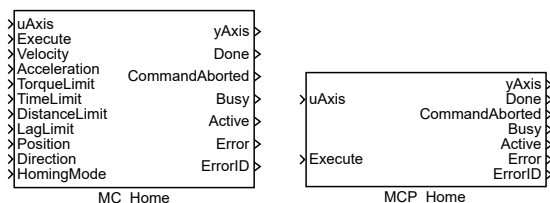
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## MC\_Home, MCP\_Home – Nalezení výchozí polohy

## Symboly bloků

Licence: MOTION CONTROL



## Popis funkce

*Bloky MC\_Home a MCP\_Home mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_Home provede algoritmus hledání výchozího bodu. Detaily závisí na parametrech bloku (zejména HomingMode). Podrobnější popis lze nalézt v PLCopen . Vstup Position je poloha, která je nastavena v okamžiku dosažení výchozí polohy. Po úspěšném ukončení algoritmu tohoto bloku je osa ve stavu „StandStill“.

Poznámka 1: Parametr či vstup BufferMode není podpořen - vždy je BufferMode = Aborting. To je nevýznamné omezení, protože tento blok se vykonává vždy jako první.

Poznámka 2: Tento blok vyžaduje připojené vstupy bloku RM\_Axis. Podle zvolené metody může být potřeba ActualPos, ActualTorque, LimP, LimZ, LimN . Předpokládá se, že metoda je zvolena dopředu podle konstrukce stroje, proto není zvlášť vstup na nulový koncový spínač a referenční pulz - jeden z nich se připojí k LimZ a zvolí se HomingMode=3(nulový pulz).

Poznámka 3: HomingMode=4(uživatelská hodnota) pouze nastaví aktuální polohu. Proto také není implementován blok MC\_SetPosition. HomingMode=5 (absolutní snímač polohy) pouze přepne osu do stavu StandStill.

Poznámka 4: Tento blok nepodporuje jerk (limit na derivaci zrychlení) a také není použit zvlášť limit na zrychlení a brzdění. Pokud je potřeba nastavit výchozí polohu velmi přesně, doporučuje se spustit blok MC\_Home dvakrát. Poprvé s velkou rychlostí pro najetí blízko výchozí polohy a podruhé s malou rychlostí pro přesné nastavení pozice.

Poznámka 5: Pro HomingMode=6 (najetí na mechanickou překážku) se požadovaná poloha pozná podle toho, že moment překročil nastavenou mez (tj. v tomto režimu to není chyba procedury) nebo že "position lag" překročil nastavenou mez (druhá podmínka se kontroluje jen pokud parametr MaxPositionLag v příslušném bloku RM\_Axis je kladný).



## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>TorqueLimit</code>	Maximální povolený moment/síla	Double (F64)
<code>TimeLimit</code>	Maximální povolený čas pro celý algoritmus bloku [s]	Double (F64)
<code>DistanceLimit</code>	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	Double (F64)
<code>Position</code>	Požadovaná poloha [unit]	Double (F64)
<code>Direction</code>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<code>HomingMode</code>	Algoritmus hledání výchozí pozice	Long (I32)
	1 ..... nulový spínač	
	2 ..... koncové spínače	
	3 ..... nulový pulz	
	4 ..... uživatelská hodnota	
	5 ..... absolutní snímač polohy	
	6 ..... mechanický doraz	

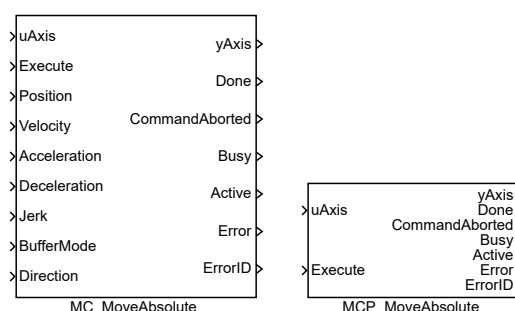
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## MC\_MoveAbsolute, MCP\_MoveAbsolute – Pohyb do pozice (absolutní souřadnice)

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveAbsolute** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně

z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

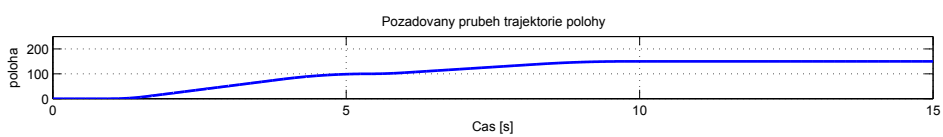
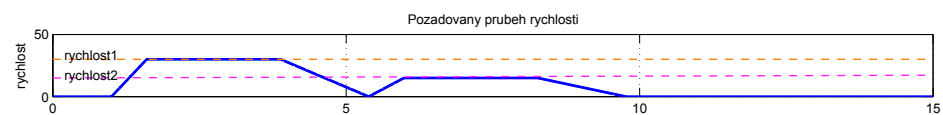
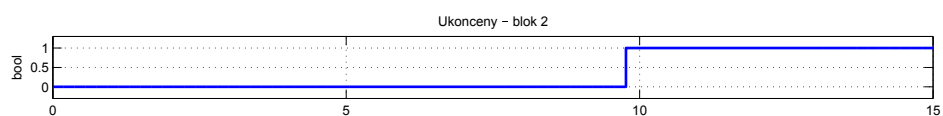
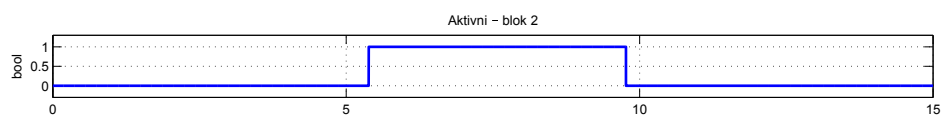
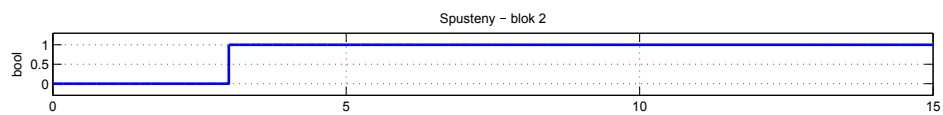
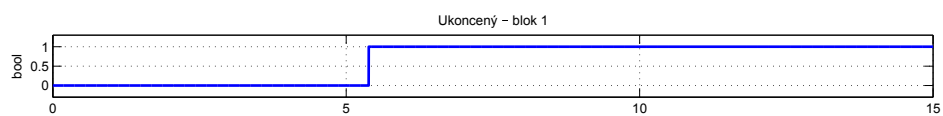
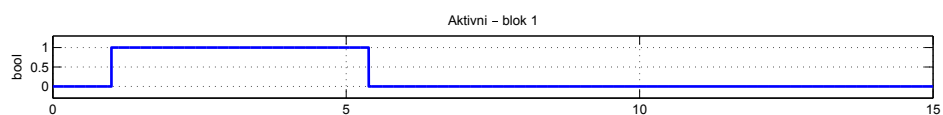
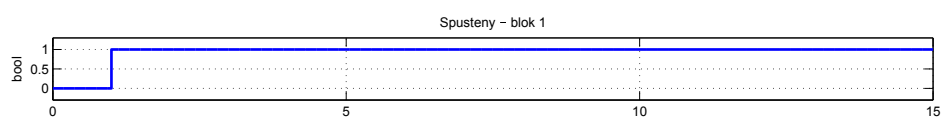
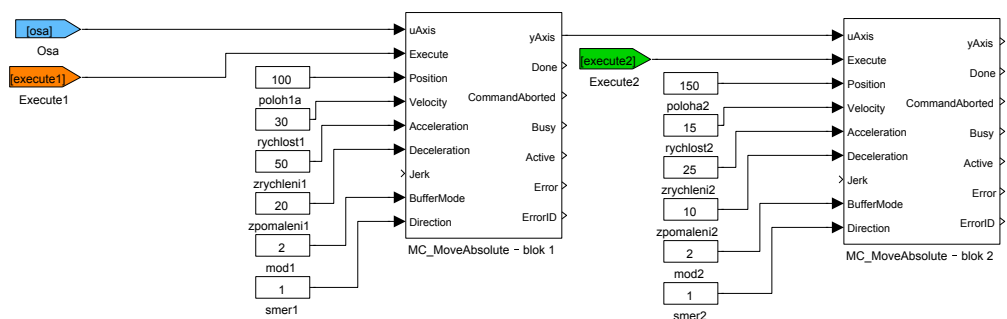
## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Position</b>	Požadovaná poloha [unit]	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

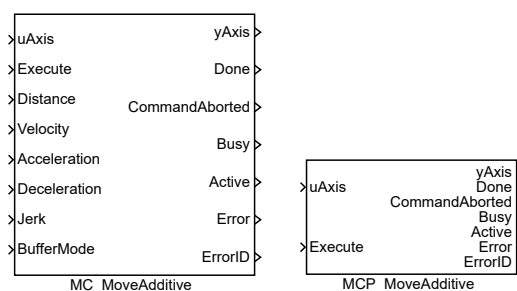
## Příklad



## MC\_MoveAdditive, MCP\_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbyly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveAdditive` přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku převzetí řízení osy přičte hodnota parametru `Distance`. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru `BufferedMode` následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr `Velocity`). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mód buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně

z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

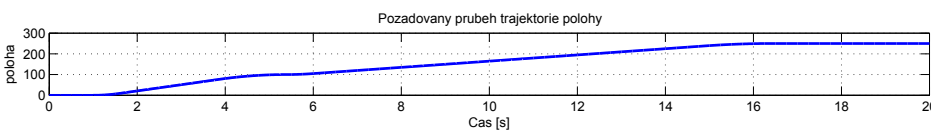
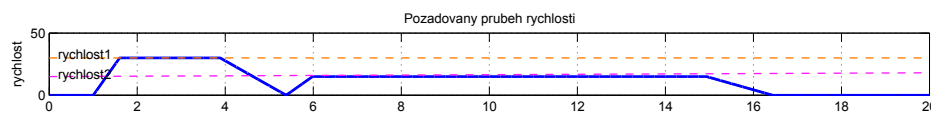
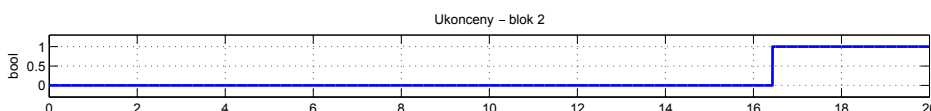
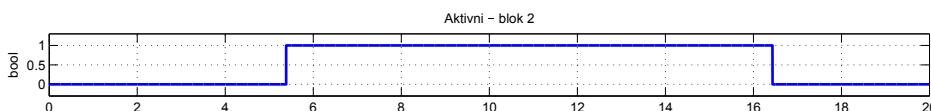
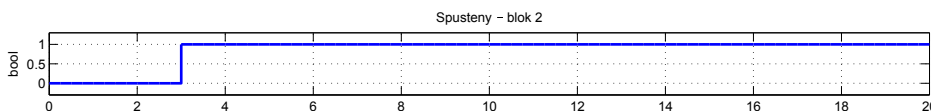
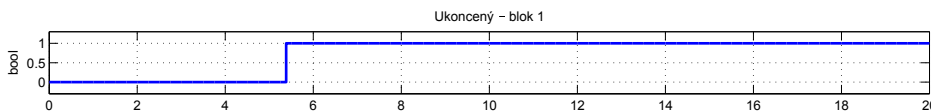
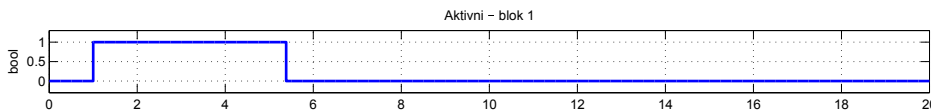
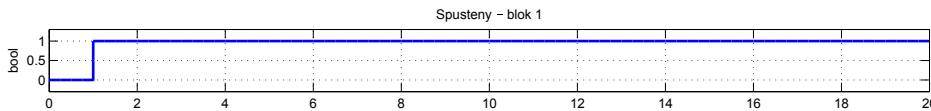
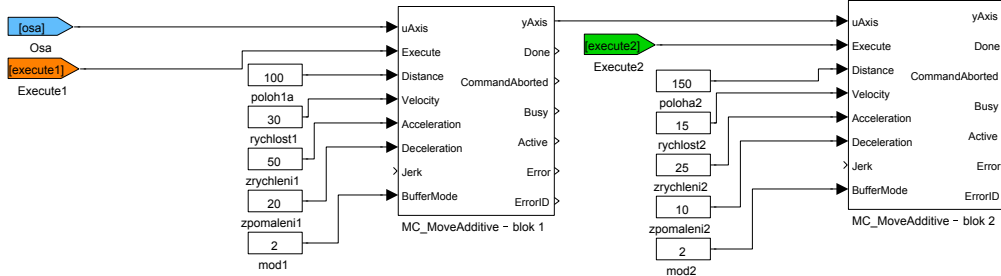
## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Distance</code>	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

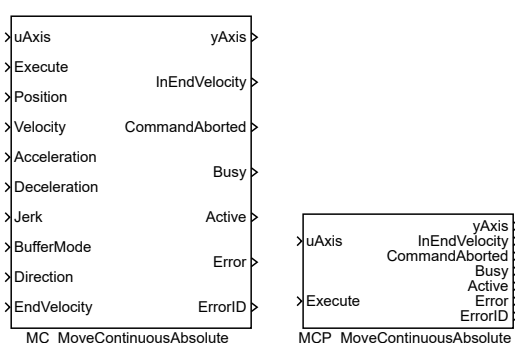
### Příklad



## MC\_MoveContinuousAbsolute, MCP\_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveContinuousAbsolute` přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru `EndVelocity`, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru `BufferedMode` následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr `Velocity`). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti.

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.



Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr `EndVelocity` na nulovou hodnotu, pak se chová blok stejným způsobem jako `MC_MoveAbsolute`.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako `MC_MoveAbsolute`.

## Vstupy

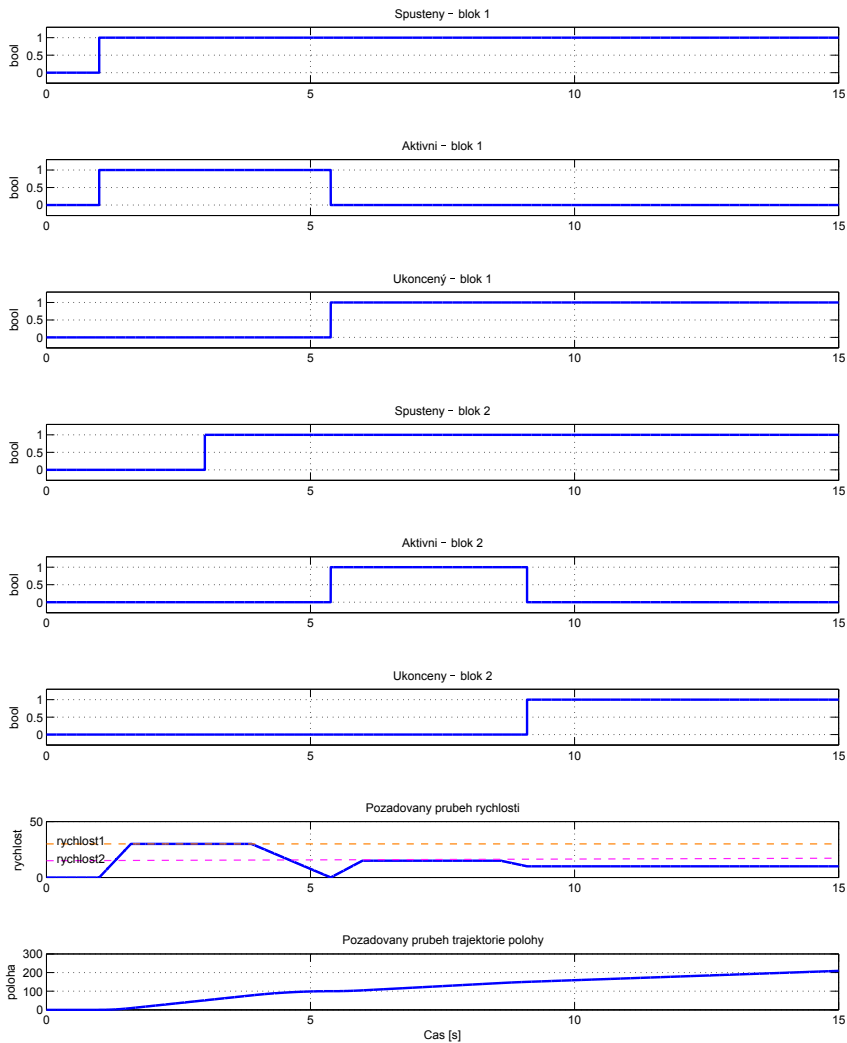
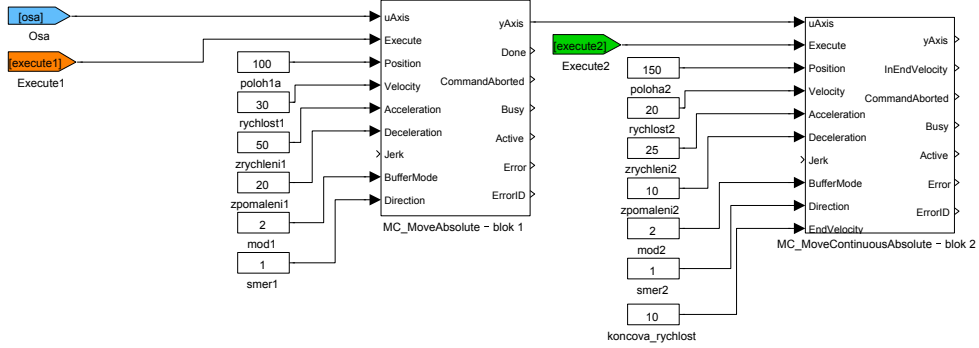
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Position</code>	Požadovaná poloha [unit]	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<code>Direction</code>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<code>EndVelocity</code>	Koncová rychlost	Double (F64)

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>InEndVelocity</code>	Příznak dokončení algoritmu	Bool

<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	<b>Error</b>

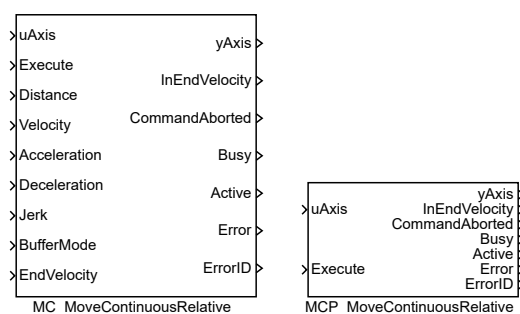
### Příklad



## MC\_MoveContinuousRelative, MCP\_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveContinuousRelative a MCP\_MoveContinuousRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveContinuousRelative** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru **EndVelocity**, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr `EndVelocity` na nulovou hodnotu, pak se chová blok stejným způsobem jako `MC_MoveRelative`.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako `MC_MoveRelative`.

## Vstupy

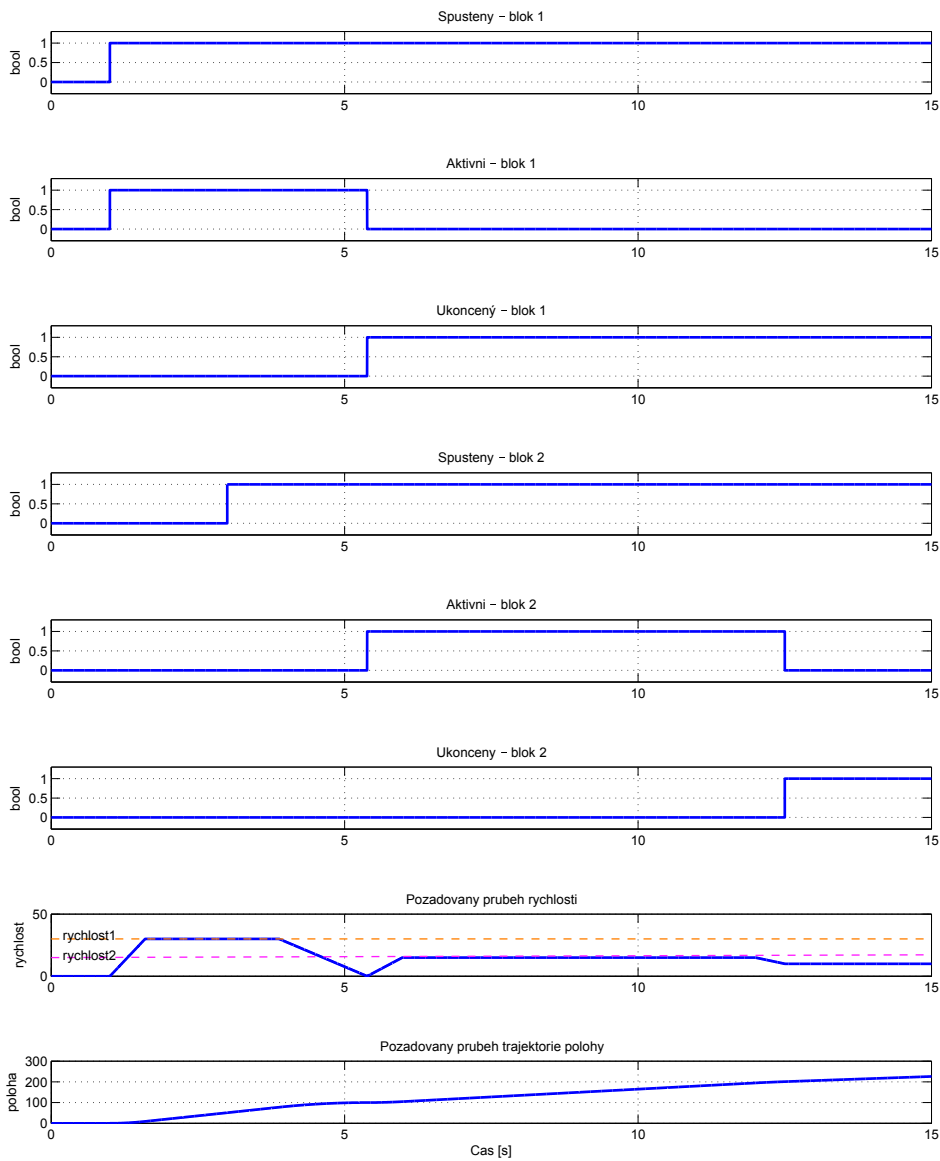
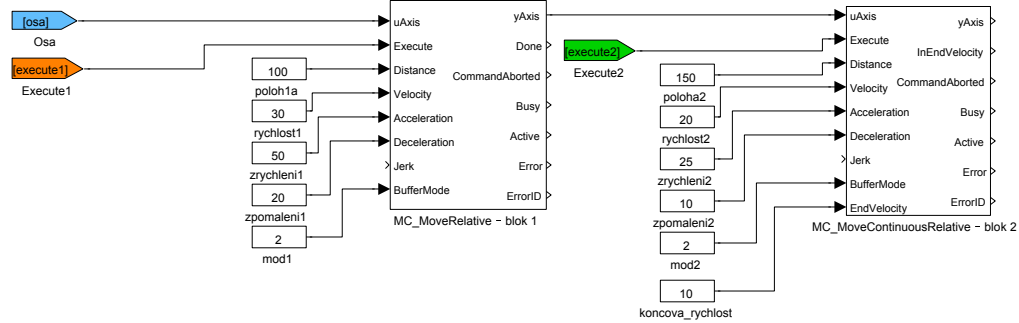
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Distance</code>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<code>EndVelocity</code>	Koncová rychlost	Long (I32)

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>InEndVelocity</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	..... obecná chyba systému REXYGEN	

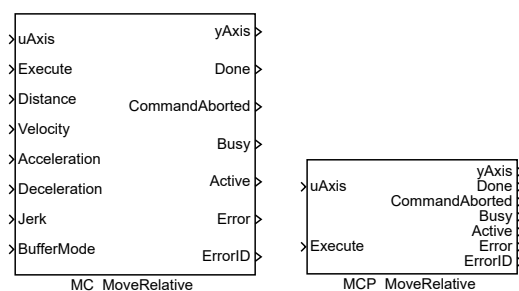
Příklad



## MC\_MoveRelative, MCP\_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění)

Symbole bloků

Licence: MOTION CONTROL



Popis funkce

*Bloky MC\_MoveRelative a MCP\_MoveRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveRelative** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.



Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

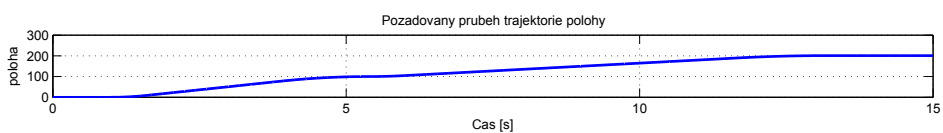
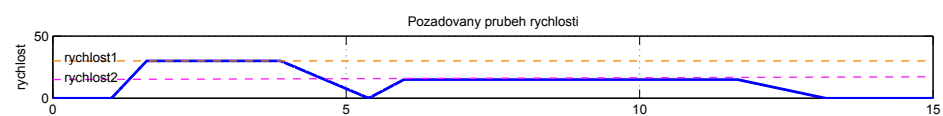
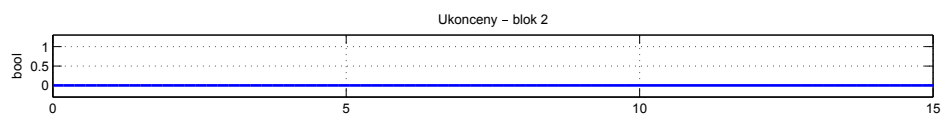
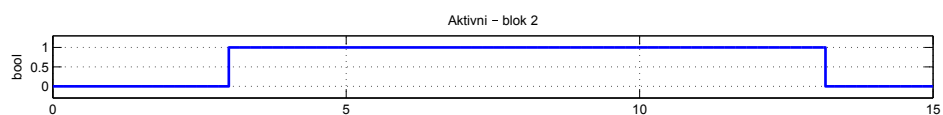
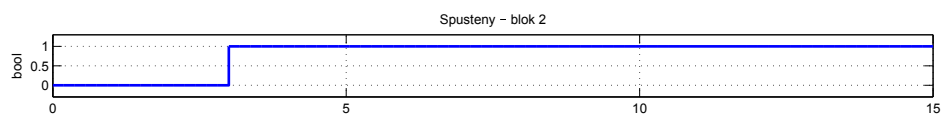
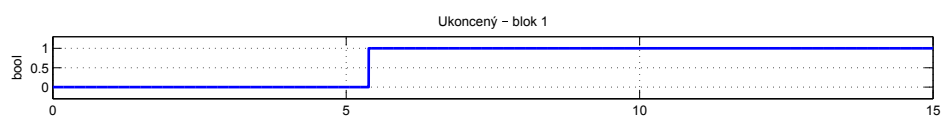
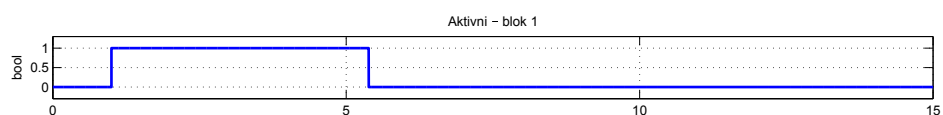
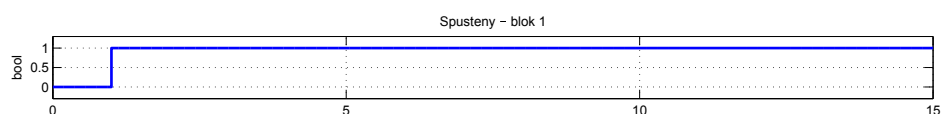
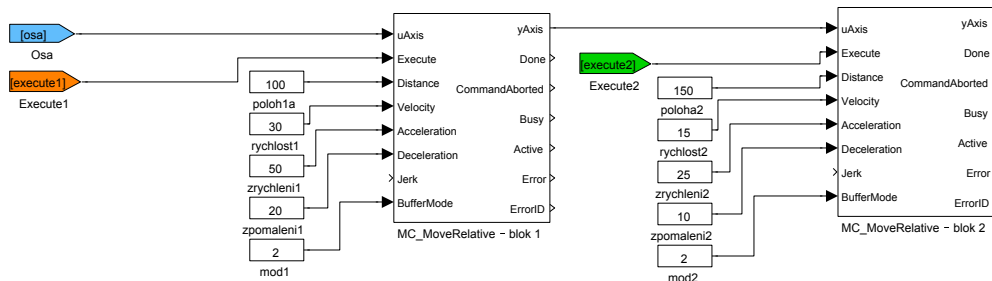
## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Distance</code>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

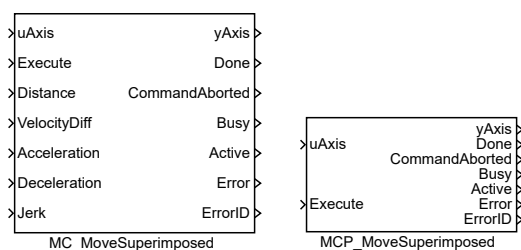
## Příklad



## MC\_MoveSuperimposed, MCP\_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)

### Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveSuperimposed a MCP\_MoveSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveSuperimposed** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha je hodnota parametru **Distance**, přičemž počáteční poloha se považuje za nulovou. Pokud již nějaký blok běží, původní blok běží dále a hodnoty (poloha, rychlost, zrychlení) superimposed bloku se přičítají k hodnotám generovaným původním blokem. Pokud žádný blok neběží, tento blok se chová stejně jako **MC\_MoveRelative**.

Poznámka: Tento blok nemá parametr **BufferMode**, protože v superimposed režimu je to irelevantní. Pokud v okamžiku spuštění (náběžná hrana na vstupu **Execute**) je již nějaký blok v režimu superimposed aktivní, počká se na doběhnutí a pak se spustí nový blok (tj. analogicky režimu buffered).

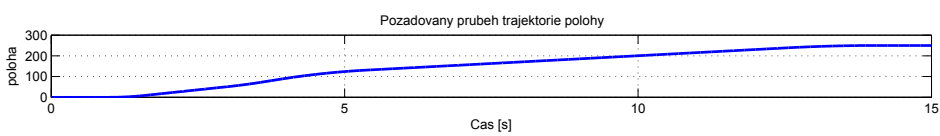
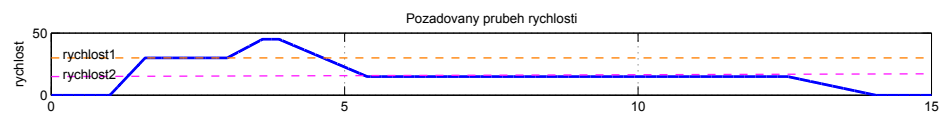
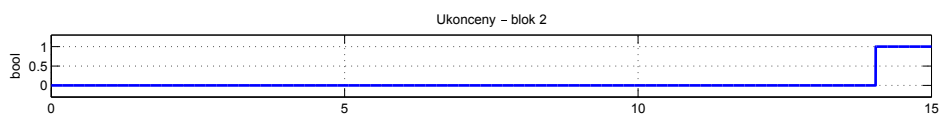
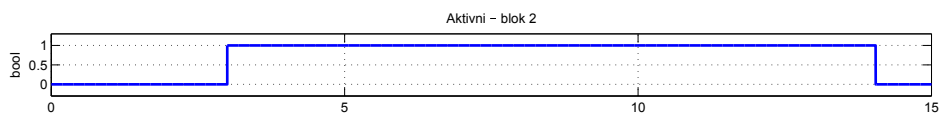
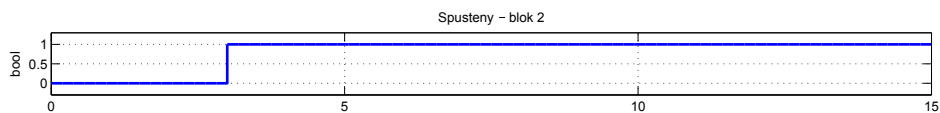
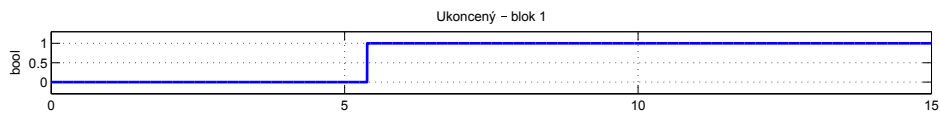
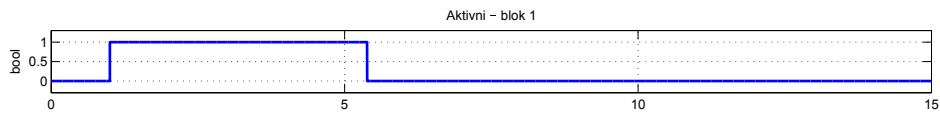
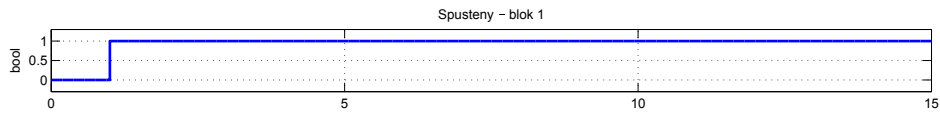
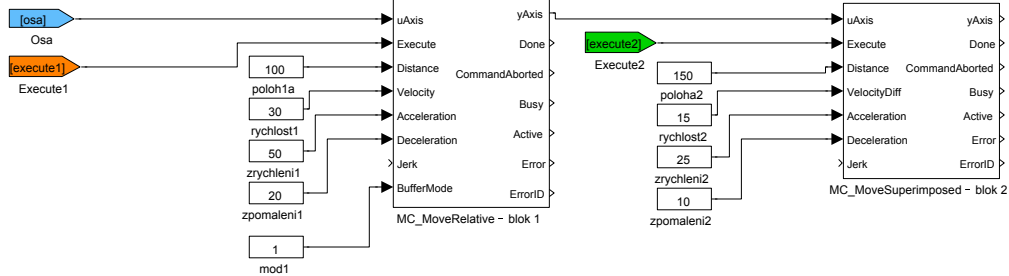
### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Distance</b>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
<b>VelocityDiff</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

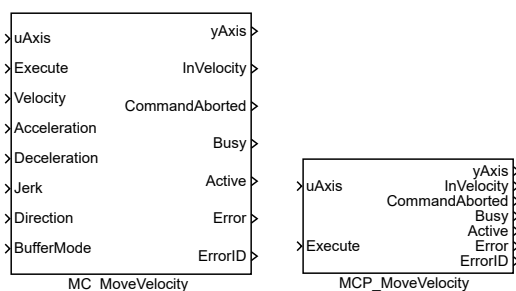
Příklad



## MC\_MoveVelocity, MCP\_MoveVelocity – Pohyb konstantní rychlostí

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

*Bloky MC\_MoveVelocity a MCP\_MoveVelocity mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_MoveVelocity změní rychlost osy na požadovanou hodnotu za nejkratší možný čas s respektováním omezení na zrychlení a popřípadě jerk. Rychlost pak zůstává konstantní, dokud není aktivován jiný blok.

Vstupy

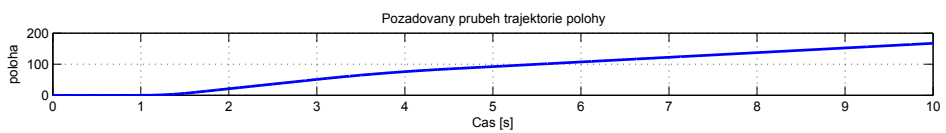
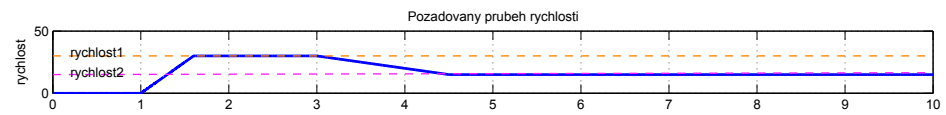
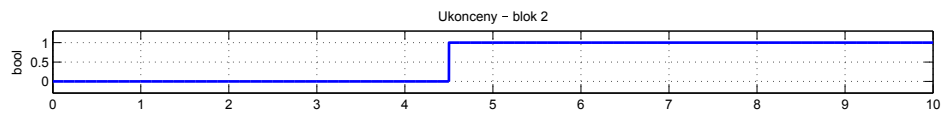
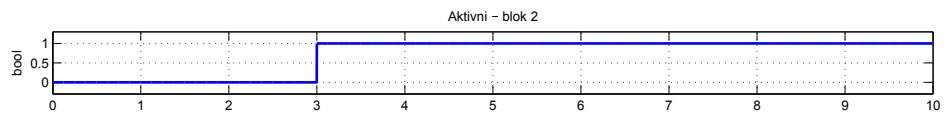
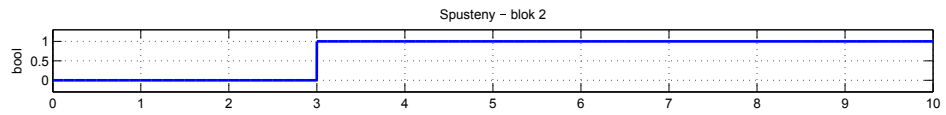
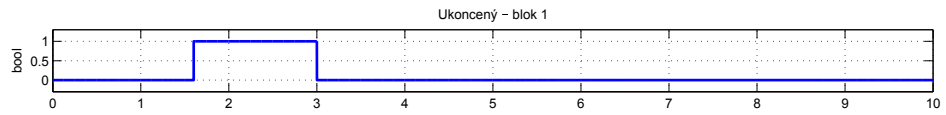
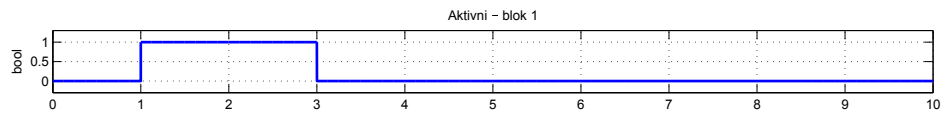
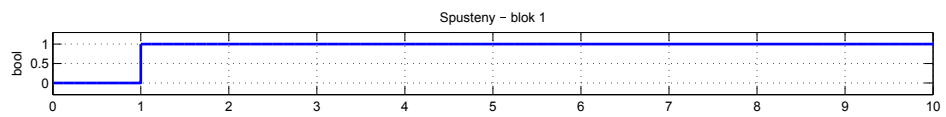
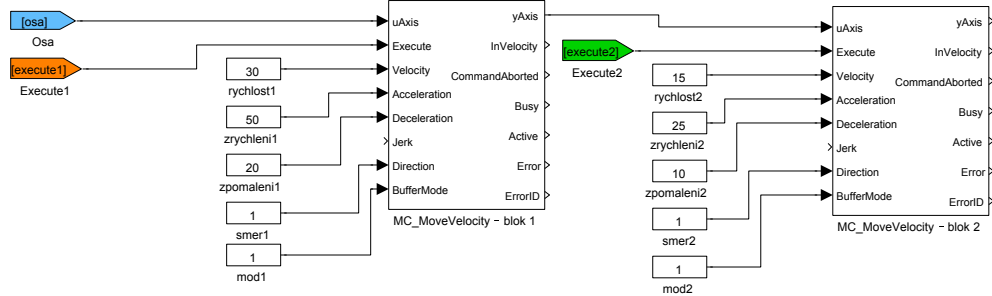
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<b>InVelocity</b>	Příznak dosažení požadované rychlosti	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

Příklad

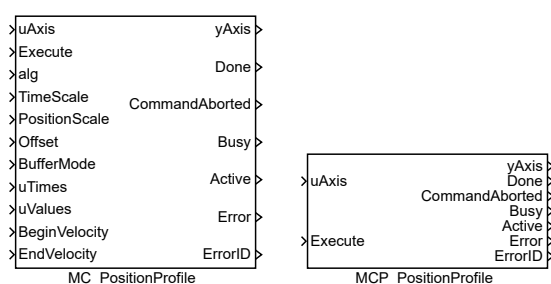




## MC\_PositionProfile, MCP\_PositionProfile – Generování trajektorie (poloha)

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_PositionProfile a MCP\_PositionProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_PositinProfile` generuje takovou trajektorii, aby poloha byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a poloha. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolace není v tomto případě vhodná, protože na okrajích intervalu by byl skok v rychlosti). Hodnoty času (v sekundách) se zadávají do pole/parametru `times`, příslušné hodnoty polohy do pole/parametru `values`. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli `times`. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli `values` ve vzestupném pořadí (tj. pole `values` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli `times` jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Parametr `values` musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 2: V režimu zadání funkce polynomem je pátého řádu a nelze to nijak

měnit. Polynomy na sebe musí hladce navazovat, jinak dochází ke skokům v rychlosti a/nebo poloze. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 3: Block neobsahuje tzv. ramp-in mode. Pokud tedy rychlost nebo poloha osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti a poloze profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému s rychlostí lze předejít, pokud se použije `BufferMode=BlendingNext`. Skok v poloze se musí řešit správně nastaveným parametrem `Offset`.

Poznámka 4: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací `PLCopen`).

Poznámka 5: pokud je parametr `alg=1` nebo `alg=5` nebo `alg=9` je možné nechat pole `times` prázdné (popř. nepřipojený vstup) a pak zadávat celou vačku v poli `values`, kde první sloupec jsou časové okamžiky, druhý sloupec polohy, třetí (nepovinný) sloupec rchlosti, čtvrtý (nepovinný) sloupec zrychlení.

## Vstupy

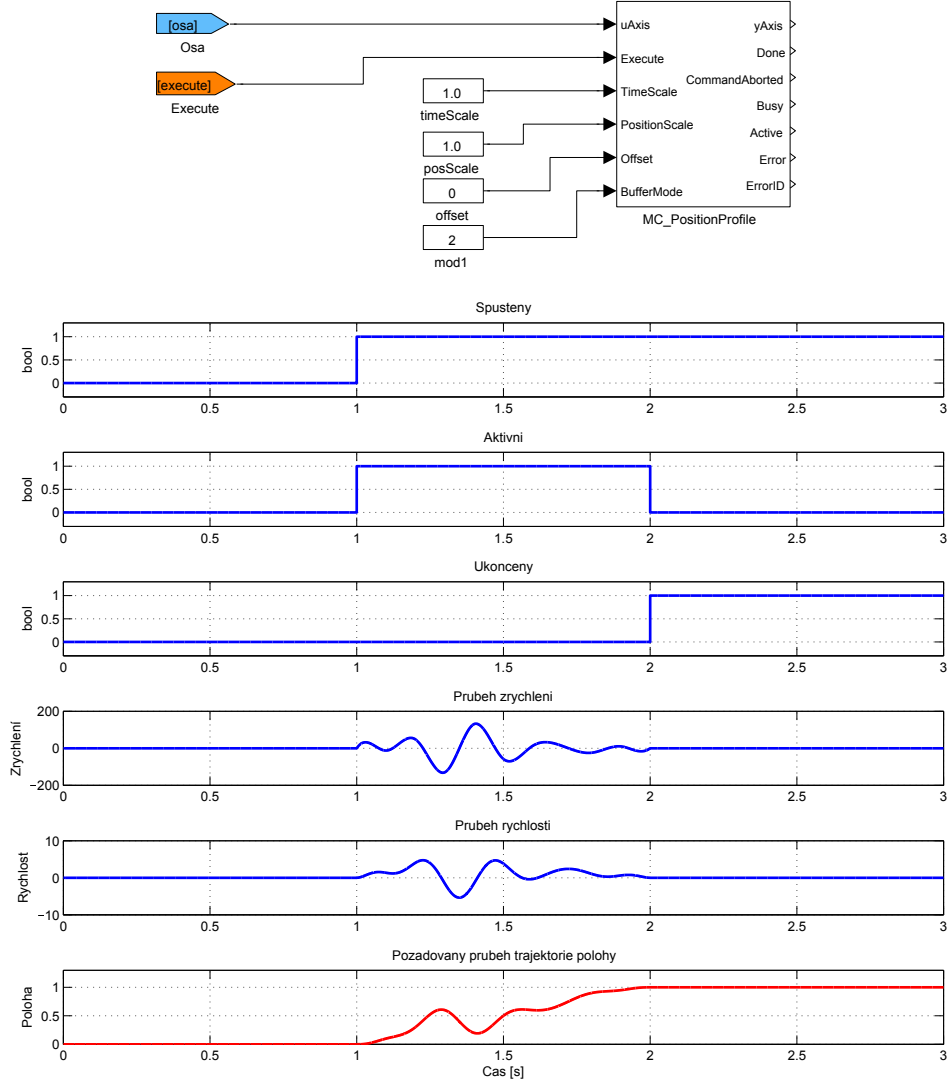
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>alg</code>	Typ interpolace	⊙2 Long (I32)
	1 ..... tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 ..... interpolace polynomy 5. řádu	
	4 ..... polynomy 5. řádu s ekvidistantními intervaly	
	5 ..... tabulka čas/hodnota + počáteční a koncová derivace	
	6 ..... hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 ..... hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 ..... hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 ..... tabulka čas/hodnota (lineární interpolace)	
<code>nmax</code>	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
<code>TimeScale</code>	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
<code>PositionScale</code>	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
<code>Offset</code>	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
<code>uTimes</code>	vektor s časovými hodnotami	Reference
<code>uValues</code>	vektor s hodnotami polohy nebo koeficienty polynomů	Reference

<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>BeginVelocity</b>	počáteční hodnota rychlosti (jen alg=5 nebo 6)	Double (F64)
<b>EndVelocity</b>	koncová hodnota rychlosti (jen alg=5 nebo 6)	Double (F64)

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
i	..... obecná chyba systému REXYGEN	

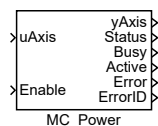
## Příklad



## MC\_Power – Aktivace osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC\_Power** musí být použit s každou osou. Je to jediný blok, který převádí osu ze stavu **Disabled** do stavu **StandStill** (tj. aktivní režim). Vstup **Enable** musí být nastaven po celou dobu práce s osou. Pokud je připojena fyzická osa/měnič (block **RM\_AxosSpline**), nejprve se zapne měnič a teprve potom přejde osa do stavu **Standstill**, popřípadě **ErrorStop** (pokud byla před vypnutím v chybě nebo pokud selhalo zapnutí měniče).

Pokud je osa vypnuta (nastavením vstupu **Enable** na nulu) při aktivním bloku (je nenulová rychlost), je nejprve aktivována zastavovací sekvence a teprve po jejím skončení je osa nastavena na stav vypnuto (disabled).

Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	Bool

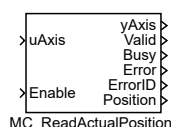
Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Status</b>	Skutečný stav napájení osy	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_ReadActualPosition – Skutečná poloha osy

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Blok `MC_ReadActualPosition` zpřístupňuje na výstupu `Position` aktuální polohu připojené osy. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Blok zobrazuje logickou polohu, tj. hodnoty, které se do všech MC bloků zadávají jako poloha a která je též na výstupu `CommandedPosition` bloku `RM_Axis`. Hodnota na snímači motoru (tj. hodnota na výstupu `PhysicalPosition` bloku `RM_Axis`) může být odlišná.

Poznámka: Tento blok je implementován z důvodu kompatibility s PLCopen neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu `CommandedPosition` bloku `RM_Axis`. Někdy se blok může hodit v rozsáhlých schématech se subsystémy.

## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

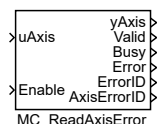
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error
<code>Position</code>	Aktuální poloha osy	Double (F64)

## MC\_ReadAxisError – Chyba osy

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MC_ReadAxisError` zpřístupňuje na výstupu `AxisErrorID` aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu `ErrorID` bloku [RM\\_Axis](#).

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

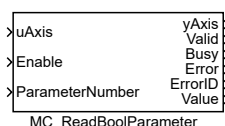
### Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error
<code>AxisErrorID</code>	Chybový kod přečtený z osy i ..... obecná chyba systému REXYGEN	Error

## MC\_ReadBoolParameter – Čtení parametru (logická hodnota)

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Blok `MC_ReadBoolParameter` zpřístupňuje na výstupu `Value` aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu `ParameterNumber`. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť zobrazuje parametry bloku `RM_Axis`.

## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	Long (I32)
	4 ..... kontrola kladného omezení polohy	
	5 ..... kontrola záporného omezení polohy	
	6 ..... kontrola regulační odchylky (poloha)	

## Výstupy

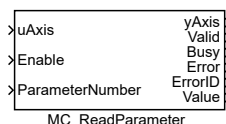
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
<code>Value</code>	Hodnota parametru	Bool



## MC\_ReadParameter – Čtení parametru (číselná hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MC_ReadParameter` zpřístupňuje na výstupu `Value` aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu `ParameterNumber`. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť zobrazuje parametry a výstupy bloku `RM_Axis`.

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	Long (I32)
	1 ..... požadovaná poloha	
	2 ..... kladné omezení polohy	
	3 ..... záporné omezení polohy	
	7 ..... maximální odchylka polohy	
	8 ..... maximální rychlost (systém)	
	9 ..... maximální rychlost (bloky)	
	10 .... skutečná rychlost	
	11 .... požadovaná rychlost	
	12 .... maximální zrychlení (systém)	
	13 .... maximální zrychlení (bloky)	
	14 .... maximální zpomalení (systém)	
	15 .... maximální zpomalení (bloky)	
	16 .... maximální změna zrychlení (jerk)	
	1000 .. skutečná poloha	
	1001 .. maximální síla/moment	
	1003 .. skutečná síla/moment	
	1004 .. požadovaná síla/moment	

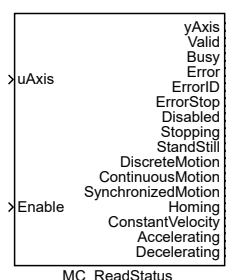
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error
<code>Value</code>	Hodnota parametru	Double (F64)

## MC\_ReadStatus – Stav osy

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MC_ReadStatus` indikuje na svých výstupech různé stavy připojené osy jako logickou hodnotu. indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

### Výstupy

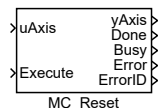
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error
<code>ErrorStop</code>	Osa je ve stavu CHYBA	Bool
<code>Disabled</code>	Osa je ve stavu VYPNUTO	Bool
<code>Stopping</code>	Osa je ve stavu STOP	Bool
<code>StandStill</code>	Osa je ve stavu PŘIPRAVEN	Bool
<code>DiscreteMotion</code>	Osa je ve stavu JEDNORÁZOVÝ POHYB	Bool
<code>ContinuousMotion</code>	Osa je ve stavu TRVALÝ POHYB	Bool
<code>SynchronizedMotion</code>	Osa je ve stavu SYNCHRONIZOVANÝ POHYB	Bool

Homing	Osa je ve stavu HLEDÁNÍ VÝCHOZÍ POLOHY	Bool
ConstantVelocity	Osa se pohybuje konstantní rychlostí	Bool
Accelerating	Osa zrychluje	Bool
Decelerating	Osa brzdí/zpomaluje	Bool

## MC\_Reset – Nulování chyb osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok `MC_Reset` převede připojenou osu ze stavu „ErrorStop“ do stavu „StandStill“ a vynuluje v ose všechny příznaky chyby. Je to v podstatě jediný blok, který ve stavu „ErrorStop“ nehlásí chybu.

Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

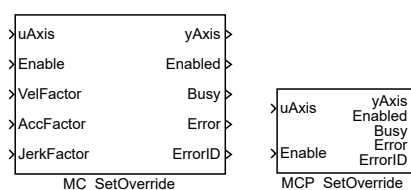
Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_SetOverride, MCP\_SetOverride – Nastavení násobivých faktorů na ose

### Symboly bloků

Licence: MOTION CONTROL



### Popis funkce

*Bloky MC\_SetOverride a MCP\_SetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_SetOverride` nastavuje násobivé faktory které se projeví ve všech blocích pracujících s osou. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je potřeba vynásobit faktorem z tohoto bloku, abychom dostali hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadaných v `RM_Axis` a administrativních bloků.

Tento blok není aktivován hranou, ale pokud je vstup `Enable` nenulový, tak se hodnoty trvale aktualizují. Pokud je aktivní blok typu `MC_MoveAbsolute`, vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr `diff`) a přepočet trajektorie je proveden až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu `errorID = -700` - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu `errorID = -701` - hodnota mimo rozsah).

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>VelFactor</code>	Faktor násobení pro rychlost	Double (F64)
<code>AccFactor</code>	Faktor násobení pro zrychlení	Double (F64)
<code>JerkFactor</code>	Faktor násobení pro změnu zrychlení	Double (F64)

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Enabled</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

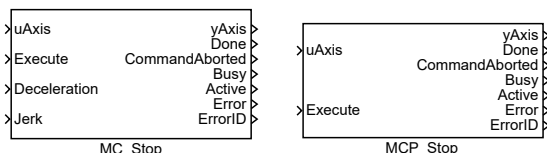
## Parametr

<code>diff</code>	Pásmo necitlivosti (pro přepočítání trajektorie)	↓0.0 ↑1.0 ⊙0.1	Double (F64)
-------------------	--	----------------	--------------

## MC\_Stop, MCP\_Stop – Zastavení pohybu

### Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_Stop a MCP\_Stop mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Block **MC\_Stop** provede zastavovací sekvenci a převede osu do stavu **Stopping**. V tomto stav není možné spustit žádný pohyb a osa v něm zůstává dokud je vstup **Execute** nenulový.

Poznámka1: Blok nemá parametr **BufferMode**. Mód je vždy **Aborting**.

Poznámka2: Protože selhání příkazu k zastavení může být nebezpečné, blok generuje chybu jen v naprosto fatálních případech (např. nezapojený vstup **uAxis**) a snaží se co nejkorektněji zastavit (např. při nekorektních parametrech použije nastavení osy nebo vyvolá sekvenci pro chybové zastavení).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>JerK</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

### Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool

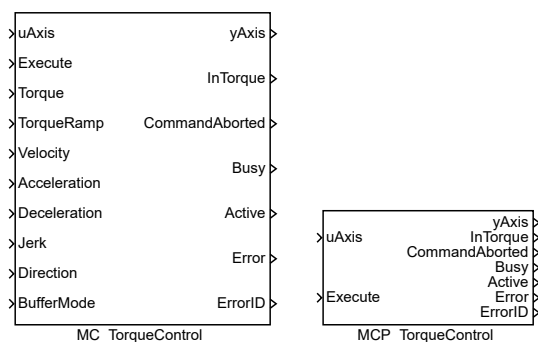


ErrorID	Výsledek poslední operace	Error
i . . . . .	obecná chyba systému REXYGEN	

## MC\_TorqueControl, MCP\_TorqueControl – Řízení síly/momentu

Symboly bloků

Licence: MOTION CONTROL



## Popis funkce

Bloky MC\_TorqueControl a MCP\_TorqueControl mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC\_TorqueControl generuje požadovaný moment/sílu nejprve s konstantním nárůstem (parametr TorqueRamp) a po dosažení maximální hodnoty (parametr Torque) je již moment/síla konstantní. Pohyb osy je řízen podle požadovaného momentu tak, aby nebyly překročeny maximální hodnoty rychlosti, zrychlení/zpomalení a případně jerku.

## Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Torque	Maximální povolený moment/síla	Double (F64)
TorqueRamp	Maximální povolená změna momentu/síly	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
Deceleration	Maximální povolené zpomalení [uunit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

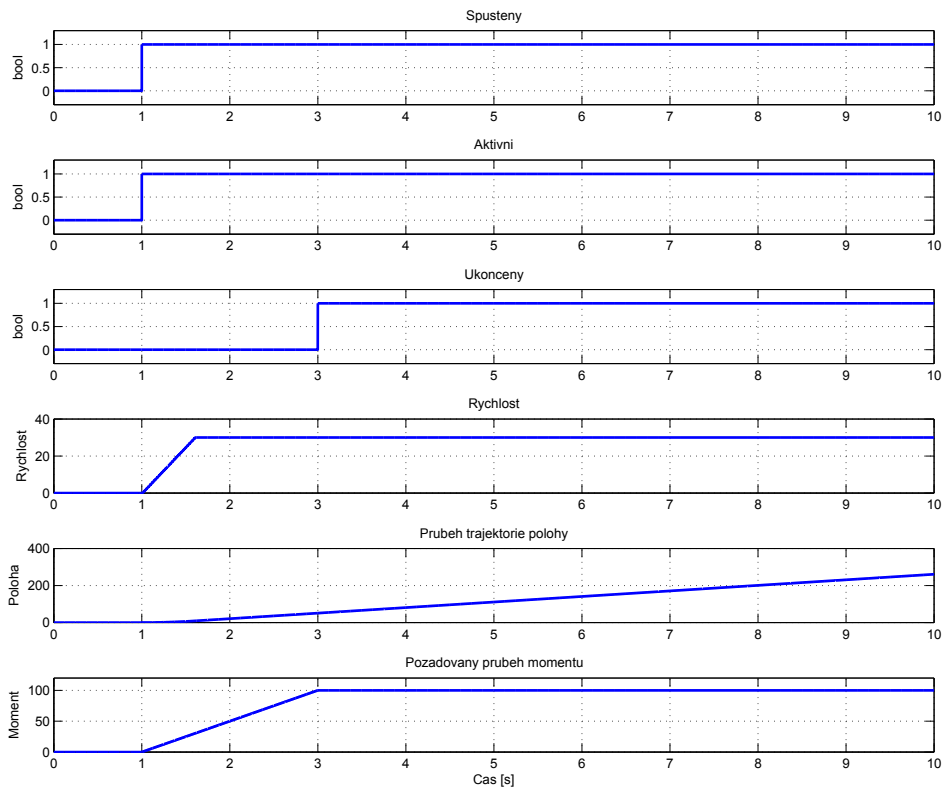
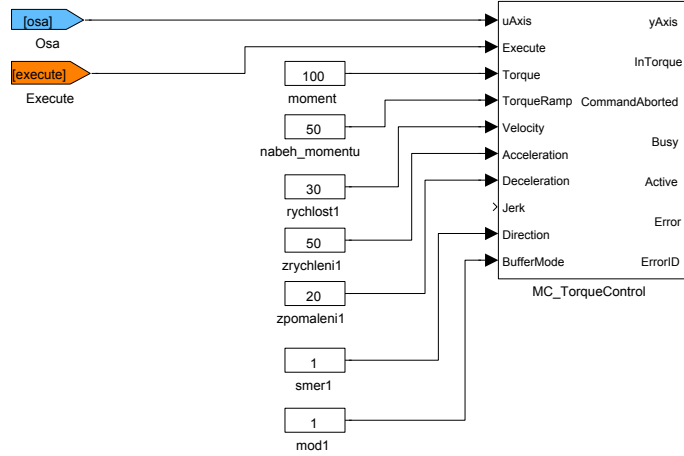
## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<b>InTorque</b>	Příznak dosažení požadovaného momentu/síly	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## Parametr

<b>kma</b>	Poměr mezi silou/momentem a zrychlením	Double (F64)
------------	--	--------------

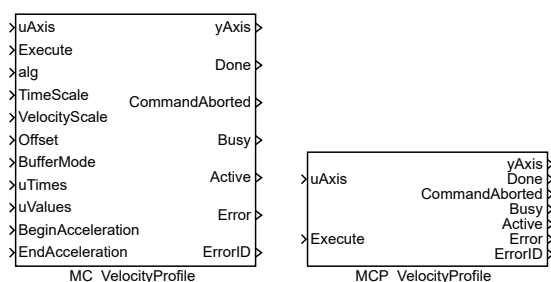
## Příklad



## MC\_VelocityProfile, MCP\_VelocityProfile – Generování trajektorie (rychlost)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_VelocityProfile a MCP\_VelocityProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_VelocityProfile` generuje takovou trajektorii, aby rychlost byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a rychlost. Mezi jednotlivými časy se hodnota rychlosti interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru `times`, příslušné hodnoty rychlosti do pole/parametru `values`. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli `times`. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koefficienty  $a_i$  jsou uloženy v poli `values` ve vzestupném pořadí (tj. pole `values` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli `uTimes` jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Vstup/parametr `uValues` musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 2: Dialog je jednotný pro `MC_AccelerationProfile`, `MC_VelocityProfile`, `MC_PositionProfile`. Některé režimy (parametr `alg` nedávají pro `VelocityProfile` smysl

a nefungují (aproximace B-spline nefunguje, režimy 1,2,5,6 používají lineární interpolaci)

Poznámka 3: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak měnit. `VelocityScale` a `Offset` je samozřejmě pro rychlost. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 4: Blok neobsahuje tzv. ramp-in mode. Pokud tedy rychlost osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému lze předejít, pokud se použije `BufferMode=BlendingNext` nebo je potřeba správně nastavit parametr `Offset`.

Poznámka6: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací `PLCopen`).

## Vstupy

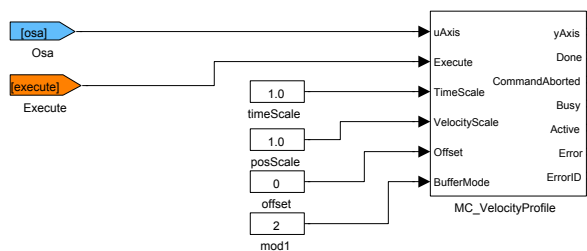
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>alg</code>	Typ interpolace	⊙2 Long (I32)
	1 ..... tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 ..... interpolace polynomy 5. řádu	
	4 ..... polynomy 5. řádu s ekvidistantními intervaly	
	5 ..... tabulka čas/hodnota + počáteční a koncová derivace	
	6 ..... hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 ..... hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 ..... hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 ..... tabulka čas/hodnota (lineární interpolace)	
<code>nmax</code>	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
<code>TimeScale</code>	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
<code>VelocityScale</code>	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
<code>Offset</code>	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
<code>uTimes</code>	vektor s časovými hodnotami	Reference

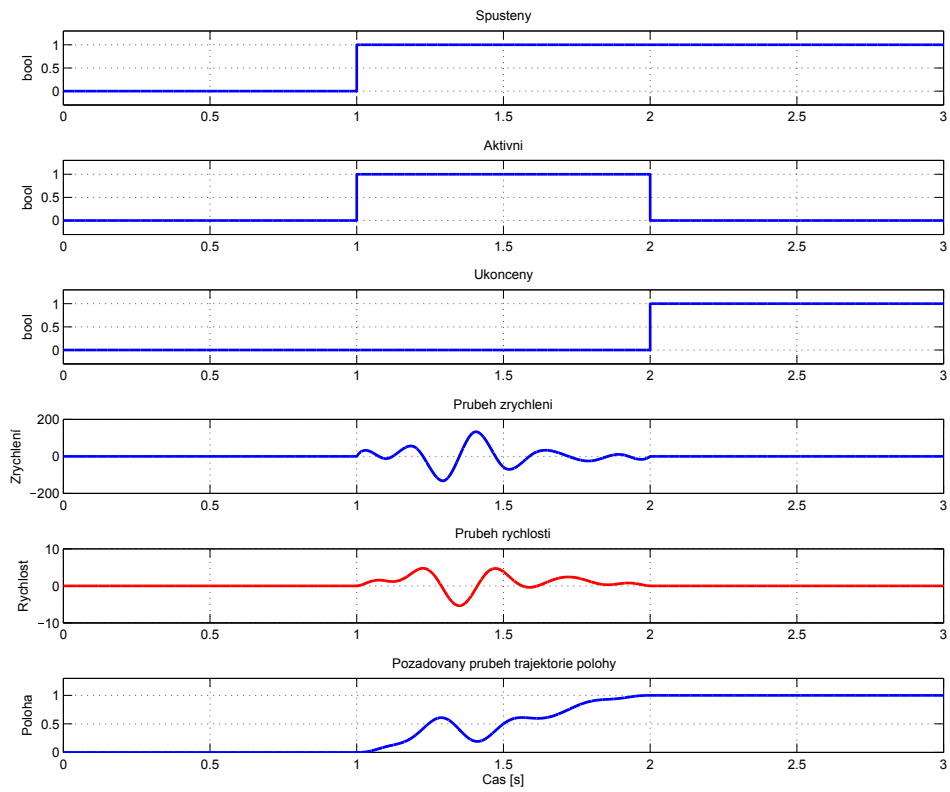
<b>uValues</b>	vektor s hodnotami rychlosti nebo koeficienty polynomů	<b>Reference</b>
1 .....	Aborting (nový blok se spustí okamžitě)	
2 .....	Buffered (nový blok se spustí po dokončení předchozího)	
3 .....	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4 .....	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5 .....	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6 .....	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>BeginAcceleration</b>	počáteční hodnota zrychlení (jen alg=5 nebo 6)	<b>Double (F64)</b>
<b>EndAcceleration</b>	koncová hodnota (jen alg=5 nebo 6)	<b>Double (F64)</b>

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>Reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>Bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace	<b>Error</b>
i .....	obecná chyba systému REXYGEN	

## Příklad



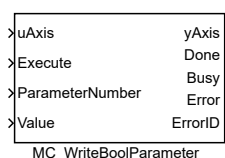




## MC\_WriteBoolParameter – Nastavení parametru (logická hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MC_WriteBoolParameter` změní hodnotu parametru připojené osy na hodnotu danou vstupem `Value`. Číslo požadovaného parametru musí být přivedeno na vstup `ParameterNumber`.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť nastavuje parametry bloku `RM_Axis`, což umožňuje i blok `SETPB`.

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	Long (I32)
	4 ..... kontrola kladného omezení polohy	
	5 ..... kontrola záporného omezení polohy	
	6 ..... kontrola regulační odchylky (poloha)	
<code>Value</code>	Hodnota parametru	Bool

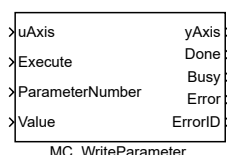
### Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## MC\_WriteParameter – Nastavení parametru (číselná hodnota)

Symbol bloku

Licence: MOTION CONTROL



### Popis funkce

Blok `MC_WriteParameter` změní hodnotu parametru připojené osy na hodnotu danou vstupem `Value`. Číslo požadovaného parametru musí být na vstupu `ParameterNumber`.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť nastavuje parametry bloku `RM_Axis`, což umožňuje i blok `SETPR`.

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	Long (I32)
	2 ..... kladné omezení polohy	
	3 ..... záporné omezení polohy	
	7 ..... maximální odchylka polohy	
	8 ..... maximální rychlost (systém)	
	9 ..... maximální rychlost (bloky)	
	13 .... maximální zrychlení (bloky)	
	15 .... maximální zpomalení (bloky)	
	16 .... maximální změna zrychlení (jerk)	
	1001 .. maximální síla/moment	
<code>Value</code>	Hodnota parametru	Double (F64)

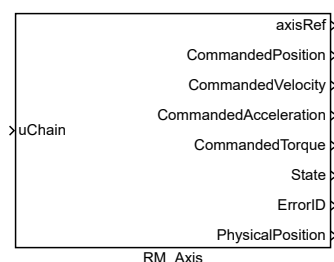
### Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## RM\_Axis – Osa pro řízení pohybu

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **RM\_AXIS** je základní blok osy pro řízení pohybu. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry osy. Algoritmus tohoto bloku se stará o kontrolu nastavených mezí, havarijní zastavení v případě potřeby a přepočty všech stavů a výstupů pro případ, že žádný blok není aktivní, ale osa (motor) má nenulovou rychlost. Výstupem tohoto bloku jsou pouze požadované hodnoty polohy, rychlosti, popřípadě zrychlení a momentu. Pokud nejde o virtuální osu a osa je spojena s reálným motorem, je nutné realizovat ještě regulátor polohy a rychlosti, který není součástí tohoto bloku – zpětnovazební signály jsou použity pouze pro kontrolu správné činnosti regulátoru, například odchylka nesmí překročit stanovenou mez, viz dále. Dále je mohou využívat některé speciální bloky, například bloky pro nalezení výchozí polohy. Zpětnovazební signál se připojují pomocí bloku [RM\\_AxisSpline](#).

Parametry tohoto bloku jsou stejné, jaké vyžaduje **PLCopen** pro osu. Pokud jsou zadány parametry nesprávně (nekonzistentně), výstup **errorID** je nastaven na hodnotu -700 (neplatný parametr) a všechny ostatní bloky navázané na osu skončí s chybou -703 (chybný stav).

Parametry pro omezení rychlosti a zrychlení jsou dvojí. Aplikační parametry omezují hodnoty, které lze zadat do pohybových bloků, ale skutečná rychlost nebo zrychlení může být někdy větší. Systémové parametry se překročit nesmí a pokud k tomu dojde, je pohyb zastaven a osa přejde do stavu **errorstop**.

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

### Vstupy

**uChain** blok vstup nepoužívá, ale je možné připojit libovolný signál a tím definovat pořadí spouštění bloků

## Výstupy

<code>axisRef</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>CommandedPosition</code>	Požadovaná poloha osy Poloha, která se zadává do pohybových bloků. V případě homingu nebo cyklické osy se liší od <code>PhysicalPosition</code>	Double (F64)
<code>CommandedVelocity</code>	Požadovaná rychlost osy	Double (F64)
<code>CommandedAcceleration</code>	Požadované zrychlení osy	Double (F64)
<code>CommandedTorque</code>	Požadovaný moment/síla	Double (F64)
<code>State</code>	Stav osy pro řízení pohybu	Long (I32)
	0 ..... Disabled (osa blokována)	
	1 ..... Stand still (osa připravena)	
	2 ..... Homing (hledání výchozí polohy)	
	3 ..... Discrete motion (jednorázový pohyb)	
	4 ..... Continuous motion (trvalý pohyb)	
	5 ..... Synchronized motion (pohyb synchronní s jinou osou)	
	6 ..... Coordinated motion (osa řízena blokem pro koordinovaný pohyb)	
	7 ..... Stopping (zastavování nebo dočasné blokování)	
	8 ..... Error stop (zastavování nebo blokování osy po chybě)	
	9 ..... Drive error (jako errorstop, ale chyba je způsobena vnějším signálem - obvykle chyba měniče)	
<code>ErrorID</code>	Kod chyby ve stavu errorstop	Error
	i ..... obecná chyba systému REXYGEN	
<code>PhysicalPosition</code>	Požadovaná poloha osy Poloha, která se zapojuje do zpětnovazebního regulátoru. V případě homingu nebo cyklické osy se liší od <code>CommandedPosition</code>	Double (F64)

## Parametry

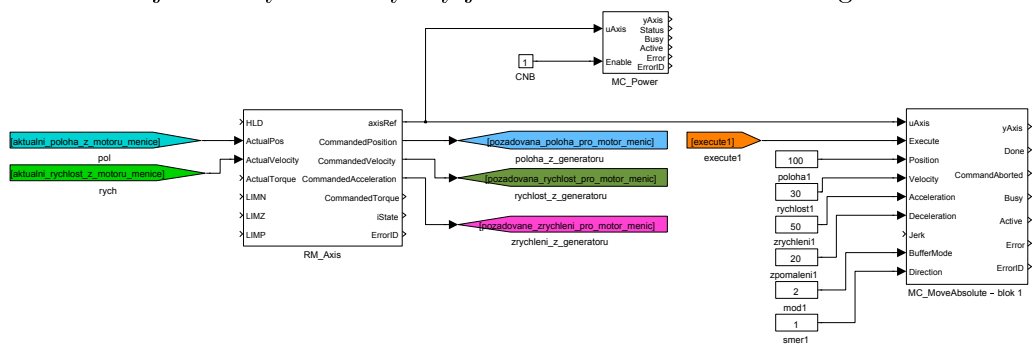
<code>AxisType</code>	Druh osy	☉1	Long (I32)
	1 ..... lineární osa		
	2 ..... cyklická osa s cyklickým polohovým senzorem		
	3 ..... cyklická osa s lineárním polohovým senzorem		
	4 ..... lineární osa v testovacím režimu. V tomto režimu jsou vypnuté některé kontroly, zejména polohové limity (včetně koncových spínačů). Tento typ je určen pro pokusy ožívování, někdy se může hodit i pro trvalý provoz na cyklických osách.		
<code>EnableLimitPos</code>	Příznak kontroly maximální polohy v kladném směru (tj. když je zaškrtnuto, je platné <code>MaxPosAppl</code> )		Bool
<code>MaxPosAppl</code>	Maximální hodnota v kladném směru pro pohybové bloky. Hodnota musí být menší než <code>MaxPosSystem</code> pro lineární osu. Pro cyklickou osu s lineárním senzorem hodnota omezuje na několik otáček (vhodné pro robotické aplikace) a musí být větší než <code>MaxPosSystem</code> .		Double (F64)

<b>MaxPosSystem</b>	Maximální hodnota v kladném směru (fyzický doraz)	Double (F64)
<b>EnableLimitNeg</b>	Příznak kontroly maximální polohy v záporném směru (tj. když je zaškrtnuto, je platné <b>MinPosAppl</b> )	Bool
<b>MinPosAppl</b>	Maximální hodnota v záporném směru pro pohybové bloky Hodnota musí být větší než (nastává dříve) <b>MinPosSystem</b> pro lineární osu. Pro cyklickou osu s lineárním senzorem hodnota omezuje na několik otáček (vhodné pro robotické aplikace) a musí být menší než (nastává později) <b>MaxPosSystem</b> .	Double (F64)
<b>MinPosSystem</b>	Maximální hodnota v záporném směru (fyzický doraz)	Double (F64)
<b>EnablePosLagMonitor</b>	Příznak kontroly skluzu (rozdílu mezi požadovanou a skutečnou polohou)	Bool
<b>MaxPositionLag</b>	Maximální hodnota skluzu	Double (F64)
<b>MaxVelocitySystem</b>	Maximální hodnota rychlosti (havarijní)	Double (F64)
<b>MaxVelocityAppl</b>	Maximální hodnota rychlosti (pracovní)	Double (F64)
<b>MaxAccelerationSystem</b>	Maximální hodnota zrychlení (havarijní)	Double (F64)
<b>MaxAccelerationAppl</b>	Maximální hodnota zrychlení (pracovní)	Double (F64)
<b>MaxDecelerationSystem</b>	Maximální hodnota brždění (havarijní)	Double (F64)
<b>MaxDecelerationAppl</b>	Maximální hodnota brždění (pracovní)	Double (F64)
<b>DefaultJerk</b>	Maximální doporučená změna zrychlení [unit/s <sup>3</sup> ] Skutečná změna zrychlení se nekontroluje a může překročit tuto hodnotu.	Double (F64)
<b>MaxTorque</b>	Maximální hodnota momentu/síly	Double (F64)
<b>TorqueRatio</b>	Poměr momentu a zrychlení. Pokud moment osy není určen jiným způsobem (např. pomocí bloku <b>MC_TorqueControl</b> ), určuje se moment ze zrychlení pomocí tohoto koeficientu.	Double (F64)
<b>LoopDelay</b>	zpoždění mezi požadovanou a skutečnou hodnotou [s] Zpoždění nastává vlivem periody vzorkování, komunikace s externím kontrolérem, zpožděním regulační smyčky. Nastavení umožňuje přesněji kontrolovat <b>PositionLag</b> . (not yet implemented)	Double (F64)
<b>StartMode</b>	Volby režimu při zapnutí	⊙1 Long (I32)
	1 ..... při zapnutí je pohyb zastaven	
	2 ..... při zapnutí se pohyb pokračuje aktuální rychlostí motoru	
<b>HomingRequired</b>	Pokud je zaškrtnuto, tak jsou blokovány všechny pohyby (kromě <b>MC_Home</b> ), dokud se neprovede homing	Bool

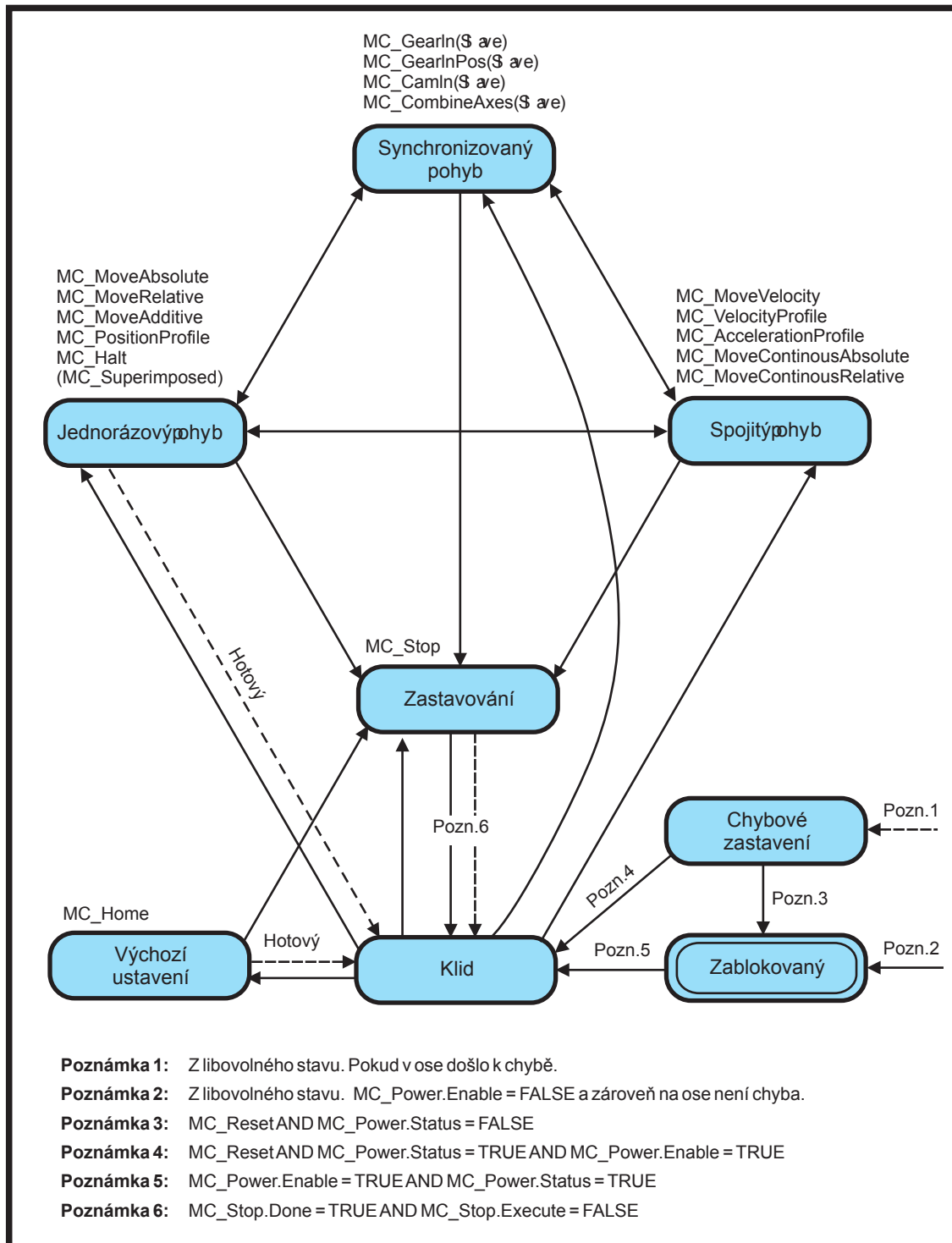
## Příklad

Následující schéma znázorňuje základní zapojení bloků z knihovny Motion Control v minimální realizaci nutné pro zahájení pohybu. Fyzická osa je reprezentována blokem **RM\_Axis** a jí příslušnou datovou strukturou. Po nastavení omezení na požadovaný průběh pohybu lze na vstupy bloku připojit signál aktuální polohy, rychlosti a momentu (využíváno pro kontrolu skluzu) popřípadě logické signály koncových spínačů pro nalezení referenční výchozí polohy. Výstupní signál **axisRef** se připojuje na vstup všech bloků pracujících s danou osou. Před zahájením pohybu je třeba osu aktivovat blokem **MC\_Power**, čímž dojde k přechodu ze zablokovaného do klidového stavu. V tuto chvíli

je již možné zahájit libovolný jednorázový, spojitý nebo synchronizovaný pohyb spuštěním příslušného funkčního bloku. Generovaná trajektorie pohybu, tedy požadovaný průběh polohy, rychlosti a zrychlení je k dispozici na výstupech bloku **RM\_Axis**. Odtud může být přiveden do regulační struktury pohonu, která existuje buďto lokálně v systému REXYGEN ve stejném nebo jiném tasku nebo je implementována přímo v řídicí jednotce motoru (zesilovač, frekvenční měnič), kam může být ze systému REXYGEN předána s použitím některé ze standardních sériových komunikací. V případě zadání chybných parametrů, volání nepřipustného příkazu nebo při překročení maximální hodnoty skluzu dochází k nouzovému zastavení, osa přechází do režimu chyby, kterou indikuje na výstupu **ErrorID**. Pro opětovné zahájení pohybu je třeba chybu kvitovat blokem **MC\_Reset**. Přechod mezi jednotlivými režimy osy je znázorněn na stavovém diagramu.



## Stavový diagram osy



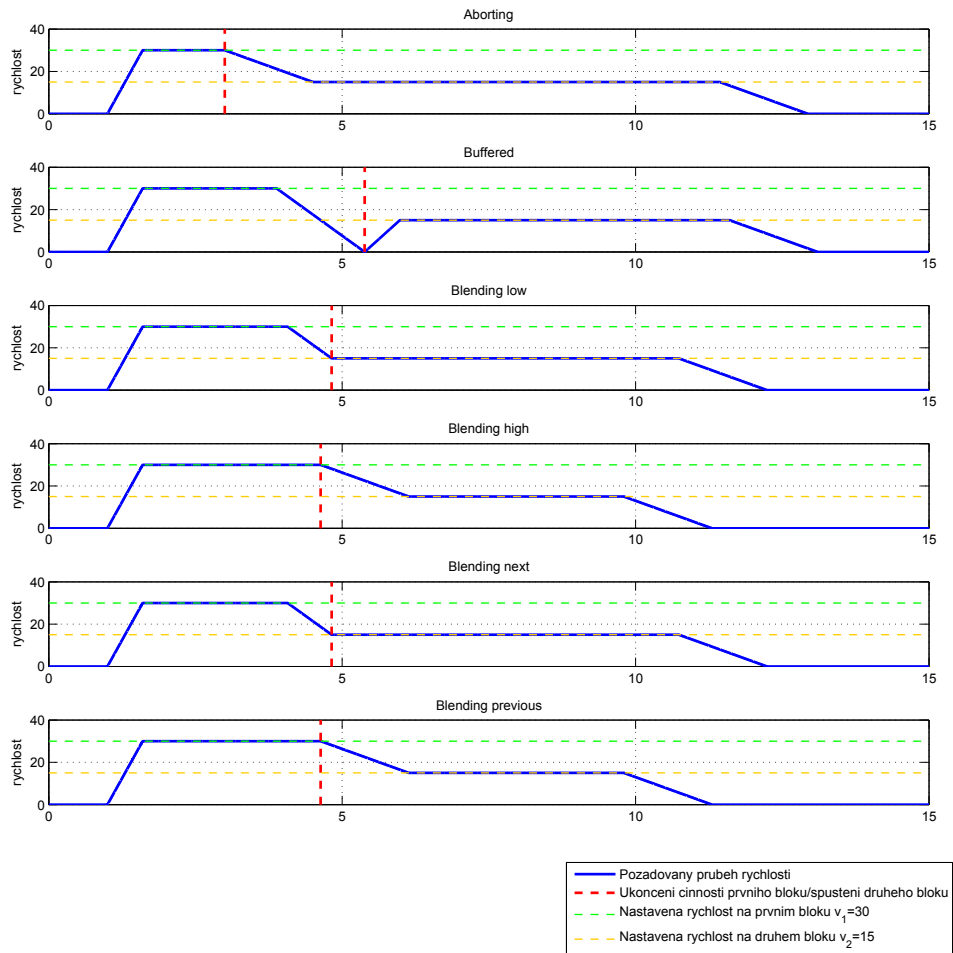
## Míchání pohybů

Norma PLCOpen definuje pro bloky spouštějící pohyb osy vstupní parametr **BufferMode**, který určuje chování osy v případě, že je za běhu zavolán nový příkaz ve formě jiného funkčního bloku. Tento přechod mezi různými pohyby (míchání, "Blending") lze řešit několika způsoby. Následující tabulka podává stručné vysvětlení funkce jednotlivých režimů míchání pohybu a jejich vliv na tvar výsledné generované trajektorie. Detailní popis lze nalézt ve specifikaci PLCOpen.

<b>Aborting</b>	nový pohyb okamžitě přerušuje původní příkaz
<b>Buffered</b>	nový pohyb je vykonán po dokončení původního příkazu, neprobíhá žádné míchání
<b>Blending low</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejnižší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupech dvou bloků, které přebírají řízení osy
<b>Blending high</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejvyšší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupech dvou bloků, které přebírají řízení osy
<b>Blending previous</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu prvního bloku
<b>Blending next</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu druhého bloku



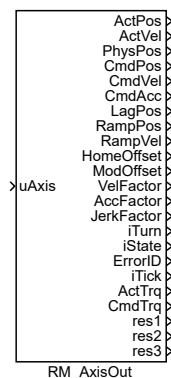
## Ilustrace míchání pohybů



## RM\_AxisOut – Výstupní blok osy

Symbol bloku

Licence: [MOTION CONTROL](#)



## Popis funkce

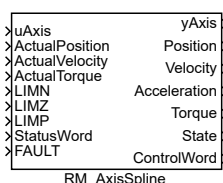
Blok **RM\_AxisOut** zpřístupňuje důležité stavy bloku **RM\_Axis**. Některé z výstupů jsou přímo na bloku **RM\_Axis**, ale ty jsou o krok zpožděné. Aby nebyly výstupy zpožděné i na tomto bloku, je potřeba je zařadit jako poslední.

Poznámka: Téměř všechny bloky nepracují s momentem, proto je na příslušném výstupu 0. Obvykle je tento signál používán jako dopředná vazba pro regulátor rychlosti, tak to nepředstavuje problém.

## RM\_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Tento blok slouží k připojení virtuální osy (reprezentované blokem `RM_Axis`) k motoru respektive jeho řídicím obvodům a vytvoření fyzické osy. To zahrnuje několik do značné míry nezávislých funkcí, které realizuje tento blok.

Základní funkce je připojit požadované a aktuální signály o poloze, rychlosti, zrychlení a momentu. To pokrývají vstupy `ActualPosition`, `ActualVelocity`, `ActualTorque` a výstupy `Position`, `Velocity`, `Acceleration`, `Torque`.

Měníč motoru obvykle pracuje s jinými jednotkami (často přímo tiky na snímači polohy) než potřebujeme v ose. Dále polohu ještě ovlivňuje převodový poměr na převodovce. Blok `RM_AxisSpline` provádí transformaci jednotek podle parametrů `DriveUnits` a `AxisUnits` (zadáva se sobě odpovídající délka v jednotkách měniče a v jednotkách osy v bloku `RM_Axis`). Předpokládá se, že rychlost na vstupu i výstupu bloku `RM_AxisSpline` je `drive units/s` a zrychlení `driveunits/s/s`; moment se netransformuje.

Řídicí obvody motoru obvykle pracují s celočíselnými typy (respektive v celočíselných typech předávají hodnoty nadřazenému systému). Problém vzniká u polohy, kde při dlouhotrvajícím pohybu jedním směrem může dojít k přetečení celočíselného typu a hodnota se změní z maximálního použitelného čísla na minimální nebo naopak. Blok `RM_AxisSpline` s tímto umí pracovat a správně naváže polohu. Podmínkou správné funkce je, že v parametru `DriveBits` je nastaven správný počet bitů (včetně příznaku, jestli je číslo se znaménkem nebo bez) a také tuto vlastnost musí podporovat měnič (řídicí obvody motoru).

Měníče mají různé stavy a režimy činnosti a definované sekvence, jak se mezi těmito stavy přechází. Nejčastěji je to podle standardu CiA402. Blok `RM_AxisSpline` tento standard podporuje a to prostřednictvím vstupu `StatusWord`, výstupu `ControlWord` a parametrů `DriveMode`, `DriveTimeout` (vstup a výstup se připojuje na stejnojmenné signály v měniči, další je zřejmé z popisu parametrů).

Mnoho bloků pro řízení pohybu obsahuje výpočetně náročný algoritmus. To vede na relativně velké vzorkovací periody (typicky mezi 10 a 200 ms). Naproti tomu regulátor

motoru vyžaduje vzorkovací periodu malou (typicky do 1ms), aby nedocházelo k trhavému pohybu. Tyto protichůdné požadavky řeší blok `RM_AxisSpline`, který může běžet v jiné úloze (s kratší periodou vzorkování) než blok `RM_Axis` a provádí interpolaci hodnot, tak aby výsledná křivka byla spojitá a pokud možno hladká.

Je mnoho možností, jak dopočítat polohu (a rychlost, zrychlení, moment) v časech mezi hodnotami z pomalejší úlohy. Protože každá možnost má určité výhody a nevýhody, může si uživatel z několika vybrat pomocí parametru `InterpolationMode`. Podporované metody jsou:

- 1: **linear**: Poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0 (tj. rychlost je po částech konstantní funkce se skoky).
- 2: **cubic spline**: Poloha je polynom 3. řádu, vypočtený na základě polohy a rychlosti na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 3: **quintic spline**: Poloha je polynom 5. řádu, vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 4: **cubic approximation (B-spline)**: Poloha je polynom 3. řádu, vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 5: **quintic approximation (B-spline)**: Poloha je polynom 5. řádu, vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 6: **all linear**: Poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti.
- 7: **all cubic**: Poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy.
- 8: *rezervováno pro pozdější použití.*
- 9: *rezervováno pro pozdější použití.*

Nejjednodušší je lineární interpolace polohy. To však způsobuje skoky v rychlosti. Alternativně lze rychlost také interpolovat lineárně, ale pak zase rychlost neodpovídá derivaci/diferenci polohy. Z tohoto pohledu lepší je interpolace polynomem vyššího řádu (v parametru lze vybrat 3. nebo 5.), což ovšem vede na velké zrychlení, pokud vstupní trajektorie neodpovídá polynomu (nastává při přechodu z konstantního zrychlení na konstantní rychlost uprostřed mezi body vzorkování i při jednoduchých pohybech). Další

možnost je aproximace B-spline funkcí (opět lze vybrat 3. nebo 5. řád). To odstraňuje nevýhody předchozích metod, ale vyžaduje více vzorků a tak zavádí větší dopravní zpoždění. Někdy také může vadit, že je to aproximační metoda a neprojízdí zadané body přesně (v režimu konstantní rychlosti je ale aproximace přesná). Ve všech případech je moment interpolován lineárně a nezávisle na ostatních veličinách.

Poznámka 1: Protože doba vykonávání bloků pro řízení pohybu značně kolísá, dostává interpolátor nové hodnoty neekvidistantně, přičemž výstup interpolátoru musí být souvislý. Proto se načte několik (závisí na metodě) hodnot dopředu a používají se hodnoty z vhodného intervalu. Právě používaný interval indikuje blok na výstupu `State` pomocí hodnot `interval1`, `interval2`, `interval3`. Důležité je pravidlo, že stavy odpovídající kladným číslům jsou normální činnost a stavy odpovídající záporným číslům představují chybu.

Poznámka 2: Protože činnost interpolátoru z principu vyžaduje použití v jiné (rychlejší) úloze než je `RM_Axis`, musí mít blok správně vyřešeno předávání reference mezi úlohami a referenci je možné bez obav předávat pomocí bloků `Inport` a `Outport`.

## Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>ActualPosition</code>	Skutečná poloha osy [drive unit]	Double (F64)
<code>ActualVelocity</code>	Skutečná rychlost osy [drive unit/s]	Double (F64)
<code>ActualTorque</code>	Skutečný moment/síla osy	Double (F64)
<code>LIMN</code>	Koncový spínač v záporném směru	Bool
<code>LIMZ</code>	Koncový spínač pro nulovou polohu	Bool
<code>LIMP</code>	Koncový spínač v kladném směru	Bool
<code>StatusWord</code>	Stavové slovo servoměniče dle CiA402 standardu	Long (I32)
<code>FAULT</code>	Vynucení vypnutí osy (obvykle chyba měniče nebo komunikace s měničem)	Bool

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Position</code>	Požadovaná (interpolovaná) poloha [drive unit]	Double (F64)
<code>Velocity</code>	Požadovaná (interpolovaná) rychlost [drive unit/s]	Double (F64)
<code>Acceleration</code>	Požadované (interpolované) zrychlení [drive unit/s/s]	Double (F64)
<code>Torque</code>	Požadovaný (interpolovaný) moment/síla	Double (F64)

<b>State</b>	Stav interpolátoru (záporné hodnoty znamenají chybu a pokud to není nesprávnou periodou, je potřeba zkontrolovat časování obou úloh)	Long (I32)
0	..... Neaktivní (interpolátor jen přeposílá vstupní data na výstup; nastává při ose ve stavu vypnuto nebo pokud bloky RM_Axis RM_AxisSpline mají stejnou periodu spouštění)	
1	..... Čekání (čekání, až bude dostatek dat v bufferu)	
2	..... Interval1 (probíhá interpolace, data z prvního intervalu)	
3	..... Interval2 (probíhá interpolace, data z prvního intervalu)	
x	..... Intervalx (další kladná čísla znamenají další interpolační intervaly; závisí na metodě, ale měla by se střídat jen 2 výjimečně 3 sousední čísla )	
-1	..... Přetečení (přetečení vyrovnávacího bufferu, tj. jsou odebírána pomaleji, než odpovídá nastavené periodě; algoritmus se restartuje automaticky, ale může dojít ke skoku polohy i rychlosti)	
-2	..... Podtečení (podtečení vyrovnávacího bufferu, tj. data přichází pomaleji, než odpovídá nastavené periodě; algoritmus se restartuje automaticky, ale může dojít ke skoku polohy i rychlosti)	
-3	..... Přetížení (data z bloku RM_Axis se nepodařilo přečíst; pravděpodobně je přetížena úloha s blokem RM_Axis)	
-4	..... Špatná perioda (úloha s blokem RM_AxisSpline má delší periodu než úloha s blokem RM_Axis)	
<b>ControlWord</b>	Řídící slovo servoměniče dle CiA402 standardu	Long (I32)

## Parametry

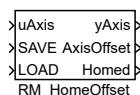
<b>InterpolationMode</b>	Algoritmus pro interpolaci	⊙9 Long (I32)
1	..... linear	
2	..... cubic spline	
3	..... quintic spline	
4	..... cubic aproximation (B-spline)	
5	..... quintic aproximation (B-spline)	
6	..... all linear	
7	..... all cubic	
8	..... —	
9	..... —	
<b>ReverseLimit</b>	Negace významu vstupů LIMN, LIMZ a LIMP	Bool

<b>InterpolationMode</b>	Režim řízení servoměniče	⊙9	Long (I32)
1	..... Zjednodušený CiA402 (ze statuswordu je kontrolován jen bit značící chybu, control word se nastavuje okamžitě bez postupné zapínací sekvence)		
2	..... Úplný CiA402 (úplná kontrola StatusWord v každém stavu, do ControlWord se posílá požadovaná sekvence a kontroluje se, jestli měnič na každý krok reaguje, pokud nereaguje do času daného parametrem DriveTimeout nastává chyba)		
<b>DriveTimeout</b>	Čas pro odpověď servoměniče [s] (for Strict CiA402 mode only)		Double (F64)
<b>DriveBits</b>	Počet platných bitů (záporné číslo znamená číslo se znaménkem) v polohovém vstupu ActualPosition a výstupu Position		Long (I32)
		↓-64 ↑63 ⊖-32	
<b>DriveUnits</b>	Vzdálenost v jednotkách měniče (pro transformaci polohy, vzdálenost odpovídající AxisUnits)		Double (F64)
<b>AxisUnits</b>	Vzdálenost v jednotkách osy (pro transformaci polohy, vzdálenost odpovídající DriveUnits)		Double (F64)
<b>VelocityCalculate</b>	když je zaškrtnuto, vstup ActualVelocity se neuplatňuje a rychlost je dopočítávána z rozdílu polohy (v aktuálním a předchozím kroku)		Bool

## RM\_HomeOffset – \* Nastavení počáteční polohy pomocí offsetu

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>SAVE</b>	Nastavení offsetu z osy do bloku	<b>Bool</b>
<b>LOAD</b>	Nastavení offsetu z bloku do osy	<b>Bool</b>

### Parametr

<b>SavedOffset</b>	Rozdíl mezi logickou a fyzickou polohou (pro nastavení/načtení z osy)	<b>Double (F64)</b>
--------------------	---	---------------------

### Výstup

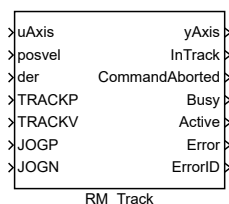
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	<b>Reference</b>
<b>AxisOffset</b>	Aktuální rozdíl fyzické a logické polohy v ose	<b>Bool</b>
<b>Homed</b>	Příznak nastavení výchozí polohy	<b>Bool</b>



## RM\_Track – Sledování a krokování

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `RM_Track` sdružuje několik užitečných doplňkových funkcí.

Pokud je aktivní (tj. nenulový) vstup `TRACK`, blok se snaží dosáhnout zadané polohy nebo rychlosti (vstup `posvel`) s respektováním omezení na rychlost, zrychlení/zpomalení a jerk. Rozdíl oproti bloku `MC_MoveAbsolute` je ten, že tento blok v každém kroku aktualizuje cílovou polohu nebo rychlost a přepočítává trajektorii. Tento režim je vhodný pro sledování trajektorie polohy nebo rychlosti generované mimo motion control subsystém. Pokud je trajektorie známa předem, je mnohem vhodnější a přesnější použít blok `MC_PositionProfile`.

Pokud je aktivní (tj. nenulový) vstup `JOGP`, blok jede maximální dovolenou rychlostí v kladném směru s respektováním maximálního zrychlení a jerku při rozjezdu. Po deaktivaci signálu blok zastaví pohyb (s respektováním maximálního zpomalení a případně jerku) a vzdá se řízení osy. Tento režim je vhodný pro najetí na výchozí polohu operátorem pomocí tlačítek vpřed a vzad.

Vstup `JOGN` má stejnou funkci jako `JOGP`, jen je pohyb v negativním směru.

Pokud je na ose již nějaký pohyb aktivní, je tímto blokem přerušen (chová se tedy jako `BufferMode=aborting`). Pokud je sepnuto více funkcí, vykonává se ta s nejvyšší prioritou a ostatní jsou ignorovány. Pořadí je `TRACK`, `JOGP`, `JOGN`. Takovém režimu je však vhodné se vyhnout, protože chování není dostatečně intuitivní a výsledek je pak nečekaný.

Poznámka: pokud jsou všechny limity (rychlost, acc, dec, jerk) 0, limity se nekontrolují a jsou rovnou předány do osy. Vstupní signál však musí mít dostatečně malé zrychlení a rychlost, jinak osa přejde do stavu `ErrorStop`.

### Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>posvel</code>	Požadovaná poloha nebo rychlost [unit, unit/s]	Double (F64)
<code>TRACKP</code>	Zapnutí režimu sledování polohy	Bool
<code>TRACKV</code>	Zapnutí režimu sledování rychlosti	Bool

JOGP	Zapnutí režimu posunu v kladném směru	Bool
JOGN	Zapnutí režimu posuvu v záporném směru	Bool

## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InTrack	Příznak dosažení požadované polohy v režimu sledování polohy	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## Parametry

pv	Maximální povolená rychlost [unit/s]	Double (F64)
pa	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
pd	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
pj	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
iLen	Počet kroků pro odhad rychlosti	⊙10 Long (I32)

## Kapitola 21

# MC\_MULTI – Řízení pohybu více OS

### Obsah

---

MCP_CamIn – * Zapnutí vačky . . . . .	645
MCP_CamTableSelect – Definice vačky . . . . .	647
MCP_CombineAxes – * Kombinace pohybu dvou os do třetí . . . . .	649
MCP_GearIn – * Zapnutí konstantního převodového poměru . . . . .	651
MCP_GearInPos – * Zapnutí konstantního převodového poměru v zadané pozici . . . . .	653
MCP_PhasingAbsolute – * Vytvoření fázového posunu (absolutní souřadnice) . . . . .	655
MCP_PhasingRelative – * Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	657
MC_CamIn, MCP_CamIn – Zapnutí vačky . . . . .	659
MC_CamOut – Vypnutí vačky . . . . .	663
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí . . . . .	665
MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru . . . . .	668
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici . . . . .	671
MC_GearOut – Vypnutí konstantního převodového poměru . . . . .	676
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice) . . . . .	678
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	681

---

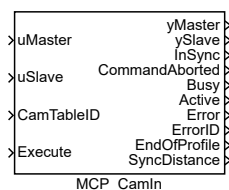
Knihovna MC\_MULTI se specializuje na řízení pohybu ve více osách. Obsahuje bloky jako `MC_CombineAxes` pro synchronizaci více os, `MC_GearIn` a `MC_GearOut` pro

převody a `MC_PhasingAbsolute`, `MC_PhasingRelative` pro přesné fázování os. Knihovna nabízí `MC_CamIn` a `MC_CamOut` pro funkce vačky, které umožňují sledování složitých profilů pohybu. `MCP_CamTableSelect` navíc poskytuje flexibilitu při výběru vačkových tabulek a `MC_GearInPos` umožňuje řazení na základě polohy. Tato knihovna je nezbytná pro pokročilé aplikace vyžadující koordinované řízení pohybu napříč více osami.

## MCP\_CamIn – \* Zapnutí vačky

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<b>uMaster</b>	Odkaz na hlavní osu	Reference
<b>uSlave</b>	Odkaz na podřízenou osu	Reference
<b>CamTableID</b>	Odkaz na vačku (spojit s MCP_CamTableSelect.CamTableID)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

## Parametr

<b>MasterOffset</b>	Posunutí v definici vačky na straně hlavní osy [unit]	Double (F64)
<b>SlaveOffset</b>	Posunutí v definici vačky na straně podřízené osy [unit]	Double (F64)
<b>MasterScaling</b>	Měřítko pro definici vačky (strana hlavní osy)	⊙1.0 Double (F64)
<b>SlaveScaling</b>	Měřítko pro definici vačky (strana podřízené osy)	⊙1.0 Double (F64)
<b>StartMode</b>	Volba absolutního nebo relativního profilu vačky	⊙4 Long (I32)
	1 ..... hlavní osa relativní podřízená osa absolutní	
	2 ..... hlavní osa absolutní podřízená osa relativní	
	3 ..... obě osy relativní	
	4 ..... obě osy absolutní	
<b>BufferMode</b>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<b>RampIn</b>	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)

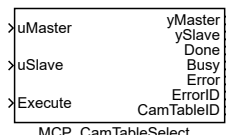
## Výstup

<code>yMaster</code>	Odkaz na hlavní osu	Reference
<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>InSync</code>	Příznak dosažení profilu vačky podřízenou osou	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby REXYGENu	Error
<code>EndOfProfile</code>	Příznak dosažení konce profilu vačky (jen necyklické vačky)	Bool
<code>SyncDistance</code>	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

## MCP\_CamTableSelect – Definice vačky

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MCP_CamTableSelect` spolupracuje s blokem `MCP_CamIn` a definuje vačku jako spojitou funkci jedné proměnné. Možnosti definování této funkce jsou analogické, jako v bloku `MC_PositionProfile`, tj. máme dvě možnosti:

1. tabulkou: zadávají se dvojice čísel poloha hlavní osy a poloha podřízené osy. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolaci lze také zvolit, ale není příliš vhodná, protože na okrajích intervalu pak je skok v rychlosti). Hodnoty polohy hlavní osy se zadávají do pole/parametru `mValues`, příslušné hodnoty polohy podřízené osy do pole/parametru `sValues`. Posloupnost hodnot `mValues` musí být stoupající.

2. polynomy: celá funkce se v hlavní ose (tj. v nezávislé proměnné) rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Polohy hlavní osy a tím i příslušné intervaly se definují jako v předchozím případě v poli `mValues`. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli `sValues` ve vzestupném pořadí (tj. pole `sValues` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly, pak je v poli `mValues` jen počáteční a koncová poloha.

Poznámka: pokud je parametr `alg=1` nebo `alg=5` nebo `alg=9` je možné nechat pole `mValues` prázdné (popř. nepřipojený vstup) a pak zadávat celou vačku v poli `sValues`, kde první sloupec jsou polohy master osy, druhý sloupec polohy slave osy, třetí (nepovinný) sloupec derivace, čtvrtý (nepovinný) sloupec druhá derivace.

### Vstupy

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

## Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
CamTableID	Odkaz na vačku (spojit s <code>MC_CamIn.CamTableID</code> )	Reference
	i ..... obecná chyba systému REXYGEN	

## Parametry

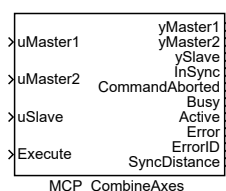
camname	Jméno souboru, kam si speciální editor ukládá data (pokud je parametr prázdný, zvolí se automaticky podle jména bloku)	String
nmax	Maximální počet segmentů profilu	⊙3 Long (I32)
alg	Typ interpolace	⊙2 Long (I32)
	1 ..... tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 ..... interpolace polynomy 5. řádu	
	4 ..... polynomy 5. řádu s ekvidistantními intervaly	
	5 ..... tabulka čas/hodnota + počáteční a koncová derivace	
	6 ..... hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 ..... hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 ..... hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 ..... tabulka čas/hodnota (lineární interpolace)	
Periodic	Příznak cyklické vačky (konec navazuje na začátek)	⊙on Bool
BeginRate	Počáteční strmost (derivace ds/dm) vačky (jen pro alg=5 a alg=6)	Double (F64)
EndRate	Počáteční strmost (derivace ds/dm) vačky (jen pro alg=5 a alg=6)	Double (F64)
mValues	Posloupnost hraničních pozic jednotlivých segmentů na hlavní ose	⊙[0 30] Double (F64)
sValues	Posloupnost poloh řízené osy nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 0] Double (F64)



## MCP\_CombineAxes – \* Kombinace pohybu dvou os do třetí

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

uMaster1	Odkaz na první hlavní osu	Reference
uMaster2	Odkaz na druhou hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

### Parametr

GearRatioNumeratorM1	Čítec převodového poměru master osy 1	⊙1	Long (I32)
GearRatioDenominatorM1	Jmenovatel převodového poměru master osy 1		Long (I32)
		⊙1	
GearRatioNumeratorM2	Čítec převodového poměru master osy 2	⊙1	Long (I32)
GearRatioDenominatorM2	Jmenovatel převodového poměru master osy 2		Long (I32)
		⊙1	
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		
CombineMode	režim sloučení os	⊙1	Long (I32)
	1 ..... součet		
	2 ..... rozdíl		
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)		Double (F64)

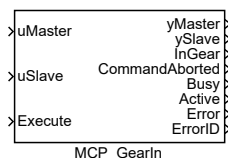
## Výstup

yMaster1	Odkaz na první hlavní osu	Reference
yMaster2	Odkaz na druhou hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . kód chyby REXYGENu	Error
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

## MCP\_GearIn – \* Zapnutí konstantního převodového poměru

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>RatioNumerator</code>	Převodový poměr - čítel (podřízená osa)	⊙1	Long (I32)
<code>RatioDenominator</code>	Převodový poměr - jmenovatel (hlavní osa)	⊙1	Long (I32)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]		Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]		Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]		Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		

### Výstup

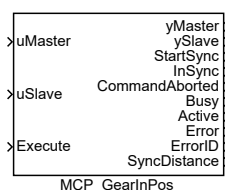
<code>yMaster</code>	Odkaz na hlavní osu	Reference
<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>InGear</code>	Příznak dosažení požadované rychlosti řízenou osou	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool

<b>Active</b>	Příznak, že blok řídí osu	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_GearInPos – \* Zapnutí konstantního převodového poměru v zadané pozici

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

### Parametr

RatioNumerator	Převodový poměr - číselník (podřízená osa)	⊙1	Long (I32)
RatioDenominator	Převodový poměr - jmenovatel (hlavní osa)	⊙1	Long (I32)
MasterSyncPosition	Poloha hlavní osy v okamžiku zasynchronizování [unit]		Double (F64)
SlaveSyncPosition	Poloha podřízené osy v okamžiku zasynchronizování		Double (F64)
MasterStartDistance	Definuje polohu hlavní osy pro spuštění procesu synchronizace		Double (F64)
Velocity	Maximální povolená rychlost [unit/s]		Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]		Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]		Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]		Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		

SyncMode	Režim synchronizace (pouze cyklické osy)	⊙2	Long (I32)
	1 . . . . . synchronizovat zrychlením		
	2 . . . . . synchronizovat na nejbližší bod		
	3 . . . . . synchronizovat zpomalením		

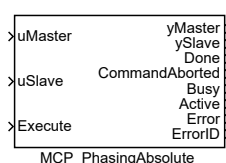
## Výstup

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
StartSync	Začíná synchronizace pohybu	Bool
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i . . . . . kód chyby REXYGENu	
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

## MCP\_PhasingAbsolute – \* Vytvoření fázového posunu (absolutní souřadnice)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>PhaseShift</code>	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	

### Výstup

<code>yMaster</code>	Odkaz na hlavní osu	Reference
<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool

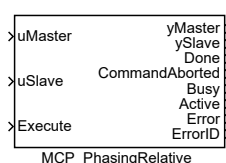
ErrorID	Výsledek poslední operace	Error
i	..... kód chyby REXYGENu	



## MCP\_PhasingRelative – \* Vytvoření fázového posunu (relativně k pozici při spuštění)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>PhaseShift</code>	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	

### Výstup

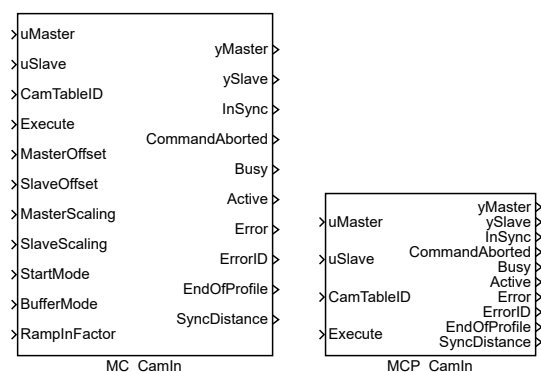
<code>yMaster</code>	Odkaz na hlavní osu	Reference
<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool

ErrorID	Výsledek poslední operace i ..... kód chyby REXYGENu	Error
---------	---	-------

## MC\_CamIn, MCP\_CamIn – Zapnutí vačky

### Symboly bloků

Licence: MOTION CONTROL



### Popis funkce

*Bloky MC\_CamIn a MCP\_CamIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_CamIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž převodní funkce je určena blokem MCP\_CamTableSelect připojeným ke vstupu CamTableID. Pokud převodní funkci označíme  $Cam(\cdot)$ , polohu hlavní osy  $PosM$  a polohu podřízené osy  $PosS$ , pak platí (pro absolutní režim, bez fázování):

$$PosS = Cam((PosM - MasterOffset) / MasterScaling) * SlaveScaling + SlaveOffset$$

Tento režim osy je často nazýván elektronická vačka.

Režim vačky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu aborting nebo spuštěním bloku MC\_CamOut. Pokud vačka (její definiční funkce - viz popis bloku není cyklická MCP\_CamTableSelect), dojde k ukončení režimu vačky také při vyjetí hlavní osy z definičního oboru funkce vačky. Toto je signalizováno výstupem EndOfProfile.

Při aktivaci funkce vačky (tj. v okamžiku, kdy blok MC\_CamIn převezme řízení osy) nemusí poloha a rychlost (popř. i zrychlení, ale současná implementace při zapnutí vačky jerk neuvažuje a připouští skok ve zrychlení) odpovídat požadovaným hodnotám, tj. poloze a rychlosti hlavní osy a profilu vačky. V takovém případě záleží na hodnotě parametru RampIn. Pokud má parametr hodnotu 0, režim vačky se nezapne a je signalizována chyba -707 (skok v poloze nebo rychlosti). Pokud je parametr kladný, nastává přechodový děj, kdy poloha ještě neodpovídá poloze podle definice vačky - tzv. RampIn režim.

Parametr **RampIn** přibližně odpovídá rychlosti, kterou by se podřízená osa pohybovala během RampIn režimu, kdyby hlavní osa stála. Pokud se hlavní osa pohybuje, pak je výsledný pohyb určen součtem pohybu daného definicí vačky a pohybu daného RampIn režimem se stojící hlavní osou. Pokud je potřeba RampIn režim použít, volíme parametr **RampIn** přibližně 0,1 až 0,5 maximální rychlosti podřízené osy a pokud dojde k chybě překročení maximální rychlosti nebo zrychlení během RampIn režimu, tak jej zmenšíme.

## Vstupy

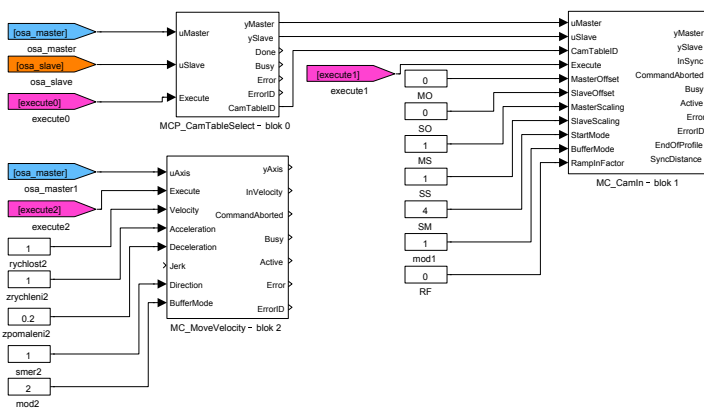
<b>uMaster</b>	Odkaz na hlavní osu	Reference
<b>uSlave</b>	Odkaz na podřízenou osu	Reference
<b>CamTableID</b>	Odkaz na vačku (spojit s <code>MCP_CamTableSelect.CamTableID</code> )	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>MasterOffset</b>	Posunutí v definici vačky na straně hlavní osy [unit]	Double (F64)
<b>SlaveOffset</b>	Posunutí v definici vačky na straně podřízené osy [unit]	Double (F64)
<b>MasterScaling</b>	Měřítko pro definici vačky (strana hlavní osy)	Double (F64)
<b>SlaveScaling</b>	Měřítko pro definici vačky (strana podřízené osy)	Double (F64)
<b>StartMode</b>	Volba absolutního nebo relativního profilu vačky	Long (I32)
	1 ..... hlavní osa relativní podřízená osa absolutní	
	2 ..... hlavní osa absolutní podřízená osa relativní	
	3 ..... obě osy relativní	
	4 ..... obě osy absolutní	
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>RampIn</b>	RampIn faktor (0 = RampIn režim se nepoužívá); odpovídá přibližně rychlosti synchronizace [unit/s] podřízené osy na polohu vačky	Double (F64)

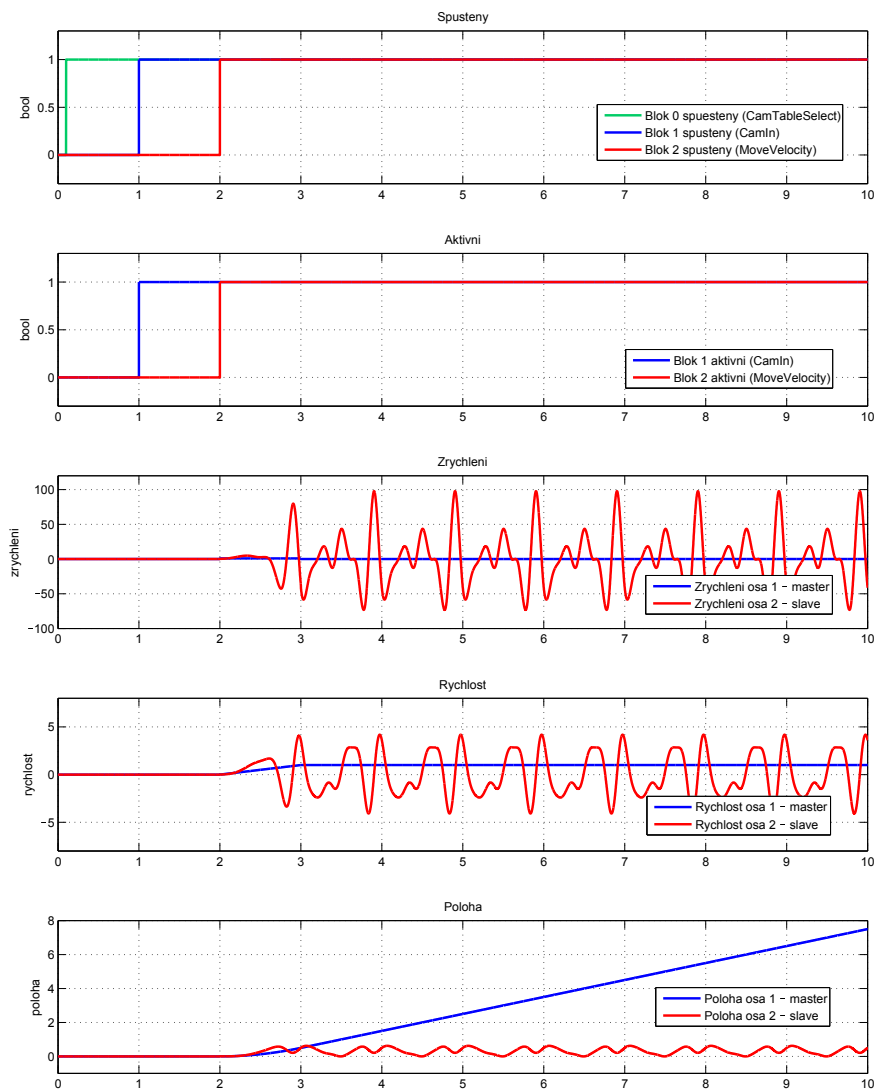
## Výstupy

<b>yMaster</b>	Odkaz na hlavní osu	Reference
<b>ySlave</b>	Odkaz na podřízenou osu	Reference
<b>InSync</b>	Příznak dosažení profilu vačky podřízenou osou	Bool

CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error
EndOfProfile	Příznak dosažení konce profilu vačky (u necyklické vačky konec pohybu)	Bool
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

## Příklady

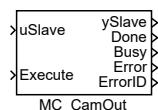




## MC\_CamOut – Vypnutí vačky

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok MC\_CamOut ukončuje režim vačky zapnutý blokem [MC\\_CamIn](#). Pokud žádná vačka není aktivní, blok nemá žádnou funkci (a ani nehlásí chybu).

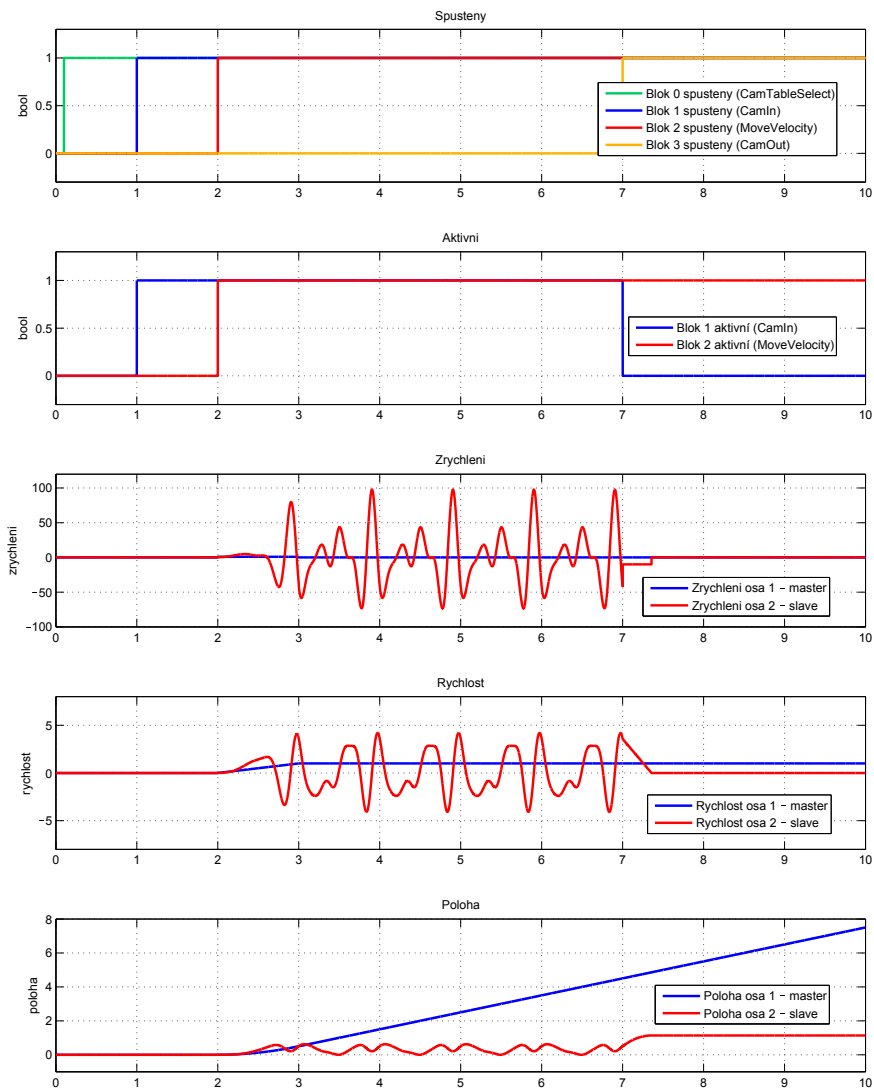
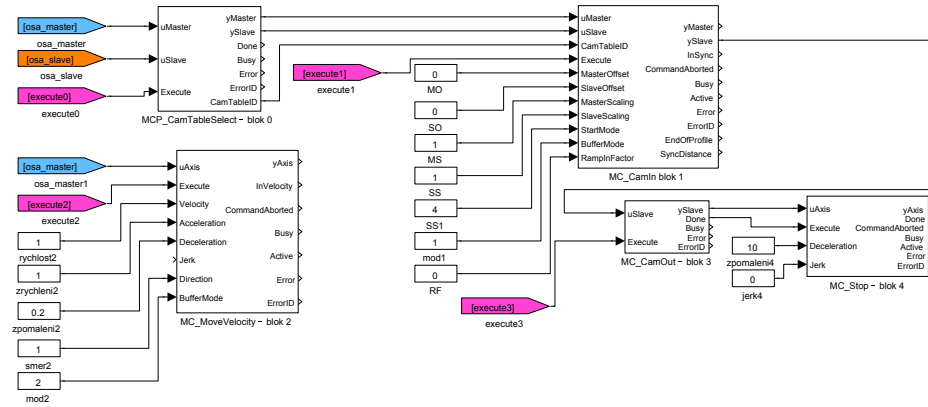
Vstupy

uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Výstupy

ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## Příklad

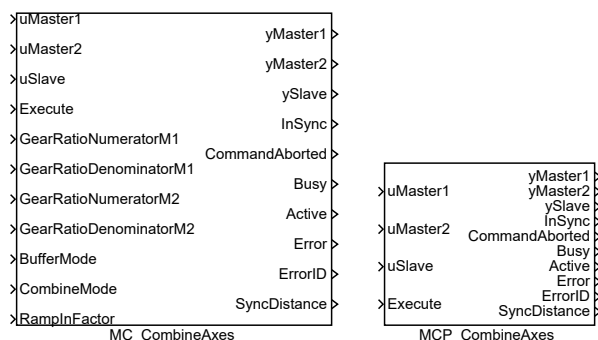




## MC\_CombineAxes, MCP\_CombineAxes – Kombinace pohybu dvou os do třetí

### Symboly bloků

Licence: MOTION CONTROL



### Popis funkce

*Bloky MC\_CombineAxes a MCP\_CombineAxes mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_CombineAxes kombinuje pohyb dvou os do třetí. V podstatě se jedná o výpočet nové žádané pozice na základě dvou poloh. Platí, že

$$\text{SlavePosition} = \text{Master1Position} \cdot \frac{\text{GearRatioNumeratorM1}}{\text{GearRatioDenominatorM1}} + \text{Master2Position} \cdot \frac{\text{GearRatioNumeratorM2}}{\text{GearRatioDenominatorM2}}$$

Blok umožňuje zadat do parametru GearRatio... záporné číslo a výsledný pohyb podřízené osy může být rozdíl poloh obou hlavních os. Primárně by se měl rozdíl realizovat pomocí parametru CombineMode

### Vstupy

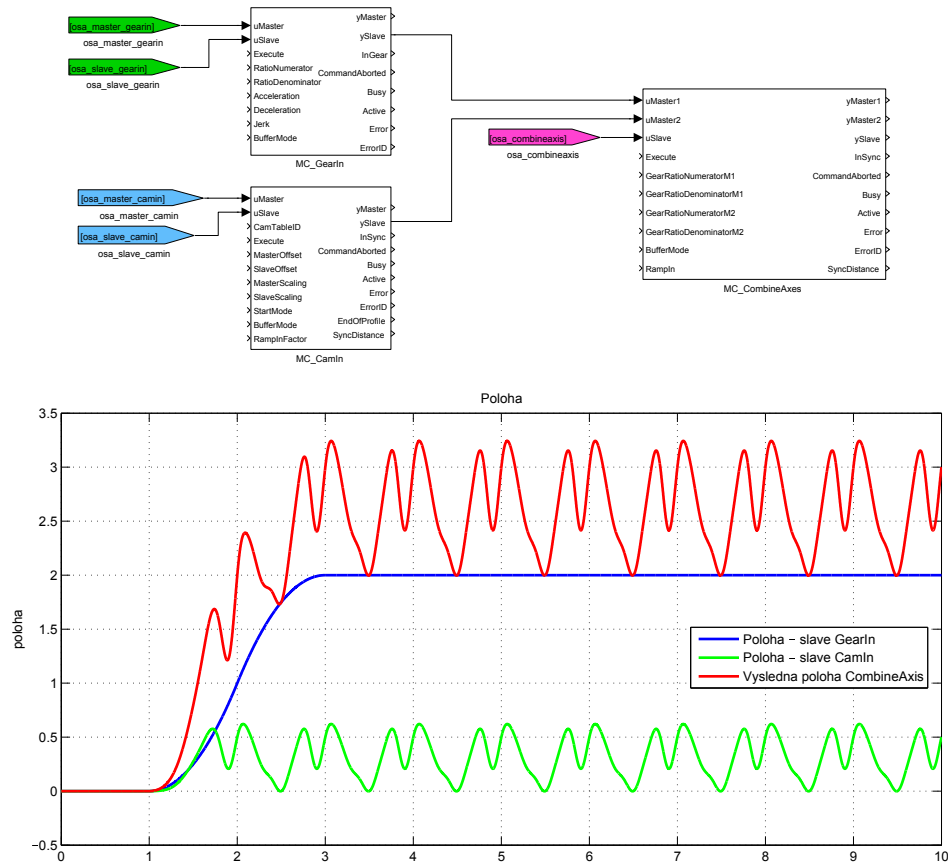
uMaster1	Odkaz na první hlavní osu	Reference
uMaster2	Odkaz na druhou hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
GearRatioNumeratorM1	Čítec převodového poměru master osy 1	Long (I32)
GearRatioDenominatorM1	Jmenovatel převodového poměru master osy 1	Long (I32)
GearRatioNumeratorM2	Čítec převodového poměru master osy 2	Long (I32)

<b>GearRatioDenominatorM2</b>	Jmenovatel převodového poměru master osy 2	Long (I32)
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>CombineMode</b>	Režim sloučení os	⊙1 Long (I32)
	1 ..... součet	
	2 ..... rozdíl	
<b>RampInFactor</b>	RampIn faktor (0 = RampIn režim se nepoužívá); rychlost synchronizace [unit/s] podřízené osy na polohu vačky odpovídá přibližně $\text{RampIn} * \text{MaxVelocityAppl}$ , kde $\text{MaxVelocityAppl}$ se zadává v bloku <code>RM_Axis</code> připojeného ke vstupu <code>uSlave</code> ; analogicky zrychlení	Double (F64)

## Výstupy

<b>yMaster1</b>	Odkaz na první hlavní osu	Reference
<b>yMaster2</b>	Odkaz na druhou hlavní osu	Reference
<b>ySlave</b>	Odkaz na podřízenou osu	Reference
<b>InSync</b>	Příznak dosažení profilu vačky podřízenou osou	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
<b>SyncDistance</b>	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

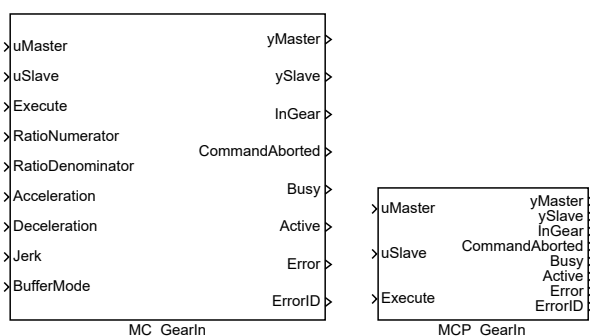
## Příklad



## MC\_GearIn, MCP\_GearIn – Zapnutí konstantního převodového poměru

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

*Bloky MC\_GearIn a MCP\_GearIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GearIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž poměr rychlostí obou os je dán parametry RatioNumerator a RatioDenominator. Pokud označíme rychlost hlavní osy  $VelM$  a rychlost podřízené osy  $VelS$ , pak platí (bez fázování):

$$VelS = VelM \cdot \frac{RatioNumerator}{RatioDenominator}$$

Tento režim osy je často nazýván elektronická převodovka. Poloha a zrychlení podřízené osy je dopočítávána konzistentně s uvedenou rychlostí.

Režim převodovky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu **aborting** nebo spuštěním bloku MC\_GearOut.

Při aktivaci funkce převodovky (tj. v okamžiku, kdy blok MC\_GearIn převezme řízení osy) nemusí rychlost (a popřípadě i zrychlení pokud jej požadujeme spojitě, tj.  $jerk \neq 0$ ) odpovídat požadované hodnotě rychlosti (a popř. i zrychlení) hlavní osy a převodovému poměru. V takovém případě nastává přechodový děj, kdy rychlost ještě neodpovídá převodovému poměru - tzv. RampIn režim. V tomto RampIn režimu jsou použity parametry Acceleration, Deceleration, Jerk a blok řídí podřízenou osu tak, aby se co nejdříve (s respektováním omezení na zrychlení a popř. jerk) dostala do synchronního stavu.

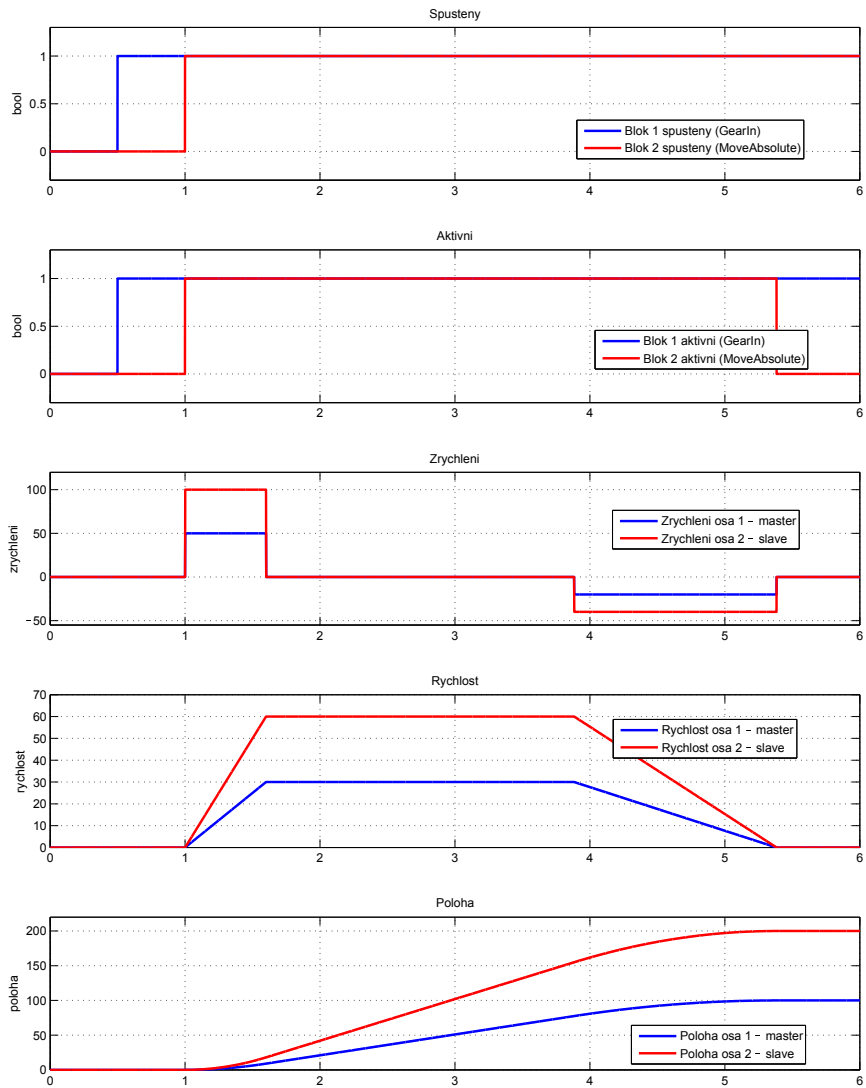
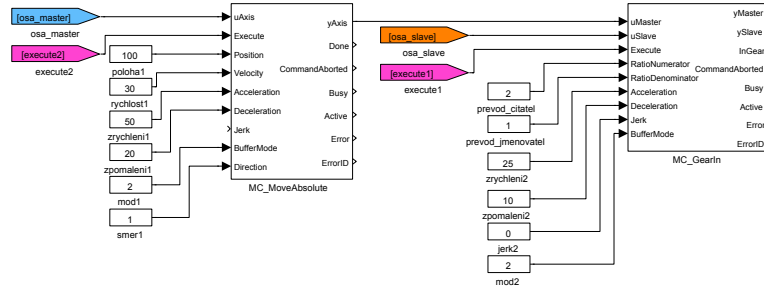
## Vstupy

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>RatioNumerator</code>	Převodový poměr - čítec (podřízená osa)	Long (I32)
<code>RatioDenominator</code>	Převodový poměr - jmenovatel (hlavní osa)	Long (I32)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<code>yMaster</code>	Odkaz na hlavní osu	Reference
<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>InGear</code>	Příznak dosažení požadované rychlosti řízenou osou	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

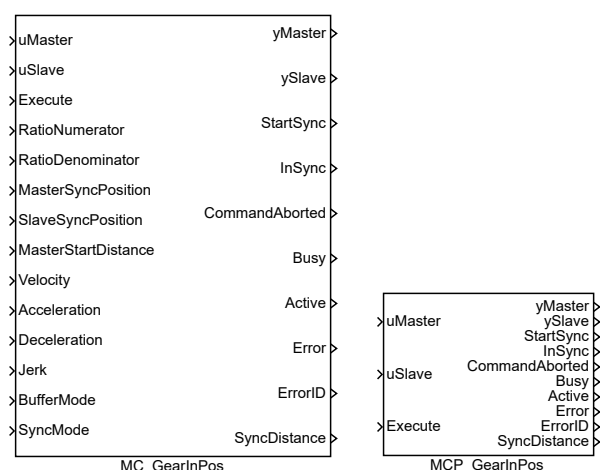
Příklad



## MC\_GearInPos, MCP\_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GearInPos a MCP\_GearInPos mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GearInPos zapne řízení podřízené osy tak, že poměr rychlostí hlavní a podřízené osy je v poměru RatioNumerator:RatioDenominator. Na rozdíl od bloku MC\_GearIn je u tohoto bloku zachován i poměr vzdáleností a je určena i poloha obou os v okamžiku zasnchronizování, tzn. platí

$$\frac{SlavePosition - SlaveSyncPosition}{MasterPosition - MasterSyncPosition} = \frac{RatioNumerator}{RatioDenominator}$$

Pokud v okamžiku aktivace bloku (tj. v okamžiku náběžné hrany na vstupu Execute a v případě buffered režimu ještě po ukončení předchozích bloků) neodpovídá poloha podřízené osy poloze hlavní osy (dle výše uvedeného vzorce), je spuštěn proces synchronizace (indikováno výstupem StartSync). Během tohoto procesu je generována trajektorie tak, aby došlo co nejrychleji k synchronizaci a nebyly překročeny maximální hodnoty rychlostí, zrychlení a změny zrychlení dané parametry Velocity, Acceleration, Deceleration, Jerk. Po dokončení synchronizace se již tyto limity neuplatňují. Pokud je parameter MasterStartDistance=0, proces synchronizace se spouští hned po aktivaci bloku (vstu-

pem Execute). V opačném případě se proces synchronizace spouští ve chvíli, kdy hlavní osa dosáhne polohy v intervalu `MasterSyncPosition ± MasterStartDistance`.

Poznámky:

1. Proces synchronizace používá dva algoritmy: I. Algoritmus shodný s blokem `MC_MoveAbsolute`, přičemž trajektorie je v každém kroku přepočítána tak, aby koncová rychlost odpovídala aktuální rychlosti hlavní osy. II. Poloha (i rychlost i zrychlení) se generuje jako v případě synchronního pohybu, ale k hodnotám je po určitou dobu přičítána hodnota vhodného polynomu 5. stupně, tak aby nenastal skok v poloze, v rychlosti a v případě zadaného jerku ani ve zrychlení a aby na konci doby přičítání měl polynom nulovou hodnotu. Doba je volena tak, aby polynom nepřekročil požadované meze rychlosti, zrychlení a jerku. První metoda nevede k úspěšné synchronizaci, pokud se hlavní osa pohybuje se zrychlením, druhá metoda negarantuje dodržení limitů na rychlost, zrychlení a jerk, navíc vyžaduje ponechat určitou rychlost a zrychlení na synchronní část, takže je obecně delší. Proto se oba algoritmy vhodně kombinují.

2. Parametry bloku (ať už start procesu synchronizace nebo limity na rychlost a zrychlení) je nutné volit tak, aby podřízená osa byla v poloze `SlaveSyncPosition` (přibližně) ve stejný okamžik, jako hlavní osa v poloze `MasterSyncPosition`. V opačné případě může nastat neočekávaný pohyb podřízené osy nebo porušení zadaných limitů. Pohyb hlavní osy může být libovolný, ale v konkrétní aplikaci bývá obvykle dobře definován. Správným nastavením parametrů je tedy možné zajistit vhodný průběh synchronizace.

## Vstupy

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>RatioNumerator</code>	Převodový poměr - čítec (podřízená osa)	Long (I32)
<code>RatioDenominator</code>	Převodový poměr - jmenovatel (hlavní osa)	Long (I32)
<code>MasterSyncPosition</code>	Poloha hlavní osy v okamžiku zasynchronizování [unit]	Double (F64)
<code>SlaveSyncPosition</code>	Poloha podřízené osy v okamžiku zasynchronizování	Double (F64)
<code>MasterStartDistance</code>	Definuje polohu hlavní osy pro spuštění procesu synchronizace	Double (F64)
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

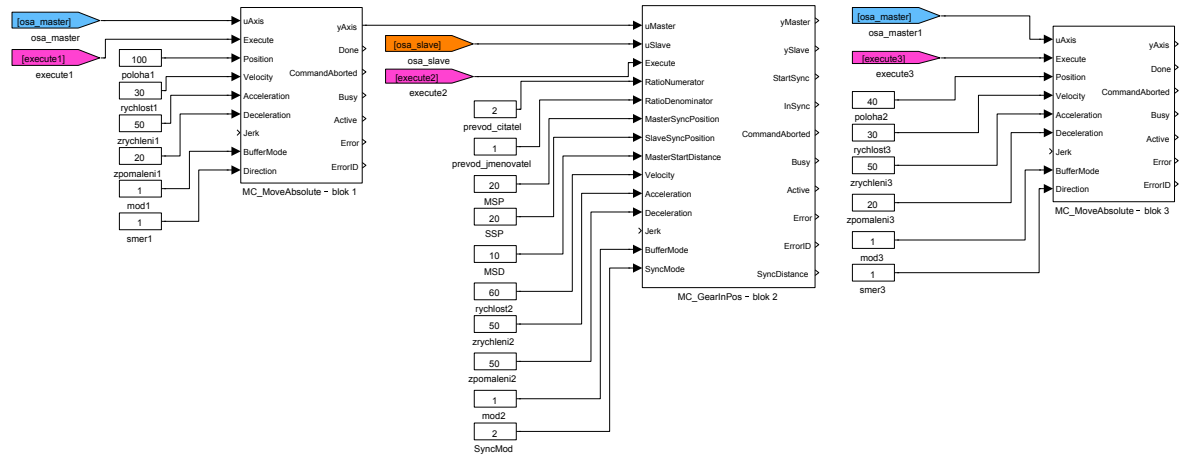


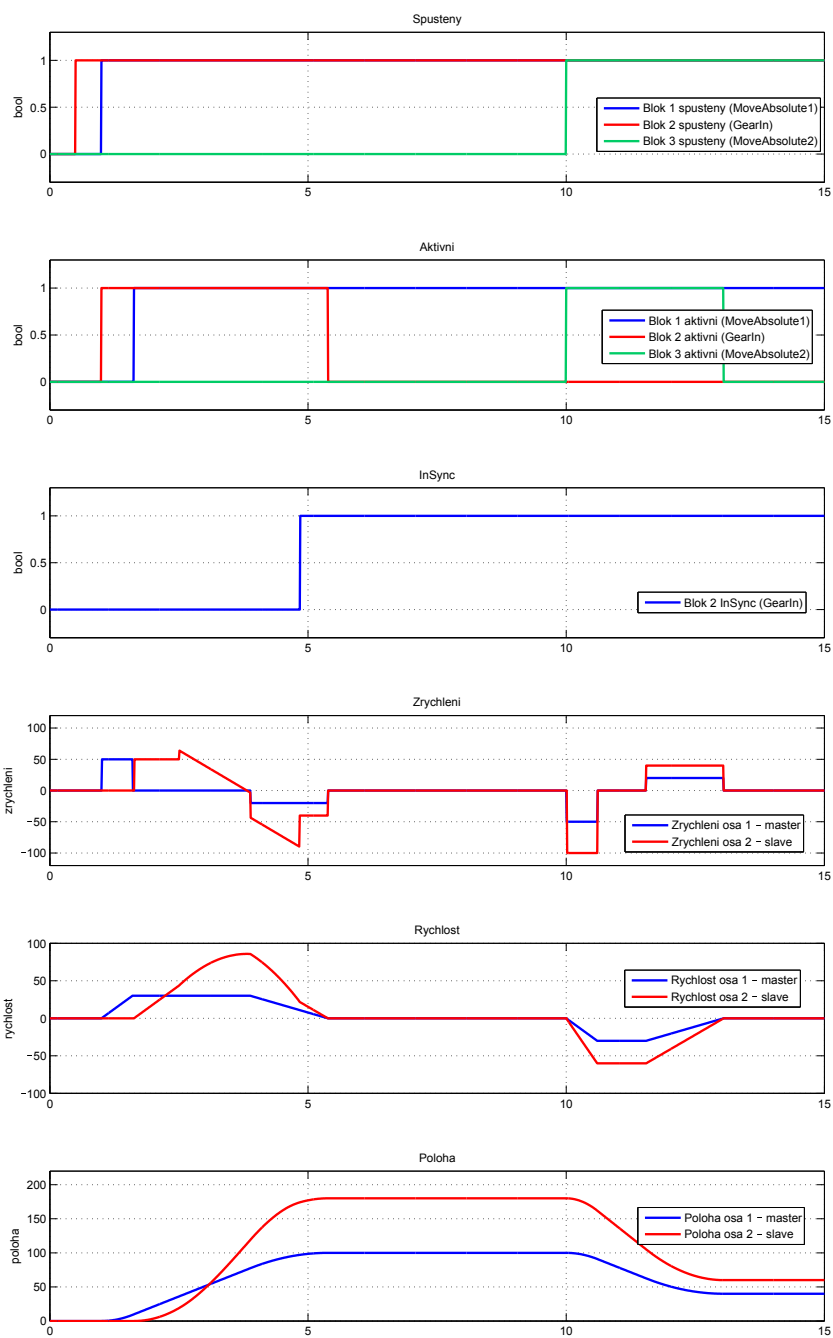
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1 . . . . .	Aborting (nový blok se spustí okamžitě)	
2 . . . . .	Buffered (nový blok se spustí po dokončení předchozího)	
3 . . . . .	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4 . . . . .	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5 . . . . .	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6 . . . . .	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>SyncMode</b>	Režim synchronizace (pouze cyklické osy)	Long (I32)
1 . . . . .	synchronizovat zrychlením	
2 . . . . .	synchronizovat na nejbližší bod	
3 . . . . .	synchronizovat zpomalením	

## Výstupy

<b>yMaster</b>	Odkaz na hlavní osu	Reference
<b>ySlave</b>	Odkaz na podřízenou osu	Reference
<b>StartSync</b>	Začíná synchronizace pohybu	Bool
<b>InSync</b>	Příznak dosažení profilu vačky podřízenou osou	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i . . . . . obecná chyba systému REXYGEN	
<b>SyncDistance</b>	Odchylka v poloze podřízené osy od synchronizované polohy [unit/s]	Double (F64)

## Příklad

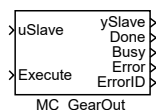




## MC\_GearOut – Vypnutí konstantního převodového poměru

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok `MC_GearOut` ukončuje režim elektronické převodovky zapnutý blokem `MC_GearIn` nebo `MC_GearInPos` nebo `MC_CombineAxis`. Pokud režim elektronické převodovky není aktivní, blok hlásí chybu -703 (Invalid state).

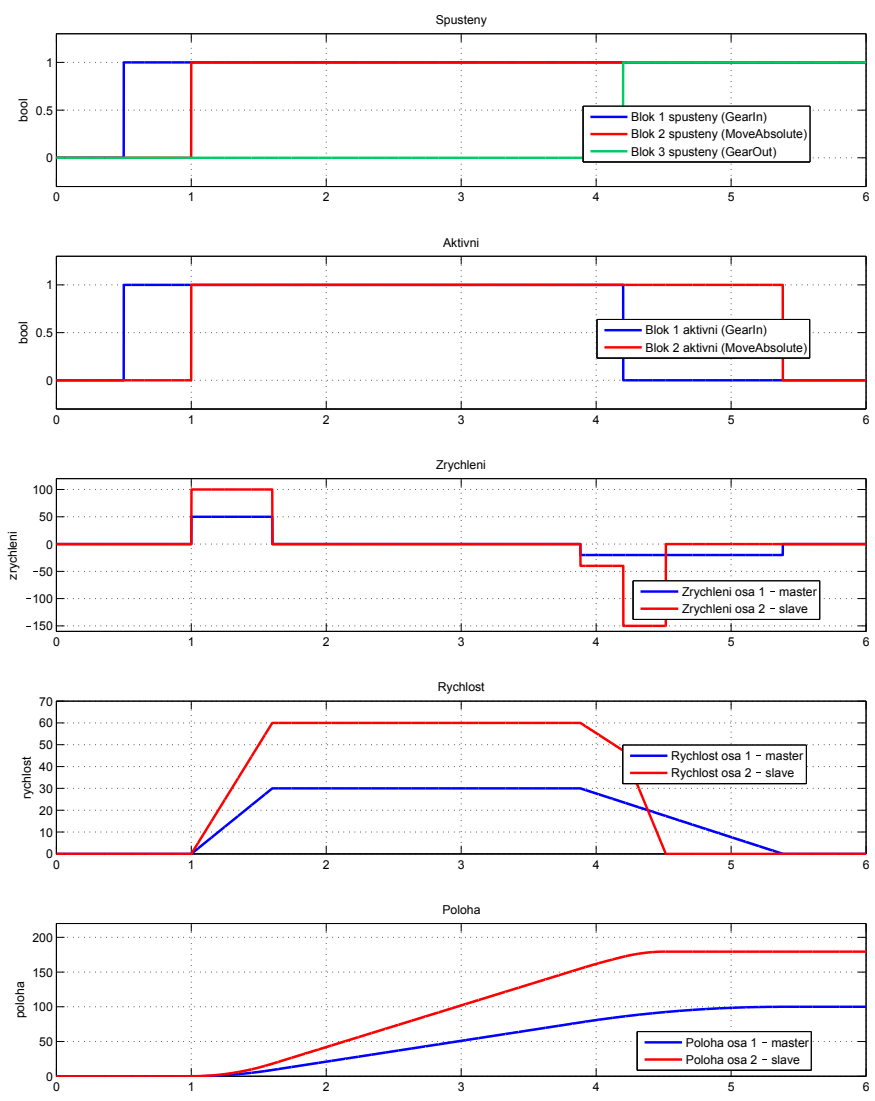
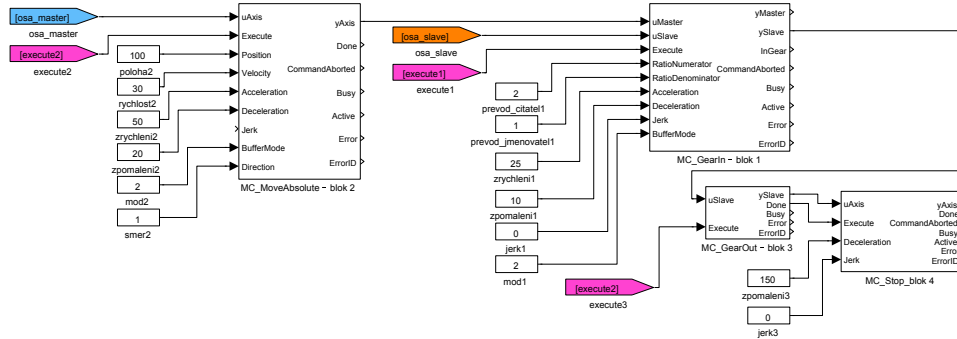
Vstupy

<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

Výstupy

<code>ySlave</code>	Odkaz na podřízenou osu	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

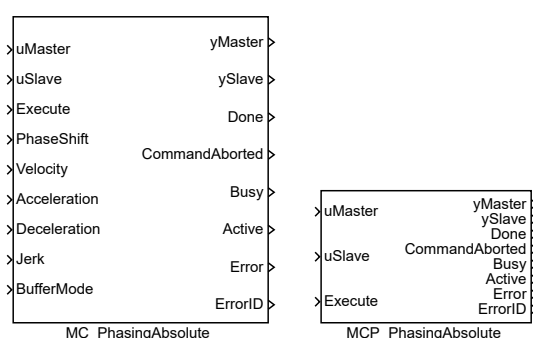
### Příklad



## MC\_PhasingAbsolute, MCP\_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)

Symbole bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_PhasingAbsolute a MCP\_PhasingAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_PhasingAbsolute** zavádí další posunutí na hlavní ose pro vačku (blok **MC\_CamIn**) a převodovku (blok **MC\_GearIn**). Blok funguje velice podobně bloku **MC\_MoveSuperimposed** (tj. generuje pohyb z bodu 0 do bodu **PhaseShift** s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku **MC\_CamIn**, **MC\_GearIn**, **MC\_GearInPos**, **MC\_CombineAxis**.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel **PhaseShift**.

Poznámka 2: Pokud již na ose nějaké fázové posunutí je, tak hodnota **PhaseShift** je výsledné posunutí. Fázové posunutí se nuluje, pokud (slave) osa přejde do stavu disabled nebo je spuštěn další pohyb v režimu aborting.

### Vstupy

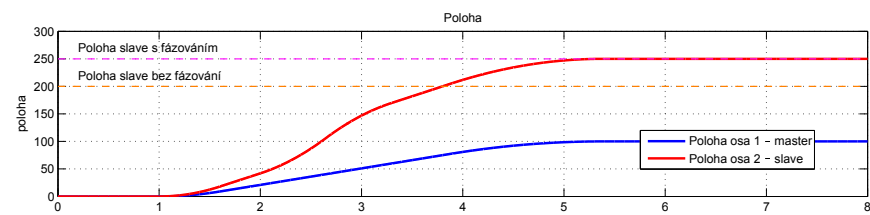
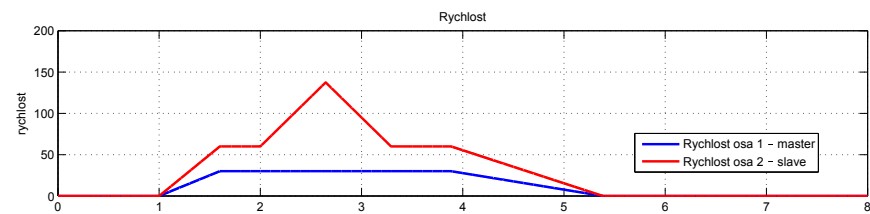
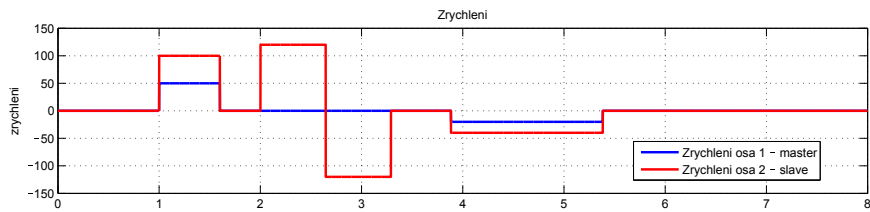
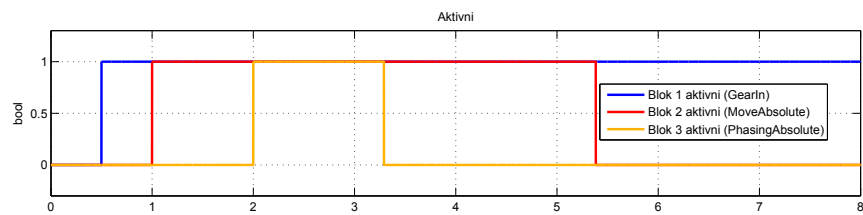
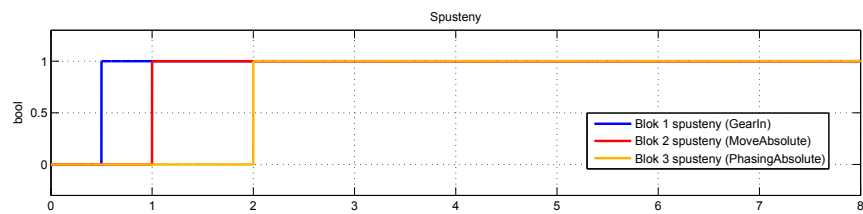
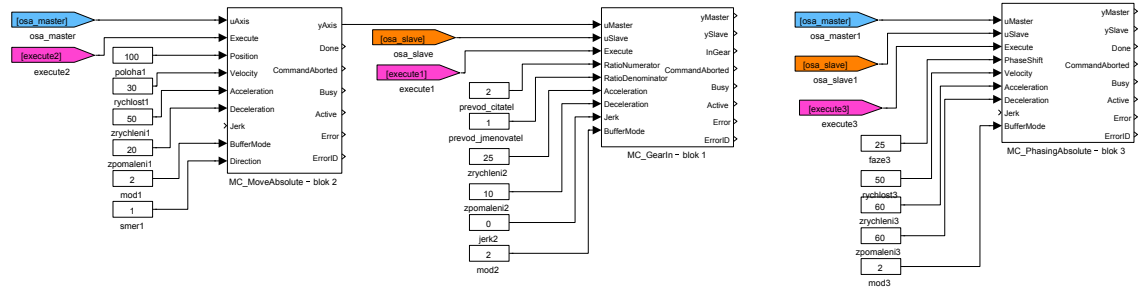
<b>uMaster</b>	Odkaz na hlavní osu	Reference
<b>uSlave</b>	Odkaz na podřízenou osu	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>PhaseShift</b>	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)

<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... aborting	
	2 ..... buffered	

## Výstupy

<b>yMaster</b>	Odkaz na hlavní osu	Reference
<b>ySlave</b>	Odkaz na podřízenou osu	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## Příklad

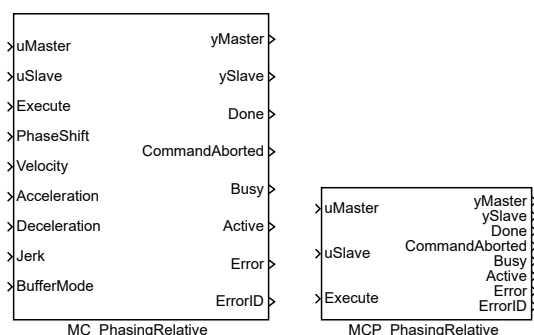




## MC\_PhasingRelative, MCP\_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)

Symbyly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_PhasingRelative a MCP\_PhasingRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_PhasingRelative` zavádí další posunutí na hlavní ose pro vačku (blok `MC_CamIn`) a převodovku (blok `MC_GearIn`). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu `Execute`) přičte hodnota parametru `PhaseShift`. Blok funguje velice podobně bloku `MC_MoveSuperimposed` (tj. generuje pohyb z bodu 0 do bodu `PhaseShift` s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku `MC_CamIn`, `MC_GearIn`, `MC_GearInPos`, `MC_CombineAxis`.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel `PhaseShift`.

Poznámka 2: Pokud již na ose nějaké fázové posunutí je, tak se hodnota `PhaseShift` přičítá k existujícímu posunutí. Fázové posunutí se nuluje, pokud (slave) osa přejde do stavu disabled nebo je spuštěn další pohyb v režimu aborting.

### Vstupy

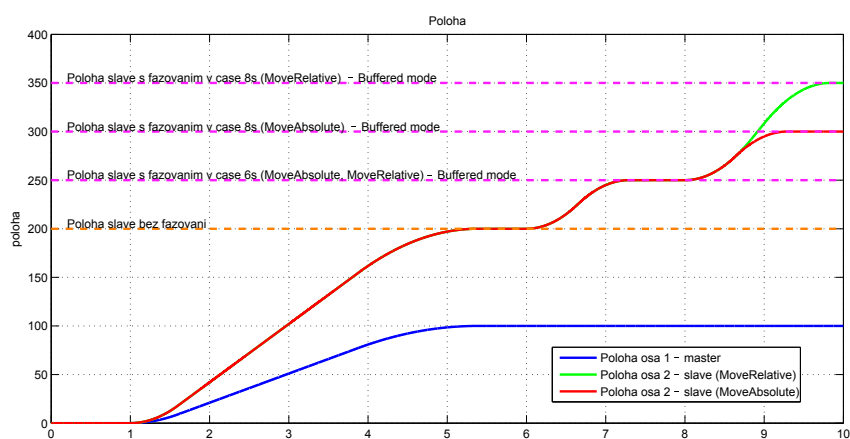
<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>PhaseShift</code>	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 ..... aborting	
	2 ..... buffered	

## Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## Příklad



## Kapitola 22

# MC\_COORD – Koordinované řízení pohybu

### Obsah

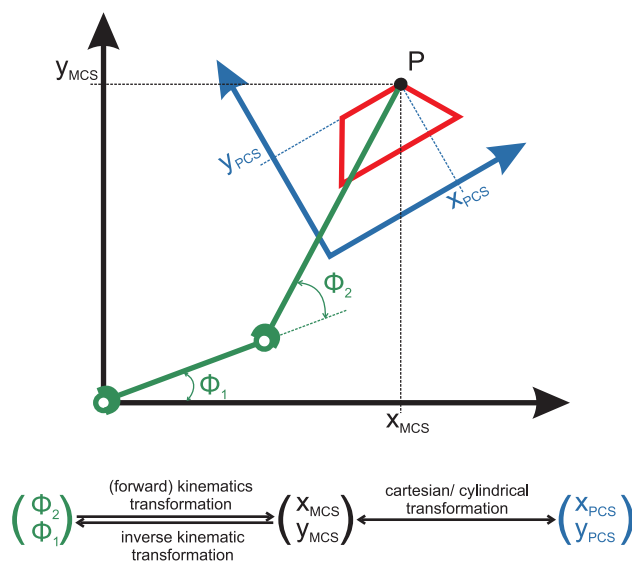
---

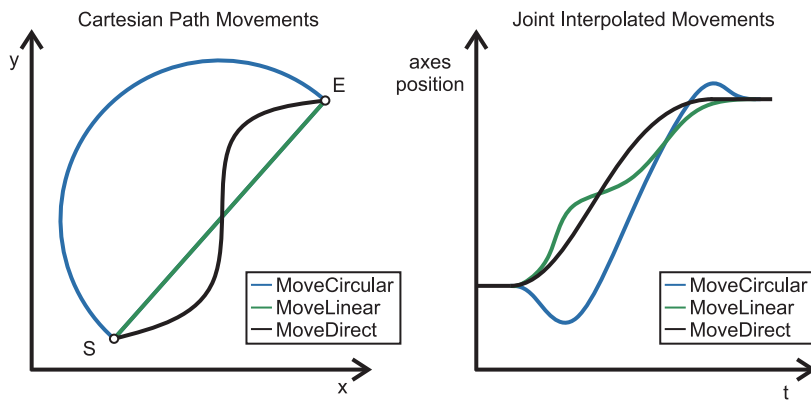
MCP_GroupHalt – * Zastavení koordinovaného pohybu (přerušitelné)	687
MCP_GroupInterrupt – * Přerušení pohybu skupiny os . . . . .	688
MCP_GroupSetOverride – * Nastavení násobivých faktorů na osách ve skupině . . . . .	689
MCP_GroupSetPosition – * Nastavení polohového offsetu skupiny os	690
MCP_GroupStop – * Zastavení koordinovaného pohybu . . . . .	691
MCP_MoveCircularAbsolute – * Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	692
MCP_MoveCircularRelative – * Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	694
MCP_MoveDirectAbsolute – * Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	696
MCP_MoveDirectRelative – * Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	698
MCP_MoveLinearAbsolute – * Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	700
MCP_MoveLinearRelative – * Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	702
MCP_MovePath – * Generování obecné trajektorie v prostoru . . . . .	704
MCP_MovePath_PH – * Generování obecné trajektorie v prostoru PH	706
MCP_SetCartesianTransform – * Kartézské transformace . . . . .	708
MCP_SetKinTransform_Arm – * Kinematické transformace robot ARM	709
MCP_SetKinTransform_UR – * Kinematická transformace pro UR robot . . . . .	711
MC_AddAxisToGroup – Přidání osy do skupiny os . . . . .	713
MC_GroupContinue – Pokračování v přerušném pohybu . . . . .	714

MC_GroupDisable – Převedení skupiny do stavu GroupDisabled . . . . .	715
MC_GroupEnable – Převedení skupiny do stavu GroupStandby . . . . .	716
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné) . . . . .	717
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušování pohybu skupiny os . . . . .	723
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os . . . . .	724
MC_GroupReadActualPosition – Aktuální poloha skupiny os . . . . .	725
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os . . . . .	726
MC_GroupReadError – Chyby ve skupině os . . . . .	727
MC_GroupReadStatus – Stav skupin os . . . . .	728
MC_GroupReset – Nulování chyb os ve skupině . . . . .	730
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině . . . . .	731
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os . . . . .	734
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu . . . . .	736
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	739
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	744
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	749
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	753
MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	757
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	761
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru . . . . .	765
MC_MovePath_PH – * Generování obecné trajektorie v prostoru PH . . . . .	768
MC_ReadCartesianTransform – Přečtení použité kartézské transformace . . . . .	770
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace . . . . .	772
MC_UngroupAllAxes – Odebrání všech os ze skupiny . . . . .	775
RM_AxesGroup – Skupina os pro koordinované řízení pohybu . . . . .	776
RM_Feed – MC „krmič“ . . . . .	781
RM_Gcode – CNC řízení pohybu . . . . .	783
RM_GroupTrack – * Sledování polohy/rychlosti . . . . .	786

---

Knihovna MC\_COORD je speciálně navržena pro koordinaci víceosého řízení pohybu v rámci komplexních systémů. Zahrnuje řadu bloků, včetně `MC_MoveLinearAbsolute` pro provádění přesných lineárních pohybů, doplněných o `MC_MoveLinearRelative` pro relativní lineární pohyb. Pro provádění kruhového pohybu knihovna obsahuje bloky `MC_MoveCircularAbsolute` a `MC_MoveCircularRelative`, což zajišťuje podrobné kruhové trajektorie. V kontextu správy řízení skupinových os tato knihovna zavádí `MC_AddAxisToGroup`, který je dále podporován funkcemi jako `MC_GroupEnable` pro aktivaci, `MC_GroupDisable` pro deaktivaci a `MC_GroupHalt` pro okamžité zastavení seskupených os. Dále knihovna poskytuje `MC_MoveDirectAbsolute` a `MC_MoveDirectRelative`, které umožňují přímou kontrolu nad pohyby os. Pro navigaci po složitých cestách je k dispozici `MC_MovePath`. Základní funkce monitorování a ovládání jsou usnadněny pomocí `MC_GroupReadActualPosition` pro údaje o poloze, `MC_GroupReadActualVelocity` pro přehled rychlosti, `MC_GroupReadError` pro detekci chyb a `MC_GroupReadStatus` pro aktualizace stavu. Knihovna navíc integruje `MC_ReadCartesianTransform` a `MC_SetCartesianTransform`, které jsou životně důležité pro procesy karteziánské transformace. Tato kolekce funkcí podtrhuje význam knihovny v aplikacích, které vyžadují synchronizované řízení více os, zejména v oblastech robotiky a automatizačních systémů.



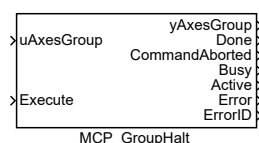


	Aborting	Buffered without Blending	Blending
Trajectory of TCP			
Speed of TCP			

## MCP\_GroupHalt – \* Zastavení koordinovaného pohybu (přerušitelné)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

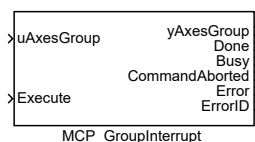
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
<code>LimitMode</code>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 ..... Relativní (k defaultním hodnotám)	
	2 ..... Absolutní [unit/s unit/s <sup>2</sup> ...]	
<code>Superimposed</code>	příznak spuštění jako aditivní pohyb k existujícímu	Bool

### Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_GroupInterrupt – \* Přerušení pohybu skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

## Parametr

<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>LimitMode</code>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 ..... Relativní (k defaultním hodnotám)	
	2 ..... Absolutní [unit/s unit/s <sup>2</sup> ...]	

## Výstup

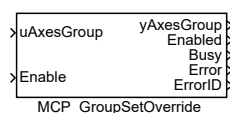
<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	



## MCP\_GroupSetOverride – \* Nastavení násobivých faktorů na osách ve skupině

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>Bool</b>

### Parametr

<b>diff</b>	Pásmo necitlivosti (pro přepočítání trajektorie)	⊙0.05	<b>Double (F64)</b>
<b>VelFactor</b>	Faktor násobení pro rychlost	⊙1.0	<b>Double (F64)</b>
<b>AccFactor</b>	Faktor násobení pro zrychlení	⊙1.0	<b>Double (F64)</b>
<b>JerkFactor</b>	Faktor násobení pro změnu zrychlení	⊙1.0	<b>Double (F64)</b>

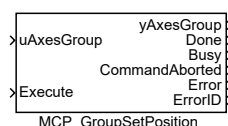
### Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Enabled</b>	Signalizuje úspěšné nastavení násobivých faktorů	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>
<b>ErrorID</b>	Výsledek poslední operace i ..... kód chyby REXYGENu	<b>Error</b>

## MCP\_GroupSetPosition – \* Nastavení polohového offsetu skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

### Parametr

Relative	Výběr absolutních (=false) nebo relativních (=true) souřadnic	Bool
CoordSystem	Volba souřadného systému	⊙3 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	
Position_	Pole souřadnic (pozic a orientací)	Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	

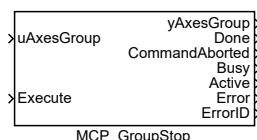
### Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_GroupStop – \* Zastavení koordinovaného pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>Deceleration</code>	Maximální povolené zpomalení [ $\text{unit}/\text{s}^2$ ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [ $\text{unit}/\text{s}^3$ ]	Double (F64)

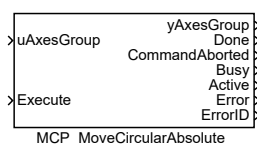
### Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby REXYGENu	Error

## MCP\_MoveCircularAbsolute – \* Pohyb do pozice po kružnicích (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>CircMode</code>	Určuje význam vstupních signálů <code>AuxPoint</code> a <code>CircDirection</code>	Long (I32)
	1 ..... BORDER	⊙1
	2 ..... CENTER	
	3 ..... RADIUS	
<code>PathChoice</code>	Volba směru	⊙1 Long (I32)
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>CoordSystem</code>	Volba souřadného systému	⊙1 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	

<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		
<b>LimitMode</b>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1	Long (I32)
	1 ..... Relativní (k defaultním hodnotám)		
	2 ..... Absolutní [unit/s unit/s^2 ...]		
<b>TransitionMode</b>	Režim míchání pohybu	⊙1	Long (I32)
	1 ..... TMNone		
	2 ..... TMStartVelocity		
	3 ..... TMConstantVelocity		
	4 ..... TMCornerDistance		
	5 ..... TMMaxCornerDeviation		
	11 ..... Smooth		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu		Bool
<b>AuxPoint</b>	Absolutně zadaná pozice	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)
<b>EndPoint</b>	Absolutně zadaná pozice koncového bodu	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

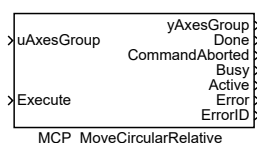
## Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_MoveCircularRelative – \* Pohyb do pozice po kružnicích (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

### Parametr

<b>CircMode</b>	Určuje význam vstupních signálů AuxPoint a CircDirection	Long (I32)
	1 ..... BORDER	⊙1
	2 ..... CENTER	
	3 ..... RADIUS	
<b>PathChoice</b>	Volba směru	⊙1 Long (I32)
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>CoordSystem</b>	Volba souřadného systému	⊙1 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	

<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		
<b>LimitMode</b>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1	Long (I32)
	1 ..... Relativní (k defaultním hodnotám)		
	2 ..... Absolutní [unit/s unit/s^2 ...]		
<b>TransitionMode</b>	Režim míchání pohybu	⊙1	Long (I32)
	1 ..... TMNone		
	2 ..... TMStartVelocity		
	3 ..... TMConstantVelocity		
	4 ..... TMCornerDistance		
	5 ..... TMMaxCornerDeviation		
	11 ..... Smooth		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu		Bool
<b>AuxPoint</b>	Absolutně zadaná pozice	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)
<b>EndPoint</b>	Absolutně zadaná pozice koncového bodu	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

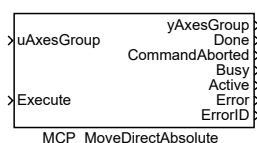
## Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_MoveDirectAbsolute – \* Nekoordinovaný pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

### Parametr

<b>CoordSystem</b>	Volba souřadného systému	⊙1	Long (I32)
1	.... ACS		
2	.... MCS		
3	.... PCS		
4	.... TCS		
<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
1	.... aborting		
2	.... buffered		
3	.... blending low		
4	.... blending high		
5	.... blending previous		
6	.... blending next		
<b>TransitionMode</b>	Režim míchání pohybu	⊙1	Long (I32)
1	.... TMNone		
2	.... TMStartVelocity		
3	.... TMConstantVelocity		
4	.... TMCornerDistance		
5	.... TMMaxCornerDeviation		
11	.... Smooth		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)



<code>Superimposed</code>	příznak spuštění jako aditivní pohyb k existujícímu	<code>Bool</code>
<code>Position_</code>	Pole souřadnic (pozic a orientací) ⊙[0.0 0.0 0.0 0.0 0.0 0.0]	<code>Double (F64)</code>

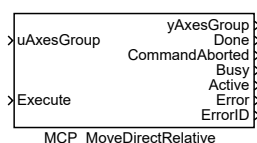
## Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	<code>Reference</code>
<code>Done</code>	Příznak dokončení algoritmu	<code>Bool</code>
<code>CommandAborted</code>	Příznak přerušení funkce bloku	<code>Bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>Bool</code>
<code>Active</code>	Příznak, že blok řídí osu	<code>Bool</code>
<code>Error</code>	Příznak chyby	<code>Bool</code>
<code>ErrorID</code>	Výsledek poslední operace i ..... kód chyby REXYGENu	<code>Error</code>

## MCP\_MoveDirectRelative – \* Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>CoordSystem</code>	Volba souřadného systému	⊙1	Long (I32)
1	.... ACS		
2	.... MCS		
3	.... PCS		
4	.... TCS		
<code>BufferMode</code>	Režim převzetí osy	⊙1	Long (I32)
1	.... aborting		
2	.... buffered		
3	.... blending low		
4	.... blending high		
5	.... blending previous		
6	.... blending next		
<code>TransitionMode</code>	Režim míchání pohybu	⊙1	Long (I32)
1	.... TMNone		
2	.... TMStartVelocity		
3	.... TMConstantVelocity		
4	.... TMCornerDistance		
5	.... TMMaxCornerDeviation		
11	.... Smooth		
<code>TransitionParameter</code>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)

<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu	Bool
<b>Distance</b>	Pole souřadnic (relativních pozic a orientací) ⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

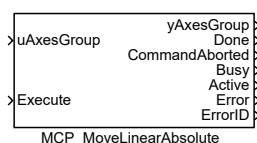
## Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace i . . . . . kód chyby REXYGENu	Error

## MCP\_MoveLinearAbsolute – \* Pohyb do pozice po přímkách (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>CoordSystem</code>	Volba souřadného systému	⊙1 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<code>LimitMode</code>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 ..... Relativní (k defaultním hodnotám)	
	2 ..... Absolutní [unit/s unit/s <sup>2</sup> ...]	

<b>TransitionMode</b>	Režim míchání pohybu	⊙1	Long (I32)
	1 ..... TMNone		
	2 ..... TMStartVelocity		
	3 ..... TMConstantVelocity		
	4 ..... TMCornerDistance		
	5 ..... TMMaxCornerDeviation		
	11 ..... Smooth		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu		Bool
<b>Position_</b>	Pole souřadnic (pozic a orientací)		Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]		

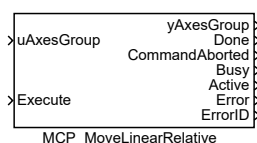
## Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_MoveLinearRelative – \* Pohyb do pozice po přímkách (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<code>CoordSystem</code>	Volba souřadného systému	⊙1 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	
<code>BufferMode</code>	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<code>LimitMode</code>	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 ..... Relativní (k defaultním hodnotám)	
	2 ..... Absolutní [unit/s unit/s <sup>2</sup> ...]	

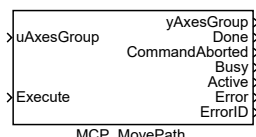
<b>TransitionMode</b>	Režim míchání pohybu	⊙1	Long (I32)
	1 ..... TMNone		
	2 ..... TMStartVelocity		
	3 ..... TMConstantVelocity		
	4 ..... TMCornerDistance		
	5 ..... TMMaxCornerDeviation		
	11 ..... Smooth		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu		Bool
<b>Distance</b>	Pole souřadnic (relativních pozic a orientací)		Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]		

## Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

MCP\_MovePath – \* **Generování obecné trajektorie v prostoru**

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

## Parametr

TotalTime	Čas celého pohybu	Double (F64)
RampTime	Čas [s] zrychlování/zpomalování	Double (F64)
CoordSystem	Volba souřadného systému	⊙2 Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
TransitionMode	Režim míchání pohybu	⊙1 Long (I32)
	1 ..... TMNone	
	2 ..... TMStartVelocity	
	3 ..... TMConstant Velocity	
	4 ..... TMCornerDistance	
	5 ..... TMMaxCornerDeviation	
	11 .... Smooth	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)



RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool
pc	Control-points matrix ⊙[0.0 1.0 2.0; 0.0 1.0 1.0; 0.0 1.0 0.0]	Double (F64)
pk	Knot-points vector ⊙[0.0 0.0 0.0 0.0 0.5 1.0 1.0]	Double (F64)
pw	Váhový vektor ⊙[1.0 1.0 1.0]	Double (F64)
pv	Polynoms for feedrate definition ⊙[0.0 0.05 0.95; 0.0 0.1 0.1; 0.0 0.0 0.0; 0.1 0.0 -0.1; -0.05 0.0 0.05; 0.0 0.0 0.0]	Double (F64)
pt	Knot-points (time [s]) for feedrate ⊙[0.0 1.0 10.0 11.0]	Double (F64)
user	Pouze pro special edit ⊙[0.0 1.0 2.0 3.0]	Double (F64)

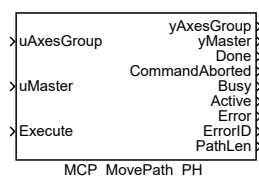
## Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... kód chyby REXYGENu	Error

## MCP\_MovePath\_PH – \* Generování obecné trajektorie v prostoru PH

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	Reference
<b>uMaster</b>	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool

### Parametr

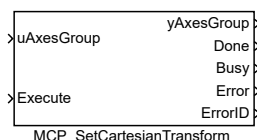
<b>CoordSystem</b>	Volba souřadného systému	⊙2	Long (I32)
	1 ..... ACS		
	2 ..... MCS		
	3 ..... PCS		
	4 ..... TCS		
<b>BufferMode</b>	Režim převzetí osy	⊙1	Long (I32)
	1 ..... aborting		
	2 ..... buffered		
	3 ..... blending low		
	4 ..... blending high		
	5 ..... blending previous		
	6 ..... blending next		
<b>Cyclic</b>	Příznak cyklicky opakovaného profilu		Bool
<b>RampIn</b>	RampIn faktor (0 = RampIn režim se nepoužívá)		Double (F64)
<b>Superimposed</b>	příznak spuštění jako aditivní pohyb k existujícímu		Bool
<b>UZV1</b>		⊙[]	Double (F64)
<b>Points</b>		⊙[]	Double (F64)

## Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yMaster</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
<code>PathLen</code>		Double (F64)

MCP\_SetCartesianTransform – \* **Kartézské transformace**

Symbol bloku

Licence: **COORDINATED MOTION**

## Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

## Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

## Parametr

<code>SelTrans</code>	Dotčená transformace souřadnic	⊙1	Long (I32)
	1 ..... Základní souřadnice výrobku (PCS)		
	2 ..... Posunutí nástroje		
	3 ..... Souřadnice základny stroje		
<code>TransX</code>	Posun v ose X		Double (F64)
<code>TransY</code>	Posun v ose Y		Double (F64)
<code>TransZ</code>	Posun v ose Z		Double (F64)
<code>RotAngle1</code>	Úhel rotace podél osy Z		Double (F64)
<code>RotAngle2</code>	Úhel rotace podél osy Y		Double (F64)
<code>RotAngle3</code>	Úhel rotace podél osy X		Double (F64)
<code>Relative</code>	Výběr absolutních (=false) nebo relativních (=true) souřadnic		Bool
<code>SSF</code>	xyz		Bool

## Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	

## MCP\_SetKinTransform\_Arm – \* Kinematické transformace robot ARM

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

### Parametr

<code>arot</code>	Úhel pro jednu otáčku (ACS)	⊖6.28318531	Double (F64)
<code>mrot</code>	Úhel pro jednu otáčku (MCS)	⊖6.28318531	Double (F64)
<code>irt</code>	Způsob zadávání otočení	⊖1	Long (I32)
	1 ..... ZYX úhly		
	2 ..... Kvaternion		
<code>a1</code>		⊖400.0	Double (F64)
<code>a2</code>		⊖300.0	Double (F64)
<code>a3</code>			Double (F64)
<code>d1</code>			Double (F64)
<code>d23</code>			Double (F64)
<code>d4</code>		⊖200.0	Double (F64)
<code>d6</code>		⊖100.0	Double (F64)
<code>xe</code>			Double (F64)
<code>ye</code>			Double (F64)
<code>ze</code>			Double (F64)
<code>gamaE</code>			Double (F64)
<code>betaE</code>			Double (F64)
<code>alphaE</code>			Double (F64)

## Výstup

		Reference
yAxesGroup	Odkaz na skupinu os	
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... kód chyby REXYGENu	Error

## MCP\_SetKinTransform\_UR – \* Kinematická transformace pro UR robot

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>Bool</b>

### Parametr

<b>arot</b>	Úhel pro jednu otáčku (ACS)	⊙6.28318531	Double (F64)
<b>mrot</b>	Úhel pro jednu otáčku (MCS)	⊙6.28318531	Double (F64)
<b>irt</b>	Způsob zadávání otočení	⊙1	Long (I32)
	1 ..... ZYX úhly		
	2 ..... Kvaternion		
11			Double (F64)
12			Double (F64)
13			Double (F64)
14		⊙400.0	Double (F64)
15		⊙400.0	Double (F64)
16		⊙400.0	Double (F64)
17		⊙400.0	Double (F64)
18		⊙400.0	Double (F64)

### Výstup

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>Reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>Bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>Bool</b>
<b>Error</b>	Příznak chyby	<b>Bool</b>

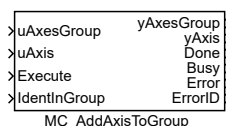
ErrorID	Výsledek poslední operace	Error
	i ..... kód chyby REXYGENu	



## MC\_AddAxisToGroup – Přidání osy do skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok `MC_AddAxisToGroup` přidá do skupiny os `uAxesGroup` osu `uAxis`. Skupina os `uAxesGroup` se založí pomocí bloku [RM\\_AxesGroup](#). Osa `uAxis` přiváděná na vstup bloku `MC_AddAxisToGroup` musí být definována stejně jako jednotlivá osa [RM\\_Axis](#) z knihovny `MC_SINGLE`.

Poznámka 1: Každý `IdentInGroup` může být použit pouze jednou, jinak nastane chyba.

### Vstupy

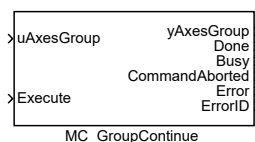
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>IdentInGroup</code>	Pořadí osy ve skupině (0=první nepřřazená)	Long (I32)

### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_GroupContinue – Pokračování v přerušném pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Blok `MC_GroupContinue` ruší působení bloku `MC_GroupInterrupt`. Skupina os dokončí původně prováděný pohyb před přerušáním. Výstup `Done` určuje, zda se pohyb vrátil do stavu před přerušáním.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

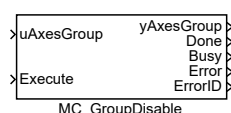
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>CommandAborted</code>	Příznak přerušování funkce bloku	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## MC\_GroupDisable – Převedení skupiny do stavu GroupDisabled

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok `MC_GroupDisable` mění stav skupiny os na stav „GroupDisabled“. Pokud je skupina os v pohybu, tak je nejprve převedena do stavu „Stopping“. Zastavuje se s maximální možnou hodnotou zpomalení (nastavenou v bloku [RM\\_Axis](#)). Po zastavení je skupina os převedena do stavu „GroupDisabled“ (viz. stavový diagram u bloku [RM\\_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

Vstupy

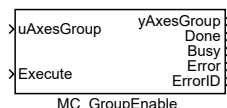
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_GroupEnable – Převedení skupiny do stavu GroupStandby

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Blok `MC_GroupEnable` převede stav skupiny os ze stavu „GroupDisabled“ do stavu „GroupStandby“ (viz. stavový diagram u bloku [RM\\_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

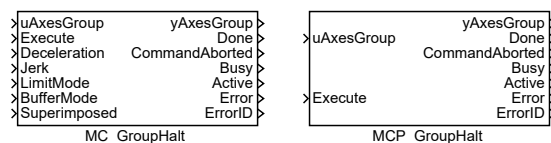
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_GroupHalt, MCP\_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_GroupHalt a MCP\_GroupHalt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GroupHalt zahajuje řízené zastavení pohybu. Osa se přesune do stavu „GroupMoving“, dokud není rychlost nulová. Společně s nastavením výstupu Done je stav změněn na „GroupStandby“.

Poznámka 1: Blok MC\_GroupHalt se používá k zastavení skupiny os za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší MC\_GroupHalt a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. MC\_GroupHalt je aktivován. Před dosažením stavu „GroupStandby“ je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

Poznámka 3: Blok nemá parametr CoordSystem, protože jej přejímá z předchozího bloku.

Vstupy

		Reference
uAxesGroup	Odkaz na skupinu os	Bool
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
LimitMode	Volba jednotek pro limity (Deceleration, Jerk)	⊙1 Long (I32)
	1 . . . . . Relative [part of group limit]	
	2 . . . . . Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	

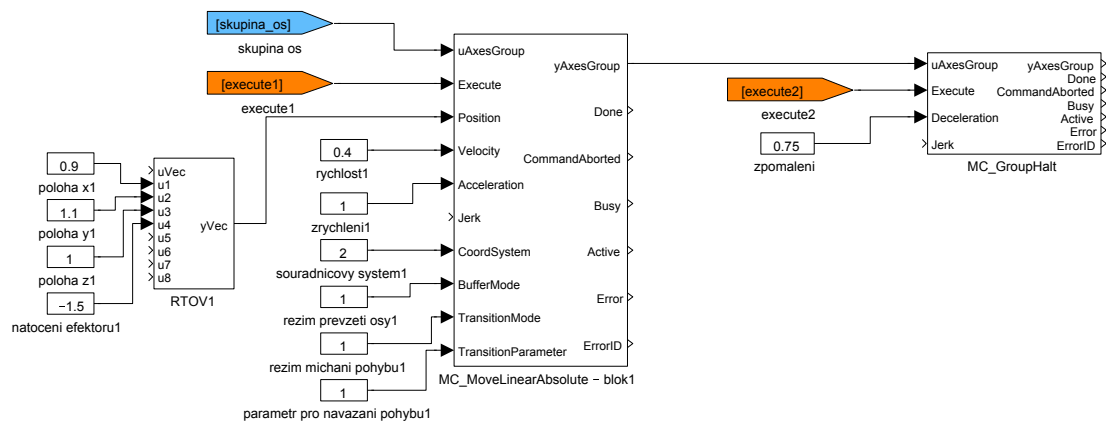
<b>BufferMode</b>	Režim převzetí osy	☉1	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)		
2	..... Buffered (nový blok se spustí po dokončení předchozího)		
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)		
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)		
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)		
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb		Bool

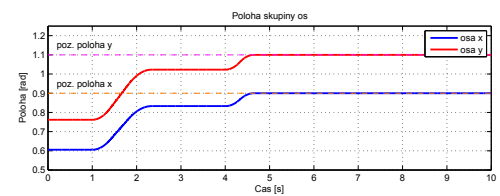
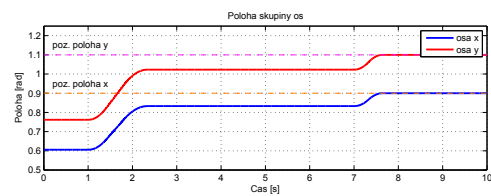
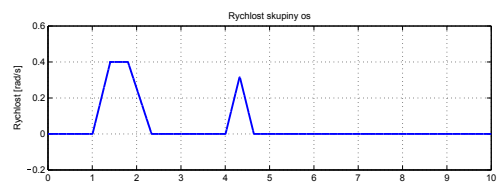
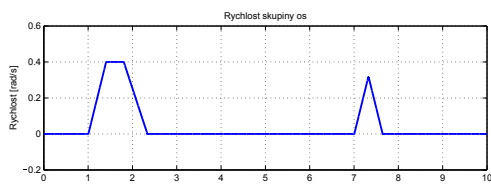
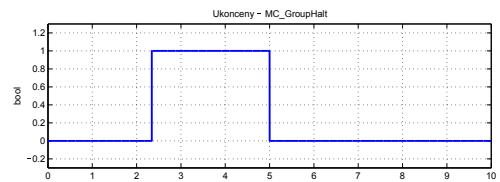
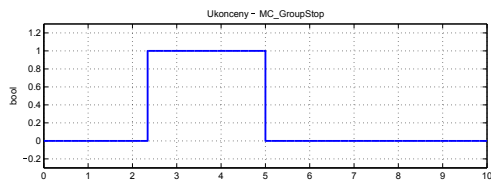
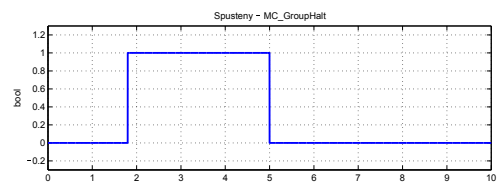
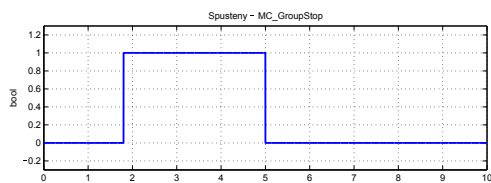
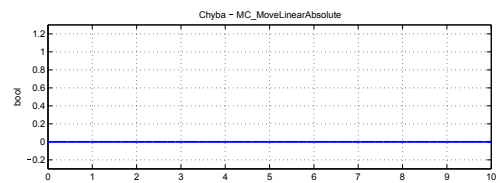
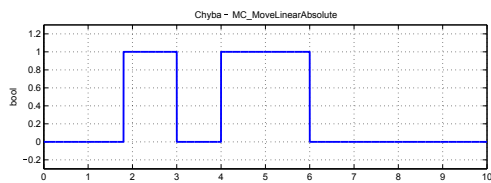
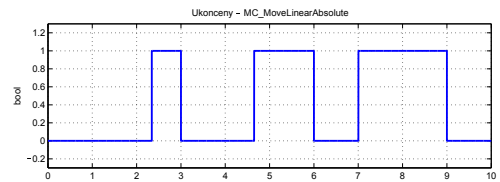
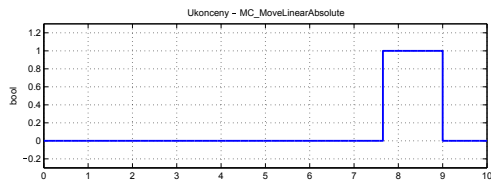
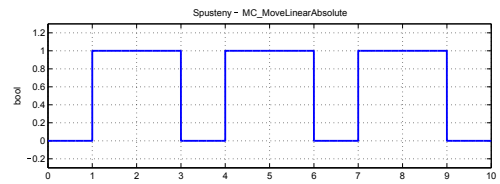
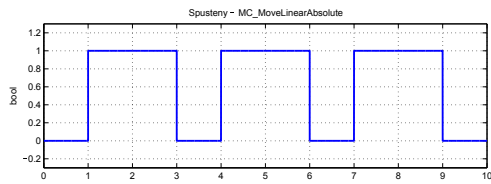
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i	..... obecná chyba systému REXYGEN

## Příklad

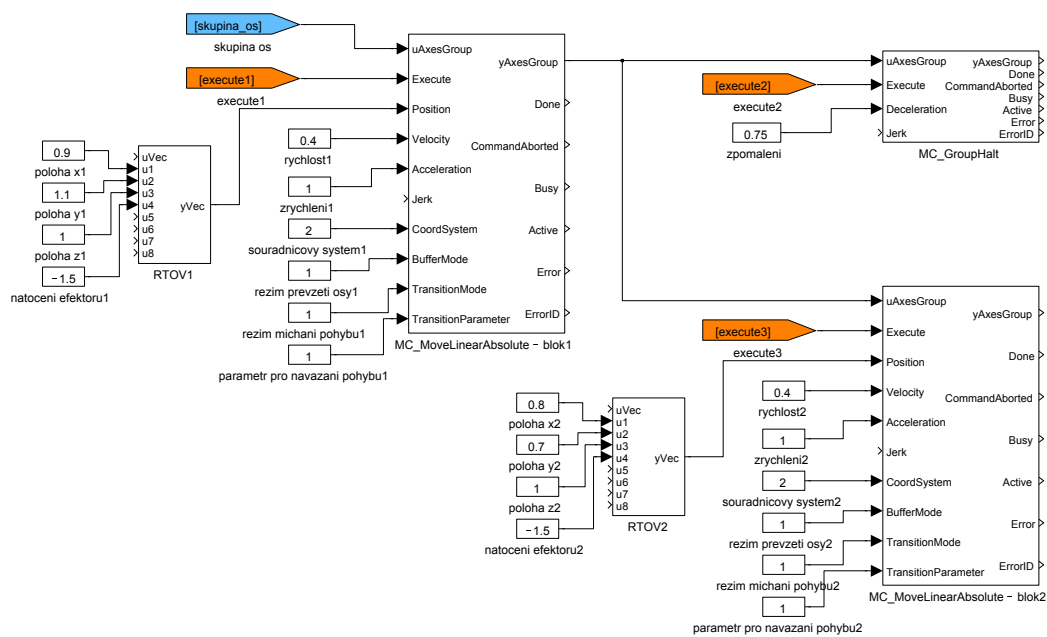
Pro porovnání je následující příklad totožný jako u bloku [MC\\_GroupStop](#). Časové průběhy v levém sloupci odpovídají bloku [MC\\_GroupStop](#). Průběhy v pravém sloupci bloku [MC\\_GroupHalt](#). Je vidět, že při použití bloku [MC\\_GroupHalt](#) dojde k dojetí do požadované polohy již na druhé spuštění exekutivy bloku [MC\\_MoveLinearAbsolute](#).

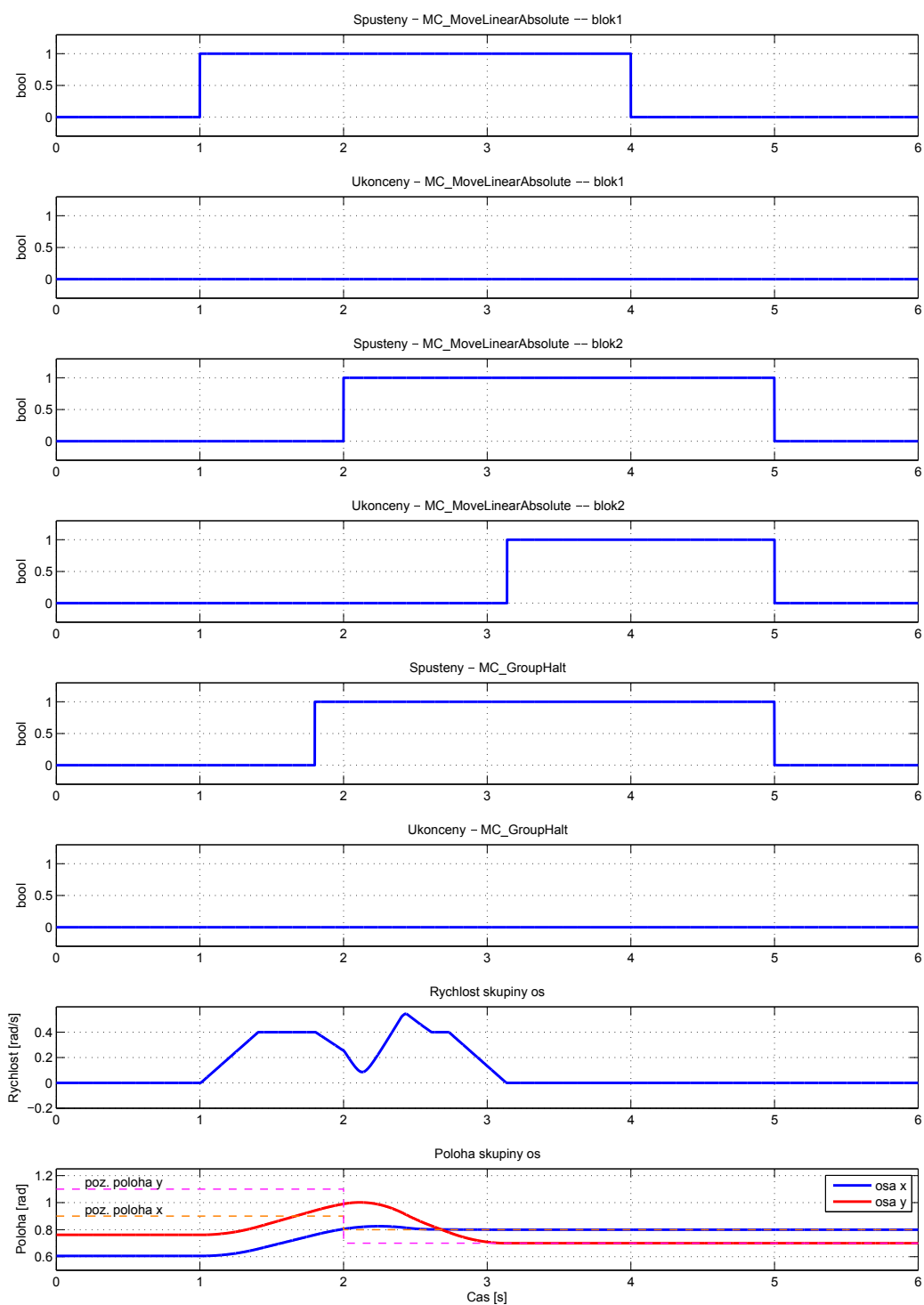






V druhém příkladě je spuštěn blok **MC\_MoveLinearAbsolute**, který je následně přerušen blokem **MC\_GroupHalt**. Ještě před zastavením je spuštěn druhý pohyb blokem dalším **MC\_MoveLinearAbsolute**. Příklad ilustruje to, že pro možnost spuštění dalšího pohybu, nemusí dojít k úplnému zastavení skupiny os.

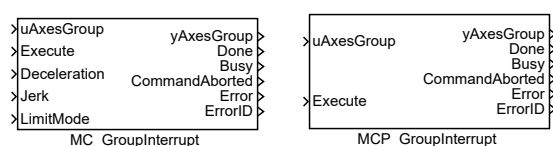




## MC\_GroupInterrupt, MCP\_GroupInterrupt – Přerušení pohybu skupiny os

Symboly bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_GroupInterrupt a MCP\_GroupInterrupt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GroupInterrupt** přeruší právě prováděný pohyb a uvede skupinu os do klidu, nicméně tento příkaz nezruší přerušený pohyb (výstup **CommandAborted** nebude nastaven na true a **Busy** zůstane true). Informace o původním pohybu zůstane uložena. Skupina os zůstane v původním stavu i po zastavení, kdy je výstup **Done** nastaven na true.

Poznámka 1: Tento blok je spárován s blokem **MC\_GroupContinue**, jehož aktivací je skupina os vrácena do stavu, před spuštěním bloku **MC\_GroupInterrupt**.

Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>LimitMode</b>	Volba jednotek pro limity (Deceleration, Jerk)	⊙1 Long (I32)
	1 ..... Relative [part of group limit]	
	2 ..... Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	

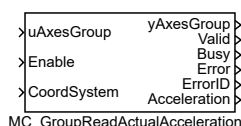
Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## MC\_GroupReadActualAcceleration – Aktuální zrychlení skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok `MC_GroupReadActualAcceleration` zpřístupňuje na výstup `Position` aktuální zrychlení skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` `true`, čehož se dosáhne nastavením vstupu `Enable` na hodnotu `true`.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů `ZYX`, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány za sekundu za sekundu).

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

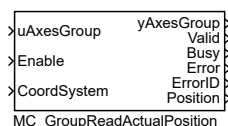
### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
<code>Acceleration</code>	Pole aktuálních zrychlení skupiny os	Reference

## MC\_GroupReadActualPosition – Aktuální poloha skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok `MC_GroupReadActualPosition` zpřístupňuje na výstup `Position` aktuální polohu skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` `true`, čehož se dosáhne nastavením vstupu `Enable` na hodnotu `true`.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů ZYX, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány).

### Vstupy

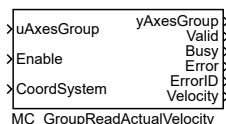
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
<code>Position</code>	Pole aktuálních souřadnic (pozice a orientace) skupiny os	Reference

## MC\_GroupReadActualVelocity – Aktuální rychlost skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Blok `MC_GroupReadActualVelocity` zpřístupňuje na výstup `Velocity` aktuální rychlost skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` `true`, čehož se dosáhne nastavením vstupu `Enable` na hodnotu `true`.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů ZYX, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány za sekundu).

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

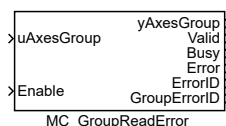
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
<code>Velocity</code>	Pole aktuálních rychlostí skupiny os	Reference

## MC\_GroupReadError – Chyby ve skupině os

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok `MC_GroupReadError` zpřístupňuje na výstupu `GroupErrorID` aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

Poznámka 1: Tento blok je implementován z důvodu kompatibility s PLCopen neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu `ErrorID` bloku `RM_AxesGroup`.

### Vstupy

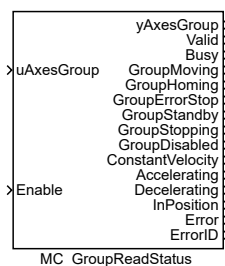
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error
<code>GroupErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## MC\_GroupReadStatus – Stav skupin os

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Blok `MC_GroupReadStatus` indikuje na svých výstupech různé stavy připojené skupiny os jako logickou hodnotu. Indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je na výstupu `Valid true`, čehož se dosáhne nastavením vstupu `Enable` na hodnotu `true`.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

## Výstupy

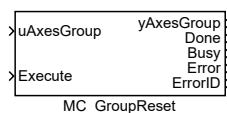
<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>GroupMoving</code>	Stav GroupMoving	Bool
<code>GroupHoming</code>	Stav GroupHoming	Bool
<code>GroupErrorStop</code>	Stav ErrorStop	Bool
<code>GroupStandby</code>	Stav Standby	Bool
<code>GroupStopping</code>	Stav Stopping	Bool
<code>GroupDisabled</code>	Stav Disabled	Bool
<code>ConstantVelocity</code>	Pohyb konstantní rychlostí	Bool
<code>Accelerating</code>	Zrychlování	Bool
<code>Decelerating</code>	Zpomolování	Bool
<code>InPosition</code>	Příznak dosažení zadané pozice	Bool
<code>Error</code>	Příznak chyby	Bool



ErrorID	Výsledek poslední operace	Error
i	..... obecná chyba systému REXYGEN	

## MC\_GroupReset – Nulování chyb os ve skupině

Symbol bloku

Licence: [COORDINATED MOTION](#)

## Popis funkce

Blok `MC_GroupReset` převede připojenou skupinu os ze stavu „GroupErrorStop“ do stavu „GroupStandBy“ a vynuluje ve skupině všechny příznaky chyby. Blok resetuje také všechny osy v dané skupině os stejně jako blok `MC_Reset` z knihovny `MC_SINGLE`.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

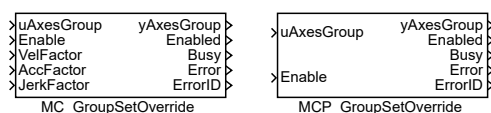
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## MC\_GroupSetOverride, MCP\_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_GroupSetOverride a MCP\_GroupSetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_GroupSetOverride` nastavuje násobivé faktory, které se projeví ve všech blocích pracujících se skupinou os. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je nutné vynásobit faktorem z tohoto bloku, tím dostaneme hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadaných v `RM_Axis` a administrativních bloků. Tento blok není aktivován hranou, ale pokud je na vstupu `Enable` true, tak se hodnoty trvale aktualizují. Pokud je aktivní například blok typu `MC_MoveLinearAbsolute`, vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr `diff`) a přepočet trajektorie je proveden, až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka 1: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu `errorID = -700` - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu `errorID = -701` - hodnota mimo rozsah).

Poznámka 2: Je dovoleno zadat `VelFactor = 0`. Vede to k dočasnému zastavení aktuálního pohybu (podobně jako `MC_GroupInterrupt`). Pro dokončení pohybu se pak ale musí nastavit `VelFactor` kladný.

### Vstupy

		Reference
<code>uAxesGroup</code>	Odkaz na skupinu os	<code>Bool</code>
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	<code>Bool</code>
<code>VelFactor</code>	Faktor násobení pro rychlost	<code>Double (F64)</code>
<code>AccFactor</code>	Faktor násobení pro zrychlení	<code>Double (F64)</code>
<code>JerkFactor</code>	Faktor násobení pro změnu zrychlení	<code>Double (F64)</code>

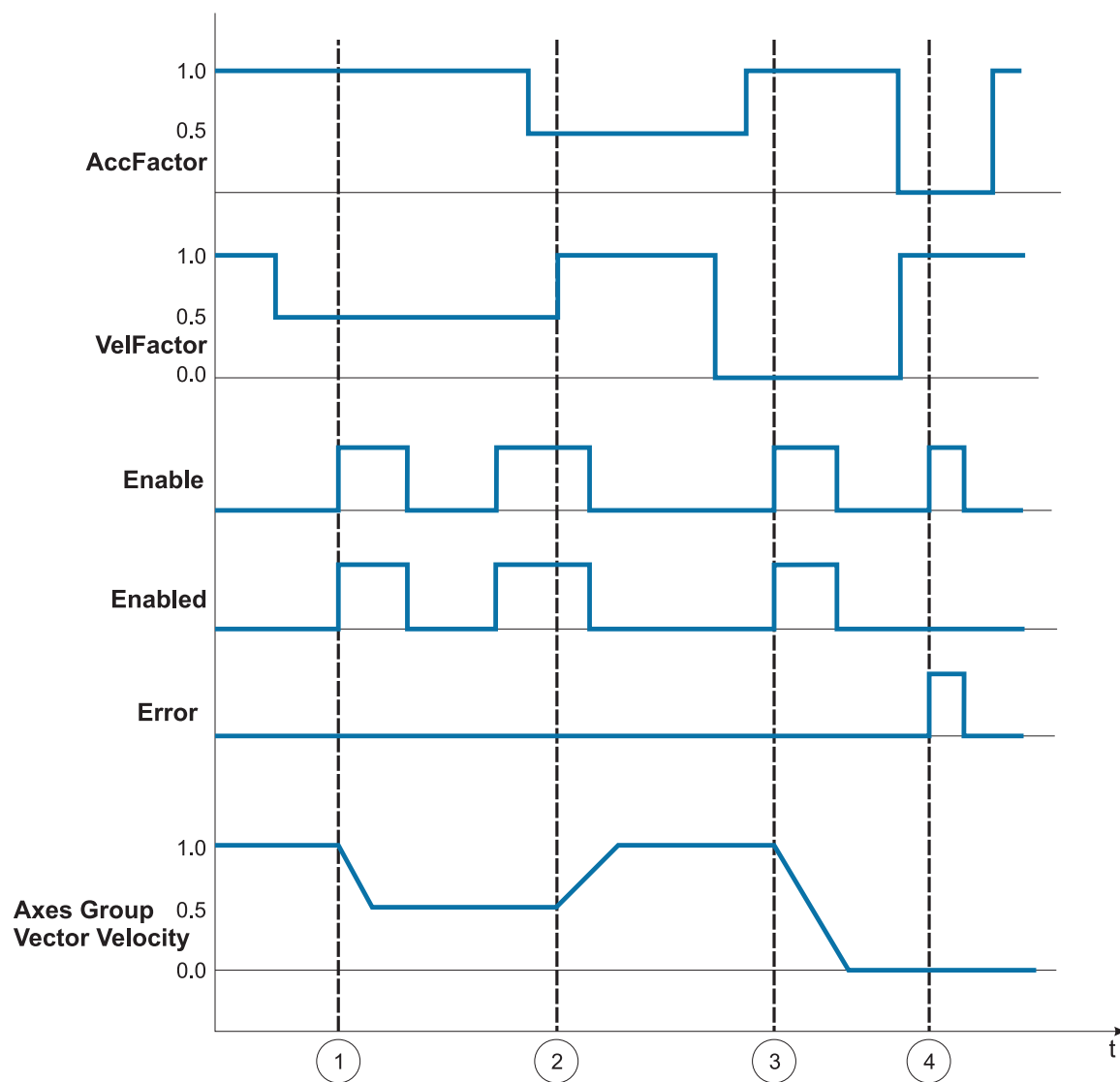
## Parametr

diff	Pásmo necitlivosti (pro přepočítání trajektorie)	⊙0.05	Double (F64)
------	--	-------	--------------

## Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Enabled	Signalizuje úspěšné nastavení násobivých faktorů	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## Příklad

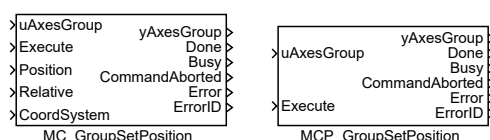


- ① Rychlost skupiny os zmenšena na 50% se 100% zpomalením
- ② Rychlost skupiny os vrácena na 100% s 50% akcelerací
- ③ Rychlost skupiny os zmenšena na 0% se 100% zpomalením
- ④ Žádná změna, protože není povolen AccFactor = 0.0, je sepnuta chyba

## MC\_GroupSetPosition, MCP\_GroupSetPosition – Nastavení polohového offsetu skupiny os

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_GroupSetPosition a MCP\_GroupSetPosition mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_GroupSetPosition` nastaví polohu všech os ve skupině bez jejich pohybu. Nové souřadnice jsou dány vstupem `Position`. Pomocí vstupu `CoordSystem` se nastaví, v jakém souřadnicovém systému se provede změna. Tato změna následně ovlivní i souřadnice ve vyšších souřadnicových systémech.

**POZOR:** blok momentálně není implementován, protože není jasný usecase a jak tedy polohy dosáhnout v ACS by mohlo dávat smysl home na osách (už řeší `RM_HomeOffset`), v MCS buď home na osách nebo posunout počátek robota nebo počátek nástroje (tj. řeší `MC_SetCartesianTransform` base offset nebo tool offset), v PCS buď předchozí možnosti nebo změnit transformaci MCS na PCS (řeší `MC_SetCartesianTransform`), v TCS nedává smysl

Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Position</code>	Pole souřadnic (pozic a orientací)	Reference
<code>Relative</code>	Výběr absolutních (=false) nebo relativních (=true) souřadnic	Bool
	off ... absolute	
	on .... relative	
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

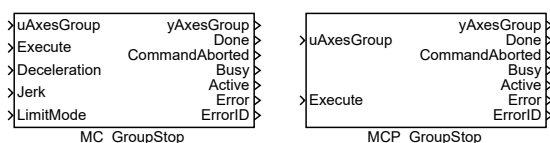
## Výstupy

		Reference
yAxesGroup	Odkaz na skupinu os	
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i . . . . . obecná chyba systému REXYGEN	

## MC\_GroupStop, MCP\_GroupStop – Zastavení koordinovaného pohybu

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_GroupStop a MCP\_GroupStop mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GroupStop zastaví pohyb. Režim je vždy aborted, tj. zastavování se spouští okamžitě. Blok se nejprve pokusí zastavit ve směru původní trajektorie. Pokud se to nepodaří, zastaví se pomocí errorstop sekvence na jednotlivých osách již nekoordinovaně. Dokud je na vstup Execute hodnota true nebo dokud se skupina pohybuje, nachází se skupina ve stavu „Stopping“ a není možné spouštět další bloky. Okamžitě po zastavení se nastaví na výstupu Done hodnota true (pokud nenastane chyba). Skupina os přejde do „Standby“ až po deaktivování vstupu Execute.

Poznámka 1: Blok nemá parametr CoordSystem, protože jej přejímá z právě běžícího bloku.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

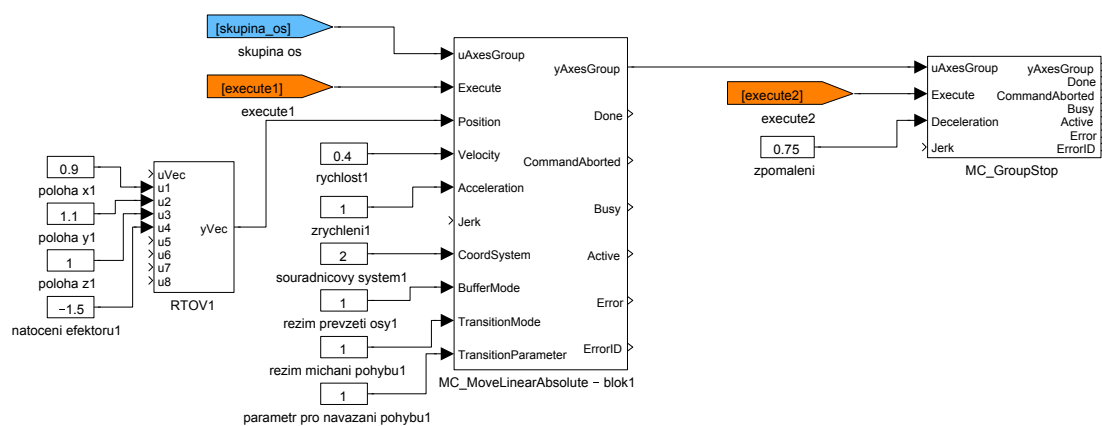


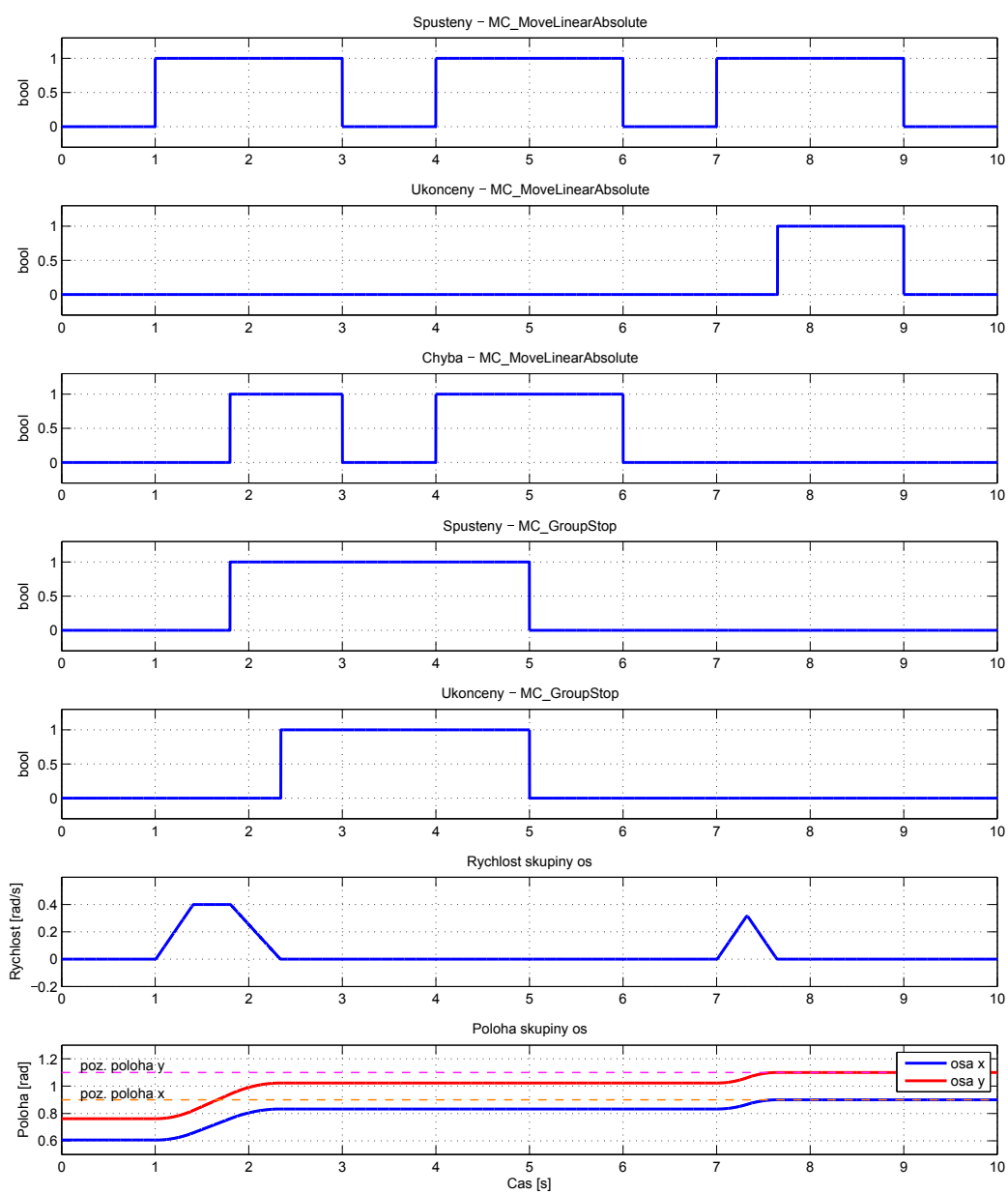
ErrorID Výsledek poslední operace  
i ..... obecná chyba systému REXYGEN

Error

## Příklad

Na časovém diagramu níže, je uveden příklad na chování bloku MC\_GroupStop. Nejprve dojde ke spuštění bloku MC\_MoveLinearAbsolute a tím dojde k uvedení osy do pohybu. Následně je již spuštěn blok MC\_GroupStop, který způsobí zastavení skupiny os. Následně druhé spuštění exekutivy bloku MC\_MoveLinearAbsolute nic nezpůsobí a to i přes to, že je skupina již v klidu. To z toho důvodu, že je stále spuštěný blok MC\_GroupStop. Až třetí spuštění exekutivy bloku MC\_MoveLinearAbsolute dostane skupinu os do požadované polohy.

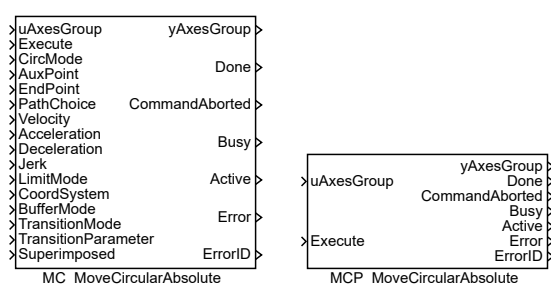




## MC\_MoveCircularAbsolute, MCP\_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)

Symbole bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

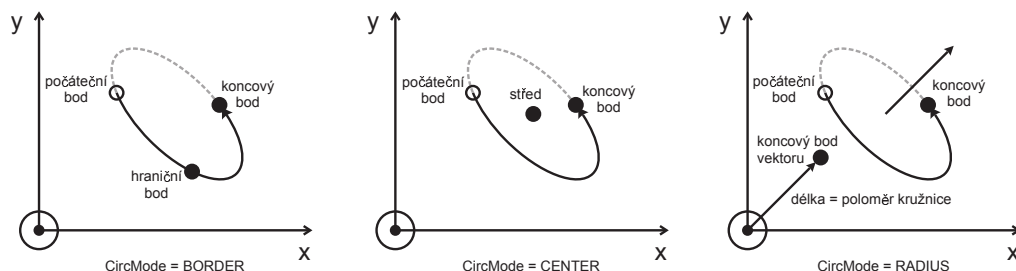
*Bloky MC\_MoveCircularAbsolute a MCP\_MoveCircularAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveCircularAbsolute` slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem `CoordSystem`. Parametry `Velocity`, `Acceleration`, `Deceleration` a `Jerk` určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr `EndPoint`. Pomocný bod je určen vektorovým parametrem `AuxPoint`. Tyto parametry musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC\\_SetKinTransform\\_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstupu `Execute`. K dispozici jsou následující způsoby zadání kružnice:

**BORDER** – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup `AuxPoint`).

**CENTER** – Zadává se koncový bod a střed kružnice (vstup `AuxPoint`). Vstup `PathChoice` potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.

**RADIUS** – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup `AuxPoint = (50,0,0)` odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (`CoordSystem = MCS`).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobů zadávání (parametr `CircMode`) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přeurečena a je nutné dát pozor, aby vzdálenost počátečního a koncevého bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncevého průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncevého nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

Poznámka 4: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty `TransitionMode`.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>CircMode</code>	Určuje význam vstupních signálů <code>AuxPoint</code> a <code>CircDirection</code>	Long (I32)
	1 ..... BORDER	
	2 ..... CENTER	
	3 ..... RADIUS	
<code>AuxPoint</code>	Absolutně zadaná pozice	Reference
<code>EndPoint</code>	Absolutně zadaná pozice koncevého bodu	Reference
<code>PathChoice</code>	Volba směru	Long (I32)
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)

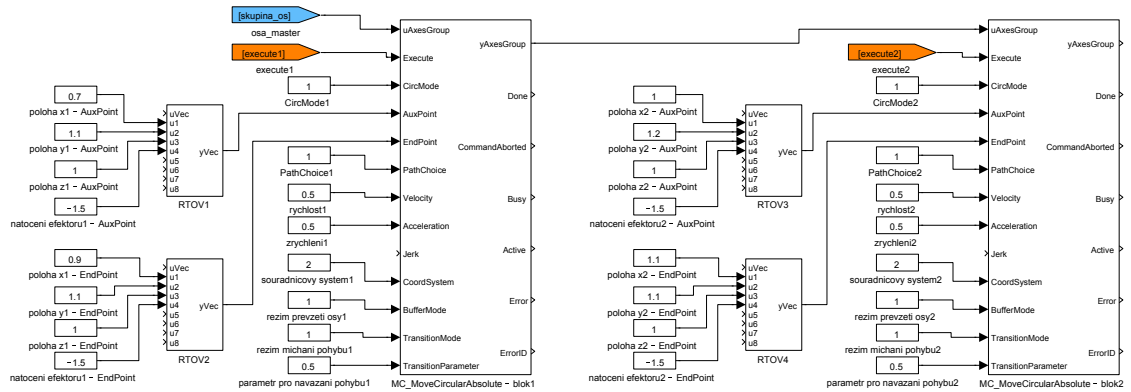
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>LimitMode</b>	Volba jednotek pro limity (Velocity, Acceleration, Jerk)   ⊙1	Long (I32)
	1 ..... Relative [part of group limit]	
	2 ..... Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	
<b>CoordSystem</b>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Výstupy

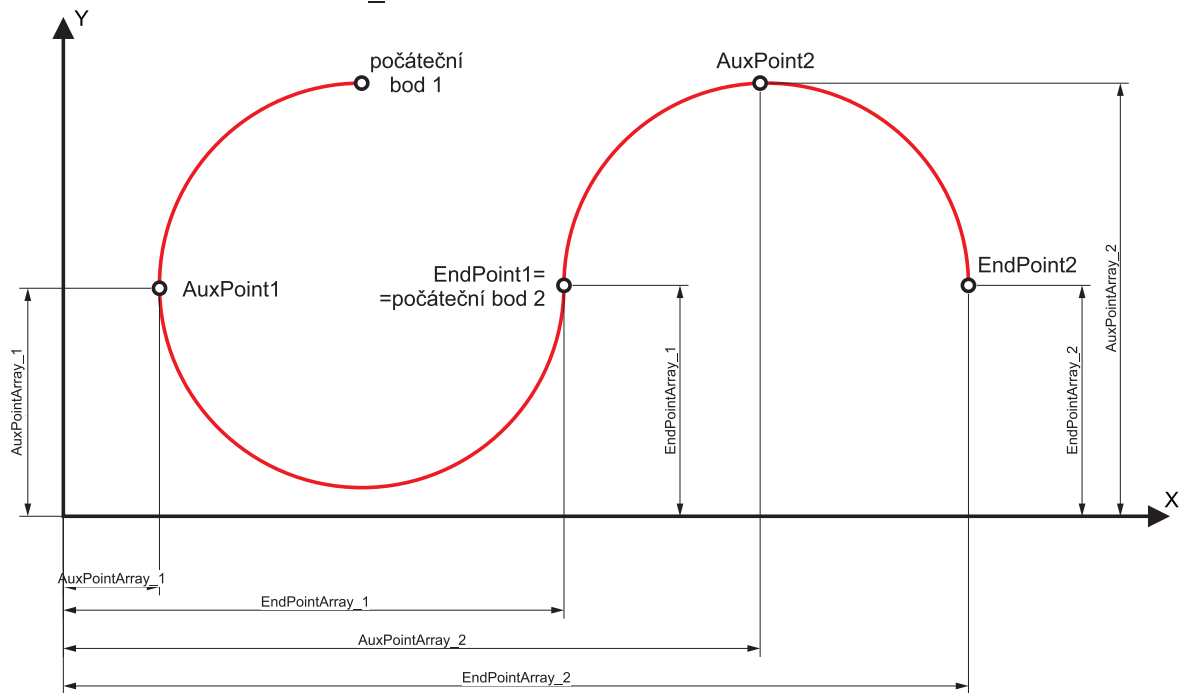
<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool

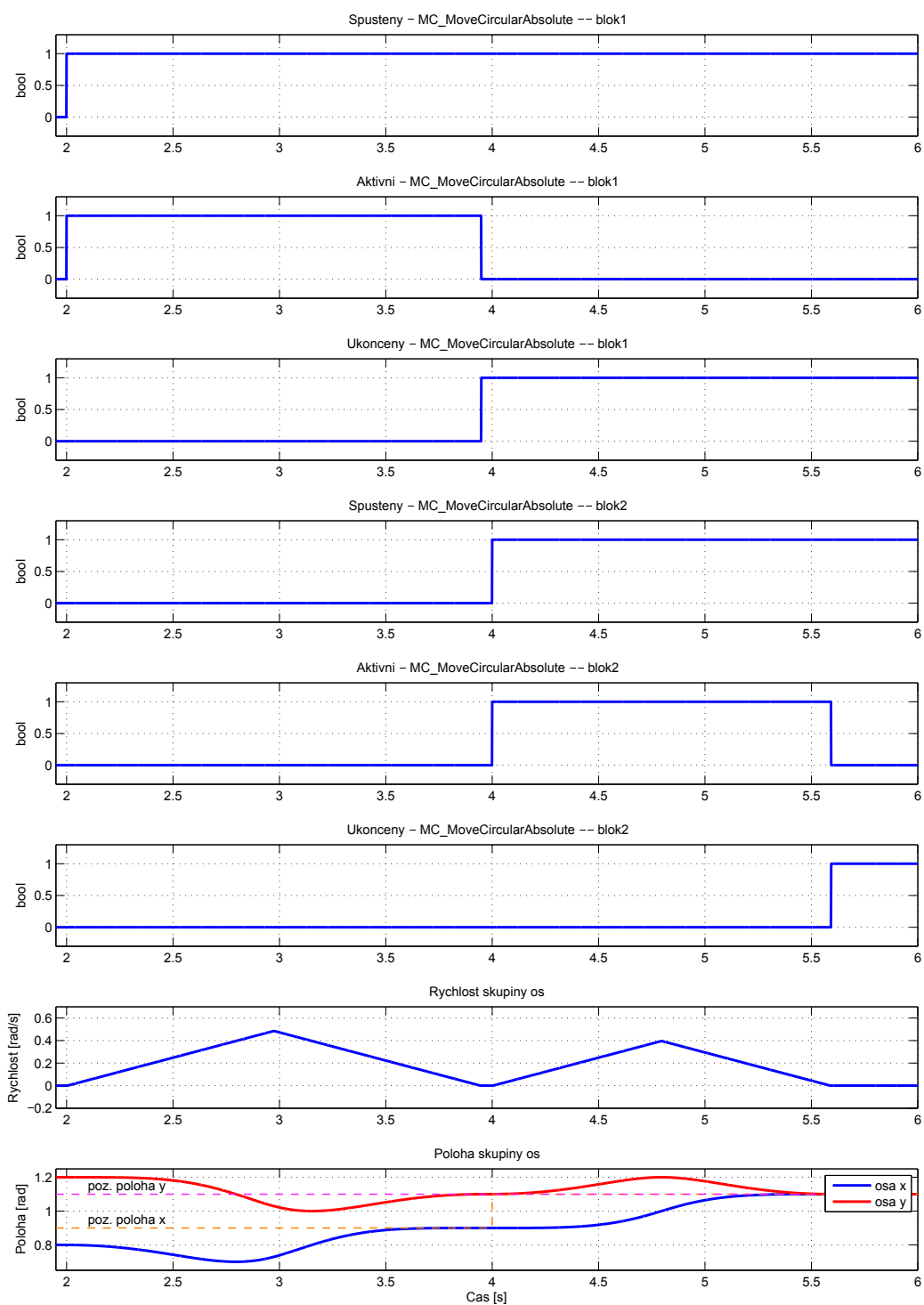
ErrorID Výsledek poslední operace  
 i ..... obecná chyba systému REXYGEN Error

Příklad



MC\_MoveCircularAbsolute - Příklad

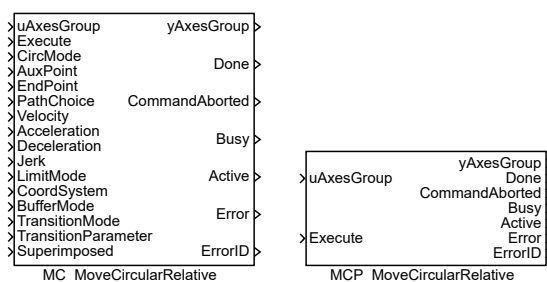




## MC\_MoveCircularRelative, MCP\_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)

Symbole bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

*Bloky MC\_MoveCircularRelative a MCP\_MoveCircularRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

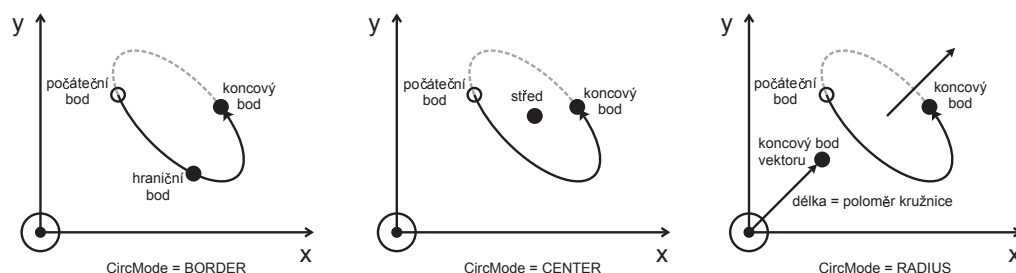
Blok `MC_MoveCircularRelative` slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem `CoordSystem`. Parametry `Velocity`, `Acceleration`, `Deceleration` a `Jerk` určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionalně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr `EndPoint`. Pomocný bod je určen vektorovým parametrem `AuxPoint`. Tyto parametry musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC\\_SetKinTransform\\_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstupu `Execute`. K dispozici jsou následující způsoby zadání kružnice:

**BORDER** – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup `AuxPoint`).

**CENTER** – Zadává se koncový bod a střed kružnice (vstup `AuxPoint`). Vstup `PathChoice` potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.



**RADIUS** – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup `AuxPoint = (50,0,0)` odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (`CoordSystem = MCS`).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobů zadávání (parametr `CircMode`) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přeurlčená a je nutné dát pozor, aby vzdálenost počátečního a koncového bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncového průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncového nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

Poznámka 4: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty `TransitionMode`.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>CircMode</code>	Určuje význam vstupních signálů <code>AuxPoint</code> a <code>CircDirection</code>	Long (I32)
	1 ..... BORDER	
	2 ..... CENTER	
	3 ..... RADIUS	
<code>AuxPoint</code>	Absolutně zadaná pozice	Reference
<code>EndPoint</code>	Absolutně zadaná pozice koncového bodu	Reference
<code>PathChoice</code>	Volba směru	Long (I32)
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)

<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>LimitMode</b>	Volba jednotek pro limity (Velocity, Acceleration, Jerk)    ☉1	Long (I32)
	1 ..... Relative [part of group limit]	
	2 ..... Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	
<b>CoordSystem</b>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

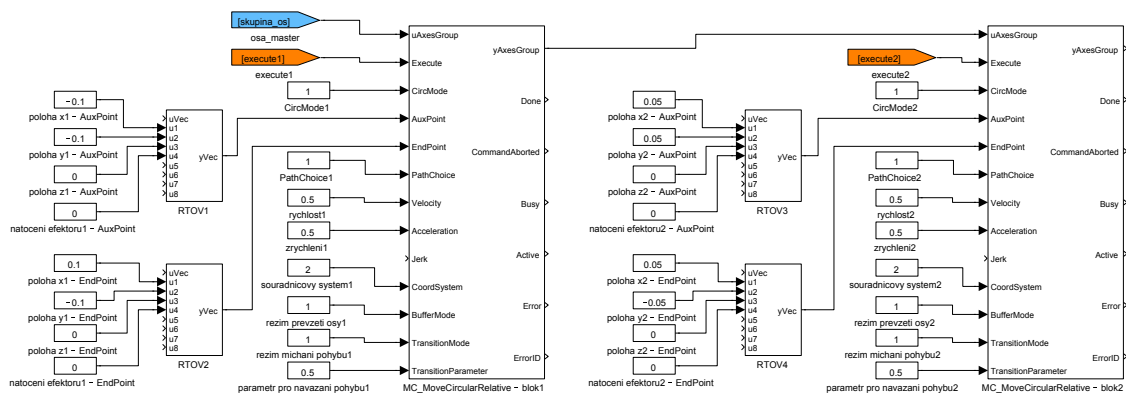
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool

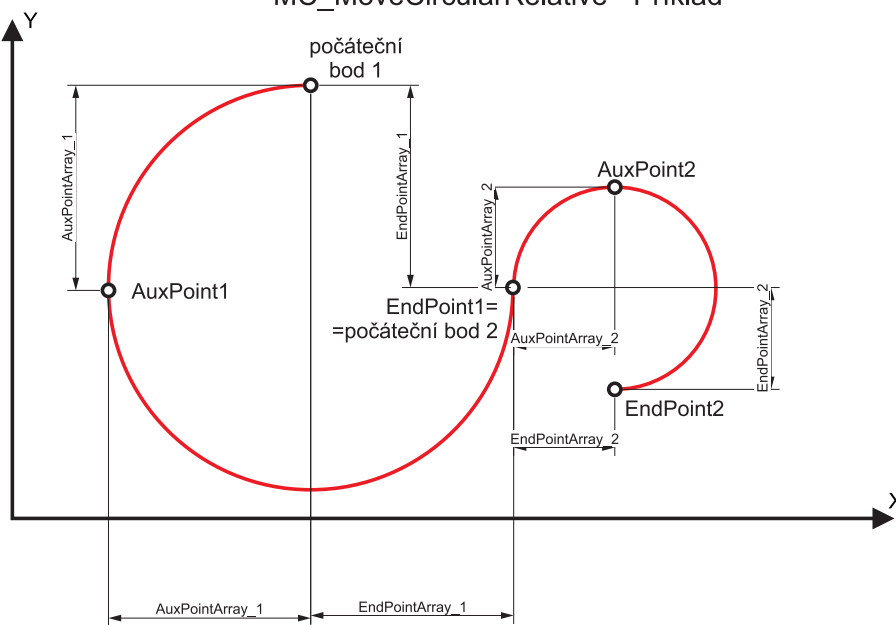
ErrorID Výsledek poslední operace  
i ..... obecná chyba systému REXYGEN

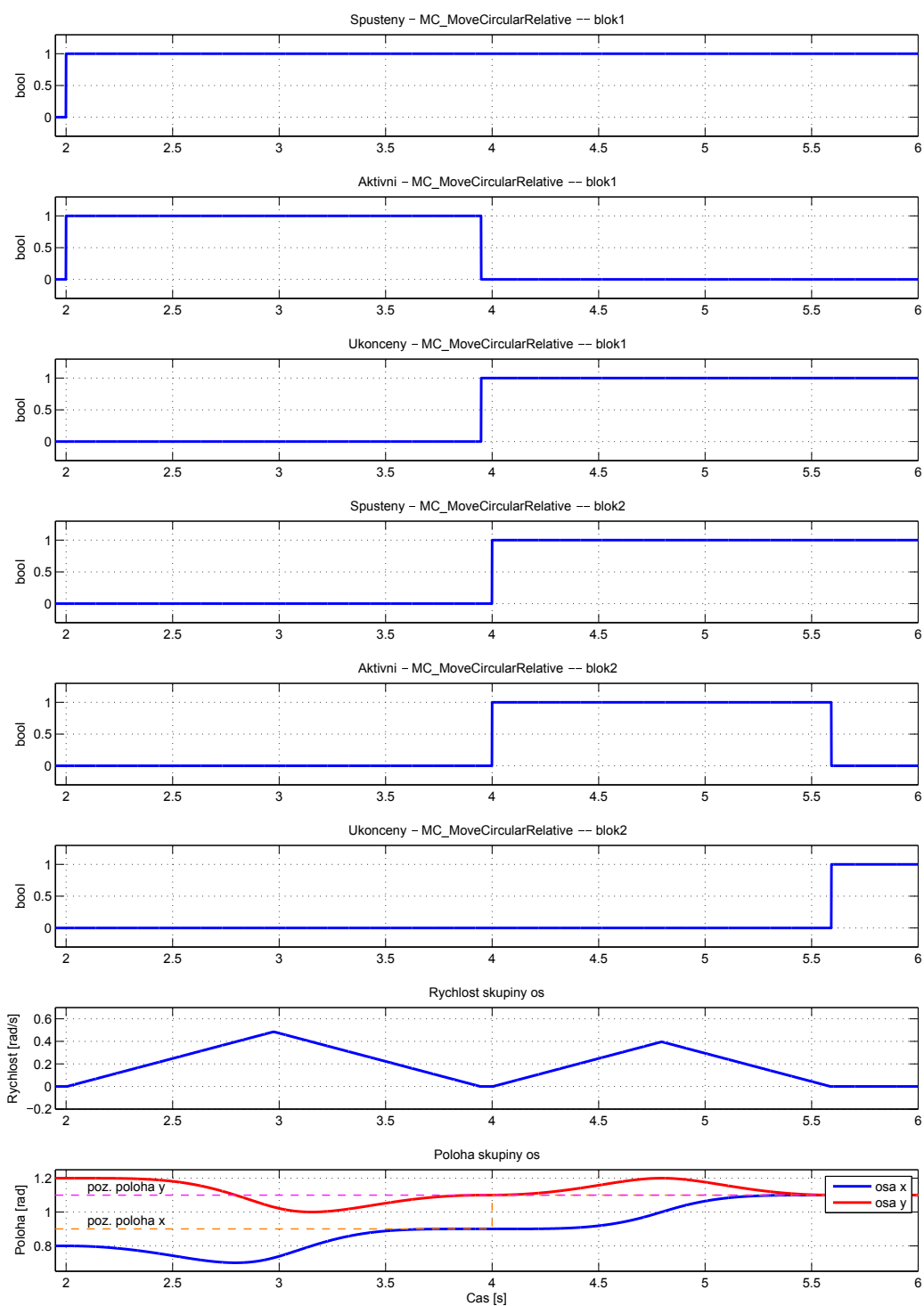
Error

Příklad



MC\_MoveCircularRelative - Příklad

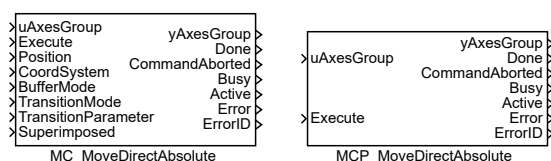




## MC\_MoveDirectAbsolute, MCP\_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice)

### Symbole bloků

Licence: **COORDINATED MOTION**



### Popis funkce

*Bloky MC\_MoveDirectAbsolute a MCP\_MoveDirectAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveDirectAbsolute** slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává v absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz **MC\_SetKinTransform\_Lin**). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

### Vstupy

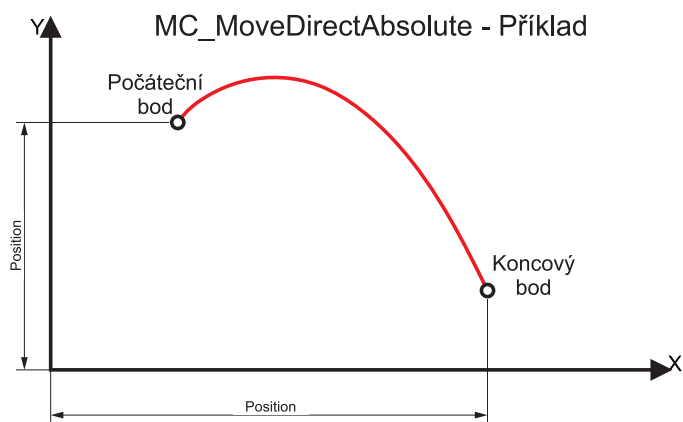
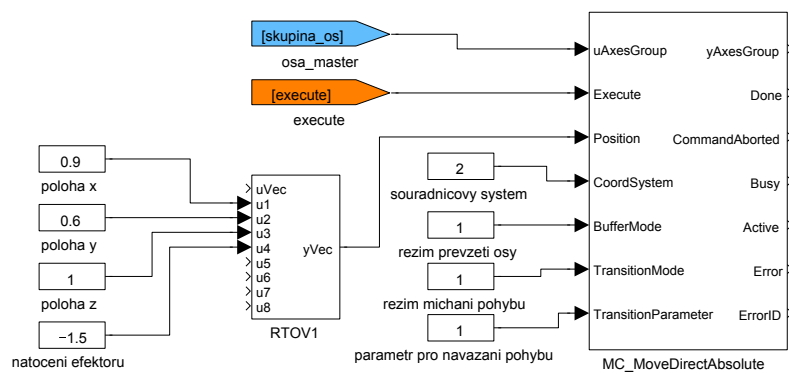
<b>uAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Position</b>	Pole souřadnic (pozic a orientací)	Reference
<b>CoordSystem</b>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

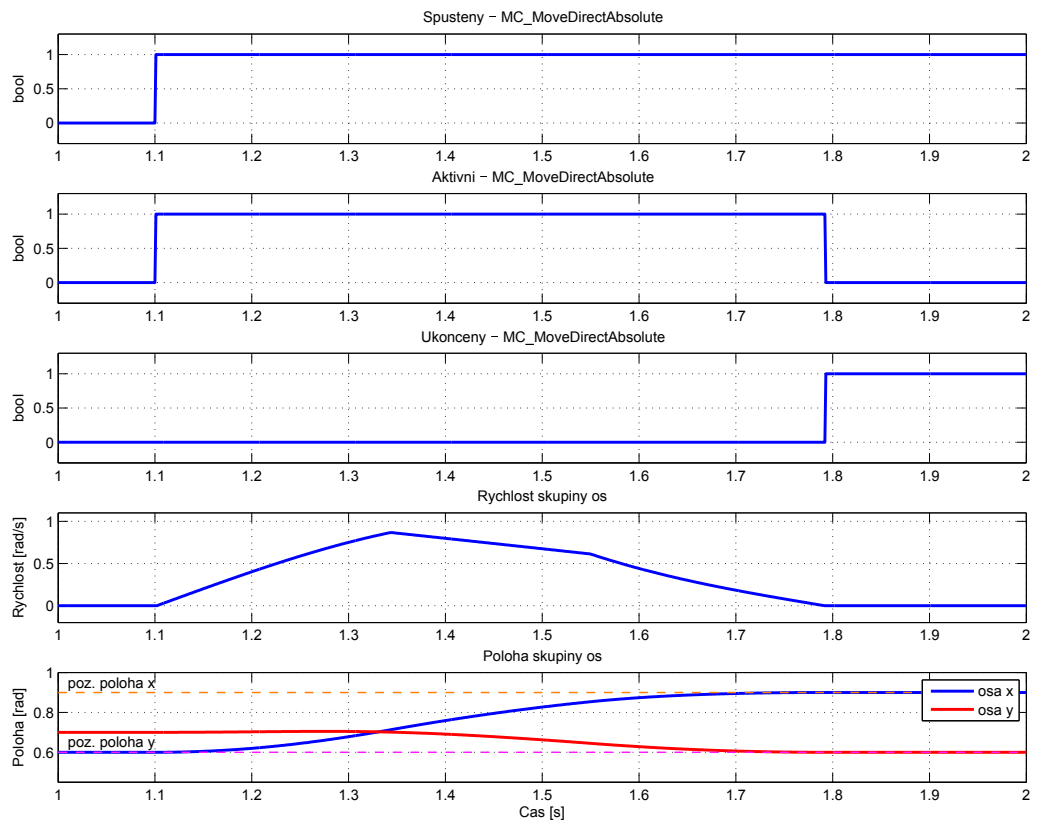
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
1	..... TMNone (xx)	
2	..... TMstartvelocity (proložení s danou počáteční rychlostí)	
3	..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	..... TMCornerDistance (xx)	
5	..... TMMaxCornerDeviation (xx)	
11	..... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
i	..... obecná chyba systému REXYGEN	

## Příklad



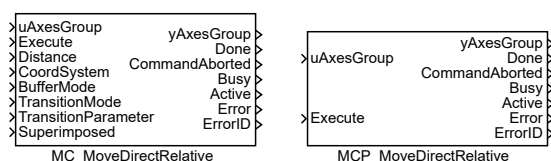




## MC\_MoveDirectRelative, MCP\_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

*Bloky MC\_MoveDirectRelative a MCP\_MoveDirectRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveDirectAbsolute` slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem `CoordSystem`. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC\\_SetKinTransform\\_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu `Execute`.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty `TransitionMode`.

Vstupy

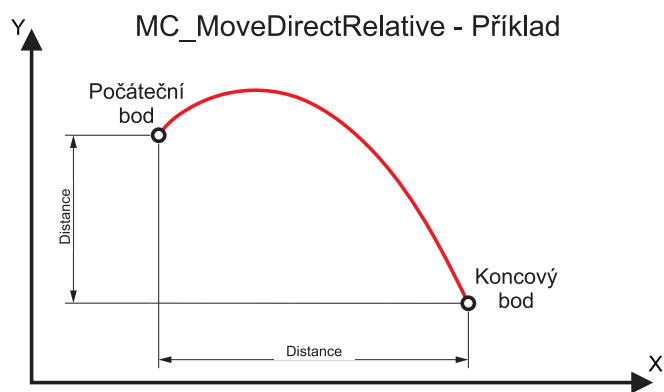
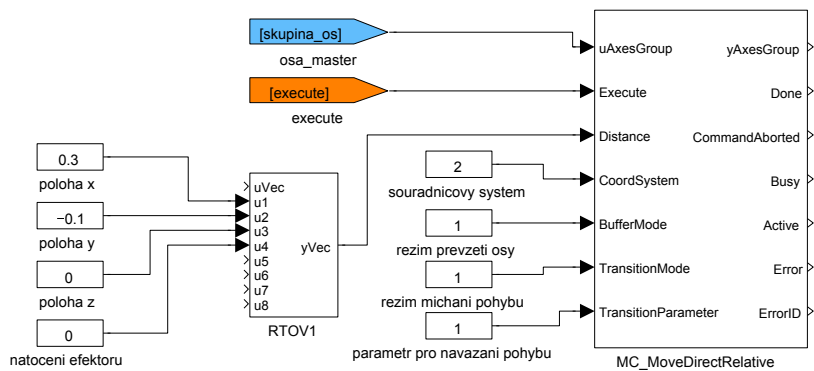
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Distance</code>	Pole souřadnic (relativních pozic a orientací)	Reference
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

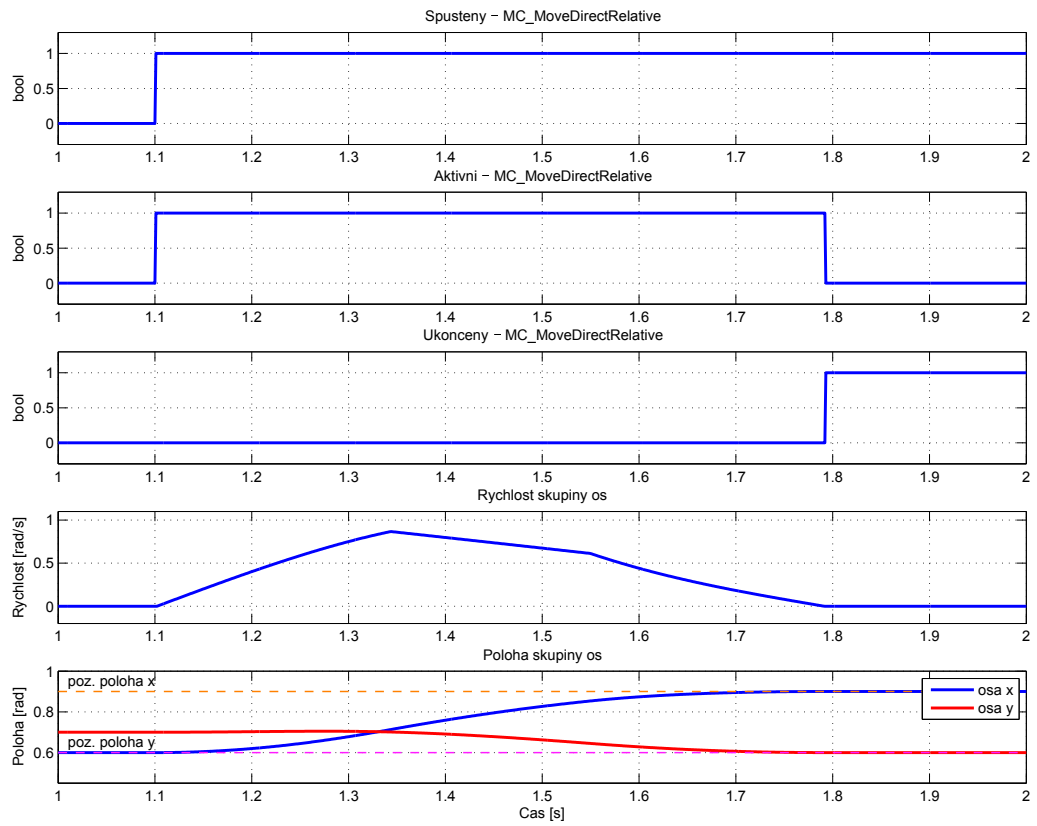
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
1	..... TMNone (xx)	
2	..... TMstartvelocity (proložení s danou počáteční rychlostí)	
3	..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	..... TMCornerDistance (xx)	
5	..... TMMaxCornerDeviation (xx)	
11	..... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
i	..... obecná chyba systému REXYGEN	

## Příklad

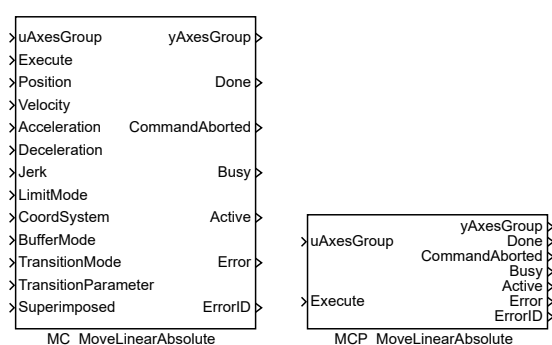




## MC\_MoveLinearAbsolute, MCP\_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice)

Symbole bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

*Bloky MC\_MoveLinearAbsolute a MCP\_MoveLinearAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveLinearAbsolute` slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem `CoordSystem`. Parametry `Velocity`, `Acceleration`, `Deceleration` a `Jerk` určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr `Position`. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC\\_SetKinTransform\\_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu `Execute`.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty `TransitionMode`.

### Vstupy

`uAxesGroup` Odkaz na skupinu os

Reference

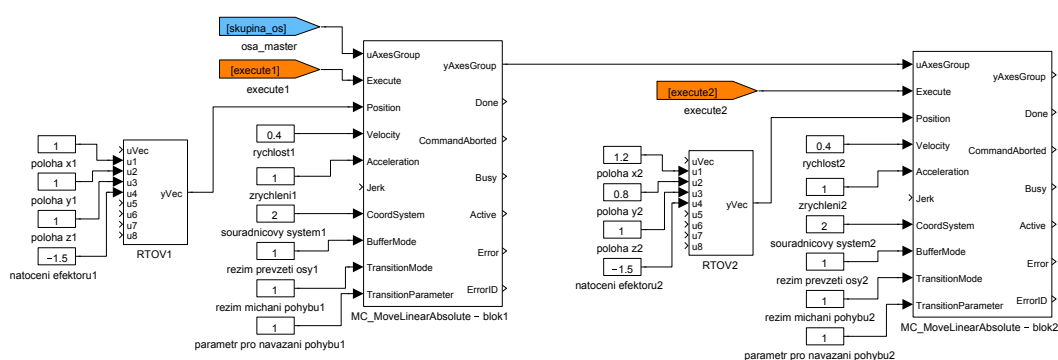
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Position</b>	Pole souřadnic (pozic a orientací)	Reference
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>LimitMode</b>	Volba jednotek pro limity (Velocity, Acceleration, Jerk)	⊙1 Long (I32)
	1 ..... Relative [part of group limit]	
	2 ..... Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	
<b>CoordSystem</b>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Výstupy

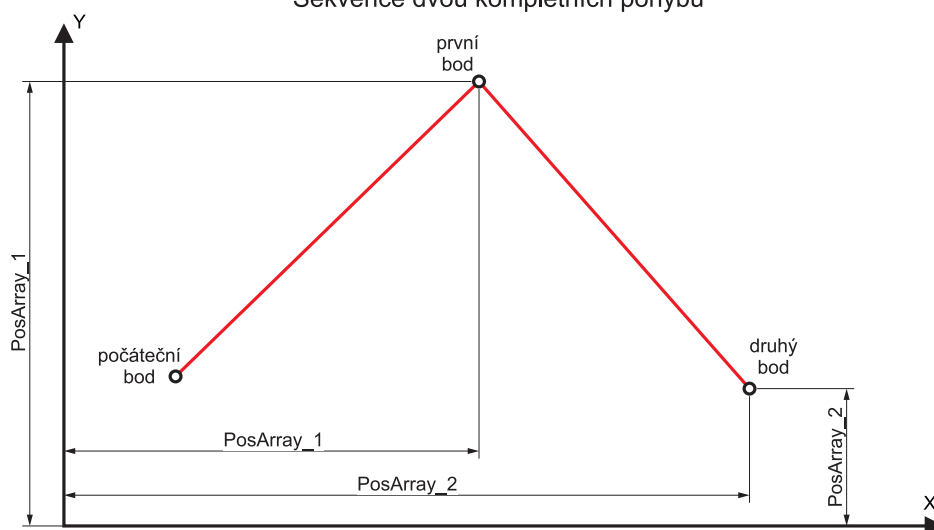
<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool

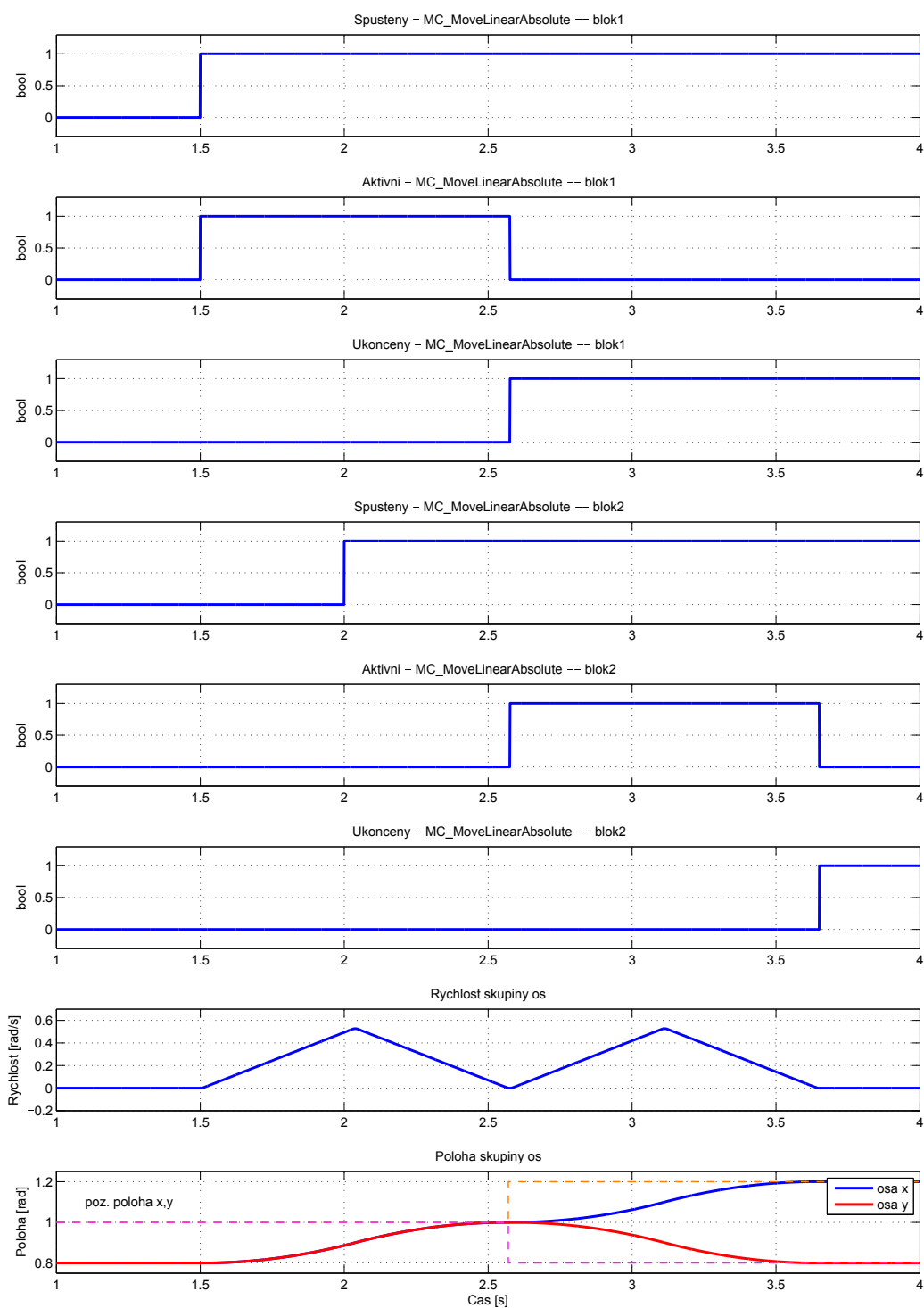
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i . . . . . obecná chyba systému REXYGEN	

## Příklad



## Sekvence dvou kompletních pohybů



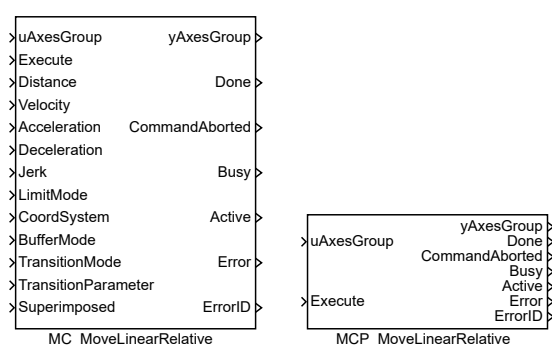




## MC\_MoveLinearRelative, MCP\_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

*Bloky MC\_MoveLinearRelative a MCP\_MoveLinearRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveLinearAbsolute** slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionalně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **Distance**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC\\_SetKinTransform\\_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

### Vstupy

**uAxesGroup** Odkaz na skupinu os

Reference

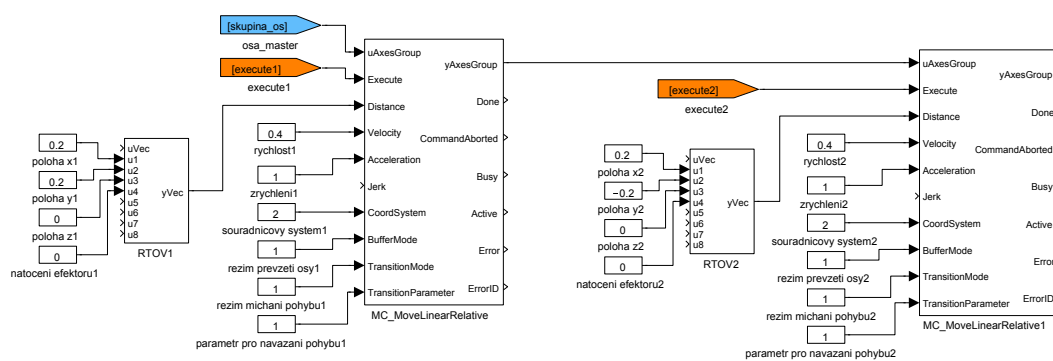
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>Distance</b>	Pole souřadnic (relativních pozic a orientací)	Reference
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	Double (F64)
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	Double (F64)
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	Double (F64)
<b>LimitMode</b>	Volba jednotek pro limity (Velocity, Acceleration, Jerk)	⊙1 Long (I32)
	1 ..... Relative [part of group limit]	
	2 ..... Absolute [unit/s, unit/s <sup>2</sup> , unit/s <sup>3</sup> ]	
<b>CoordSystem</b>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	Long (I32)
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Výstupy

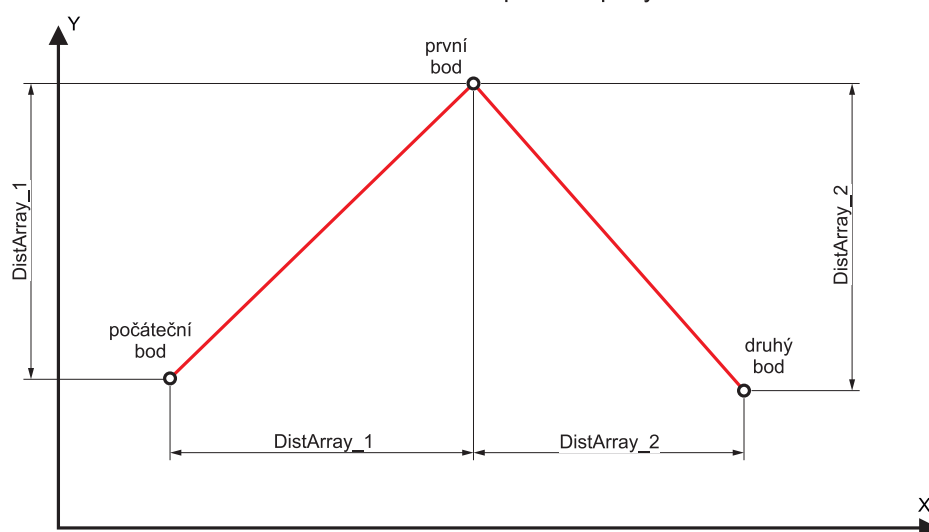
<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool

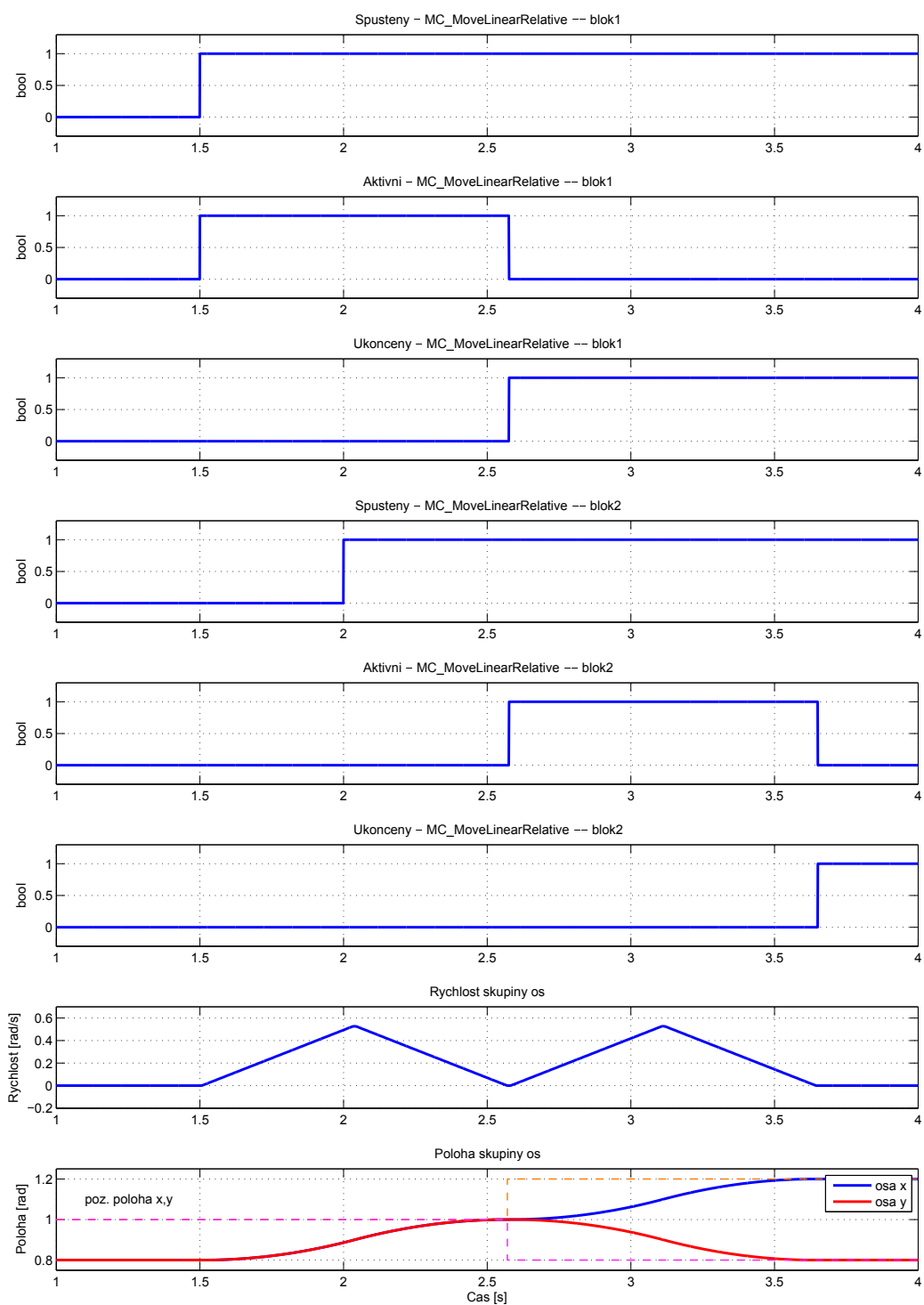
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## Příklad



Sekvence dvou kompletních pohybů

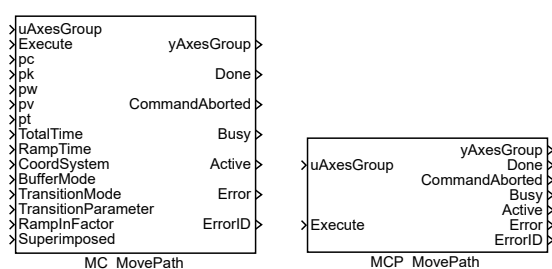




## MC\_MovePath, MCP\_MovePath – Generování obecné trajektorie v prostoru

### Symboly bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

*Bloky MC\_MovePath a MCP\_MovePath mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MovePath` slouží pro pohyb stroje (skupiny os) po definované křivce. Křivka je zadána pomocí NURBS aproximace pátého řádu. Křivka je vždy absolutní, proto je před spuštěním tohoto bloku nutné nastavit stroj do počátku křivky. V opačném případě je hlášena chyba -707 (skok v poloze). Blok umožňuje zadat parametry NURBS aproximace i ručně, ale to je prakticky neproveditelné (snad jen jako import z jiného programu). K tomuto bloku existuje speciální editor trajektorie (spouští se tlačítkem **SpecialEdit** v dialogu pro zadávání parametrů bloku), kde se křivka zadává/nakreslí pomocí bodů v prostoru, kterými má procházet a NURBS parametry jsou pak napočítány automaticky. NURBS parametrizace definuje pouze tvar křivky (pokud jistý parametr probíhá od 0 do 1 pak výstupní hodnota/vektor probíhá požadovanou křivku). Pro pohyb stroje musíme ještě definovat rychlost, s jakou křivku prochází. Blok podporuje 2 způsoby. Buď zadáme celkovou dobu jízdy a popřípadě dobu zrychlování na začátku a na konci pomocí parametrů `TotalTime` a `RampTime` nebo vytvoříme rychlostní profil pomocí parametrů `pv` a `pt`. Tento profil se skládá z několika intervalů a v každém se aproximuje polynomem 5. stupně. Ruční vytváření profilu je prakticky neproveditelné a editor trajektorií to zatím nepodporuje. Nouzově je možné použít editor bloku `MC_PositionProfile` (kde mají data stejný formát), přičemž poloha musí začínat v 0 a končit v 1. Ve většině případů však vystačíme s první metodou, protože editor trajektorií generuje NURBS tak, aby pro konstantní rychlost změny vstupního parametru byla přibližně konstantní rychlost pohybu po křivce. Teorie NURBS řeší aproximaci funkce jedné proměnné. Protože poloha je vektor, potřebujeme několik funkcí jedné proměnné (nezávisle proměnná je pro

všechny tyto funkce parametr, který probíhá interval od 0 do 1). Teoreticky máme tedy pro každou souřadnici jednu NURBS funkci, nicméně použitá implementace má některé parametry společné pro všechny souřadnice (například řád, uzlové body). Ačkoliv v bloku lze použít souřadný systém ACS, grafický editor trajektorie předpokládá pravoúhlý souřadný systém. Taktéž jsou formálně dovoleny všechny buffered a transition režimy, ale vzhledem k povaze bloku je vhodné používat jen buffered nebo blending next (ostatní většinou vedou na chybu -707 - skok v poloze). Tento blok nemá vstup `PathData`, který vyžaduje specifikace `PLCopen`. V systému `REXYGEN` má tento blok všechna potřebná data (nebo odkazy na ně) uložena v parametrech. Z tohoto důvodu se také nepoužívá blok `MC_PathSelect`.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>TotalTime</code>	Čas celého pohybu	Double (F64)
<code>RampTime</code>	Čas [s] zrychlování/zpomalování	Double (F64)
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<code>TransitionMode</code>	Režim míchání pohybu	Long (I32)
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
<b>RampInFactor</b>	faktor pro najíždění na počáteční polohu (0 = RampIn režim se nepoužívá), odpovídá přibližně relativní rychlosti a zrychlení	Double (F64)
<b>Superimposed</b>	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

## Parametry

<b>pc</b>	Matice řídicích bodů ⊙[0.0 1.0 2.0; 0.0 1.0 1.0; 0.0 1.0 0.0]	Double (F64)
<b>pk</b>	Uzlový vektor ⊙[0.0 0.0 0.0 0.0 0.5 1.0 1.0]	Double (F64)
<b>pw</b>	Váhový vektor ⊙[1.0 1.0 1.0]	Double (F64)
<b>pv</b>	Polynom pro korekci rychlosti posuvu ⊙[0.0 0.05 0.95; 0.0 0.1 0.1; 0.0 0.0 0.0; 0.1 0.0 -0.1; -0.05 0.0 0.05; 0.0 0.0 0.0]	Double (F64)
<b>pt</b>	Uzlové body pro korekci posuvu ⊙[0.0 1.0 10.0 11.0]	Double (F64)
<b>user</b>	Pouze pro special edit ⊙[0.0 1.0 2.0 3.0]	Double (F64)

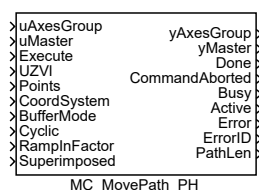
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## MC\_MovePath\_PH – \* Generování obecné trajektorie v prostoru PH

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>uMaster</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ..... ACS	
	2 ..... MCS	
	3 ..... PCS	
	4 ..... TCS	
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
<code>Cyclic</code>	Příznak cyklicky opakovaného profilu	Bool
<code>RampIn</code>	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)
<code>UZV1</code>		Reference
<code>Points</code>		Reference

### Parametr

<code>Superimposed</code>	příznak spuštění jako aditivní pohyb k existujícímu	Bool
---------------------------	---	------



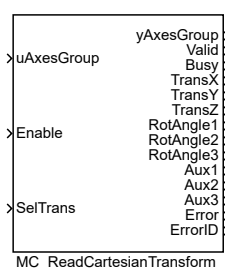
## Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yMaster</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
<code>PathLen</code>		Double (F64)

## MC\_ReadCartesianTransform – Přechtení použité kartézské transformace

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok `MC_ReadCartesianTransform` zpřístupňuje na výstup `TransX`, `TransY`, `TransZ`, `RotAngle1`, `RotAngle2`, `RotAngle3`, `Aux1`, `Aux2` a `Aux3` aktivní kartézskou transformaci mezi souřadnými systémy. V systému lze definovat několik kartézských transformací. Kterou chceme zobrazit se vybírá parametrem `SelTrans`. Význam hodnot je vysvětlen v popisu bloku `MC_SetCartesianTransform`. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true. Pokud je aktivní více než jedna kartézská transformace, tak je na výstupu dána výsledná kartézská transformace.

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>SelTrans</code>	Výběr transformace pro zobrazení	⊙1 Long (I32)
	1 . . . . . PCS offset	
	2 . . . . . Tool offset	
	3 . . . . . Machine Base offset	

### Výstupy

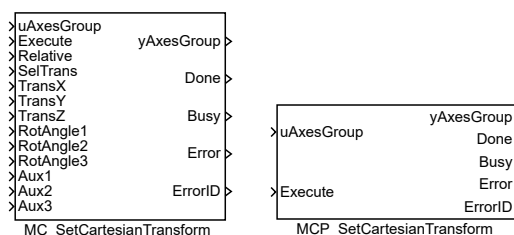
<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>TransX</code>	Posun v ose X	Double (F64)
<code>TransY</code>	Posun v ose Y	Double (F64)
<code>TransZ</code>	Posun v ose Z	Double (F64)
<code>RotAngle1</code>	Úhel rotace podél osy X	Double (F64)

RotAngle2	Úhel rotace podél osy Y	Double (F64)
RotAngle3	Úhel rotace podél osy Z	Double (F64)
Aux1	posun v 1. doplňkové ose	Double (F64)
Aux2	posun v 2. doplňkové ose	Double (F64)
Aux3	posun v 3. doplňkové ose	Double (F64)
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## MC\_SetCartesianTransform, MCP\_SetCartesianTransform – Kartézská transformace

Symbole bloků

Licence: [COORDINATED MOTION](#)



### Popis funkce

*Bloky MC\_SetCartesianTransforms a MCP\_SetCartesianTransforms mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_SetCartesianTransform` nastavuje kartézskou transformaci mezi souřadnými systémy. V systému lze definovat několik kartézských transformací. Kterou chceme nastavit se vybírá parametrem `SelTrans`. Pomocí vstupů `Trans` se nastavuje posunutí v jednotlivých osách. Pomocí vstupů `RotAngle` se pak nastavuje rotace kolem daných os (pořadě osa Z, osa Y, osa X). Pomocí vstupů `Aux` se nastavuje transformace dodatečných os, pokud je systém/kinematická transformace používá (transformace je přičtení v jednotlivých aux souřadnicích).

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os		Reference
<code>Execute</code>	Náběžná hrana aktivuje blok		Bool
<code>Relative</code>	Výběr absolutní (=false) nebo relativní (=true) transformace; absolutní transformace nastaví transformační matici podle nových hodnot, relativní transformace přidá nové hodnoty již nastavené transformaci (tj. vynásobí původní a novou transformační matici)		Bool
<code>SelTrans</code>	Výběr transformace pro nastavení	⊙1	Long (I32)
	1 ..... PCS offset; transformace mezi základnou (WCS) a výrobkem (PCS)		
	2 ..... Tool offset; transformace mezi koncem robota a středem nástroje		
	3 ..... Machine Base offset; transformace mezi základnou (WCS) a patou robota/stroje (MCS)		

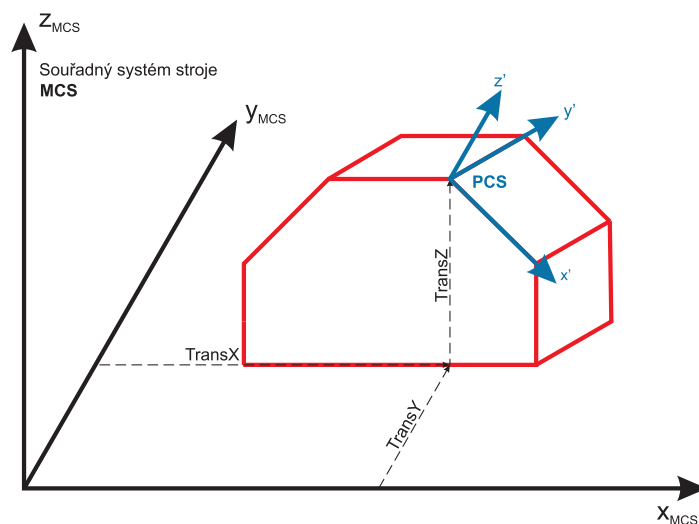
TransX	Posun v ose X	Double (F64)
TransY	Posun v ose Y	Double (F64)
TransZ	Posun v ose Z	Double (F64)
RotAngle1	Úhel rotace podél osy Z	Double (F64)
RotAngle2	Úhel rotace podél osy Y	Double (F64)
RotAngle3	Úhel rotace podél osy X	Double (F64)
Aux1	posun v 1. doplňkové ose	Double (F64)
Aux2	posun v 2. doplňkové ose	Double (F64)
Aux3	posun v 3. doplňkové ose	Double (F64)

## Výstupy

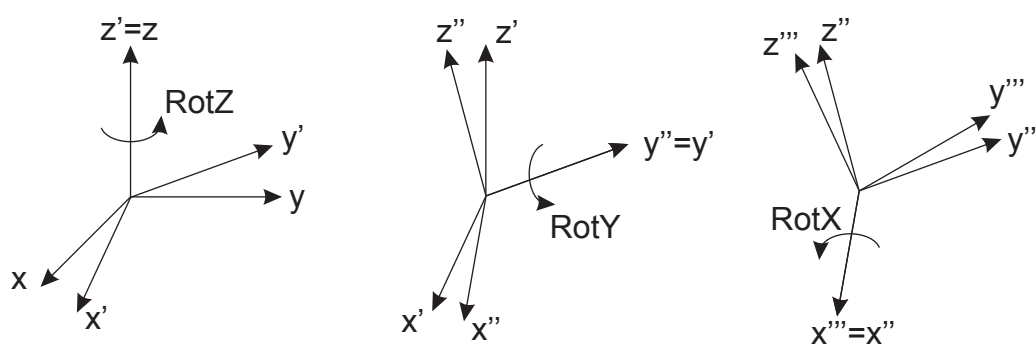
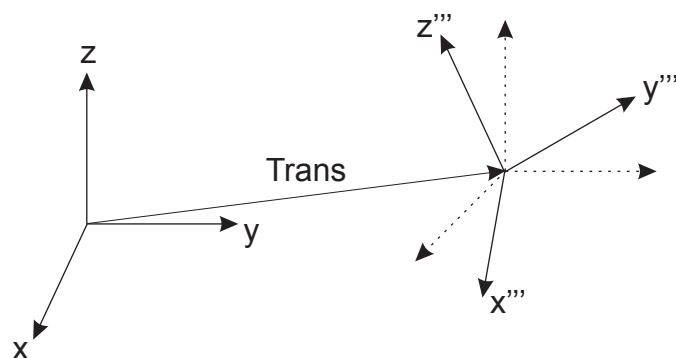
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

## Definice translace a rotace

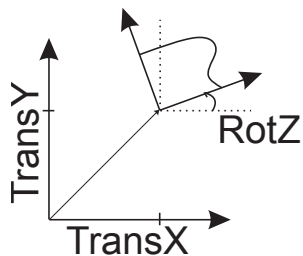
Na prvním obrázku je uveden příklad translace ze souřadného systému MCS do souřadného systému PCS.



Na druhém obrázku je pak uveden příklad rotace. Kdy celková rotace je docílena postupným provedením dílčích rotací kolem jednotlivých os.



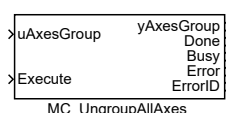
Ukázka zadání kartézské transformace (v rovině). Pro dosažení transformace na obrázku níže se spustí blok `MC_SetCartesianTransform` s nastavenou transformací  $\{50,50,0,30,0,0\}$



## MC\_UngroupAllAxes – Odebrání všech os ze skupiny

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok `MC_UngroupAllAxes` odebere všechny osy ze dané skupiny os. Po dokončení je změněn stav skupiny na „GroupDisabled“. Blok kromě odebrání os odebere i kinematickou transformaci, kartézské transformace a všechna další nastavení, tj. převede stav bloku `RM_AxisGroup` do počátečního stavu.

Poznámka 1: Pokud je blok spuštěn a skupina os není ve stavu „GroupDisabled“, „GroupStandBy“ nebo „GroupErrorStop“, tak je vyvolána chyba a blok není spuštěn.

Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

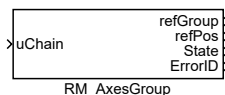
Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN	Error

## RM\_AxesGroup – Skupina os pro koordinované řízení pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **RM\_AxesGroup** je základní blok skupiny os pro koordinovaný pohyb. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry skupiny. Algoritmus tohoto bloku představuje kontrolu nastavených mezí, havarijní zastavení v případě potřeby, předávání dat do a z bloků **RM\_Axis** podřízených os, přepočítání všech stavů a výstupů pro případ, že žádný blok není aktivní, ale je potřeba generovat pohyb (např. zastavení v PCS a částečně i algoritmus pro generování požadované trajektorie). Výstupem tohoto bloku jsou pouze stavové a pomocné signály (viz dále).

Před spuštěním koordinovaného pohybu je nutné ke skupině (bloku **RM\_AxesGroup**) připojit jednotlivé osy (spuštěním bloku **MC\_AddAxisToGroup** pro každou osu), přiřadit kinematickou transformaci (spuštěním bloku jednoho z bloků **MC\_SetKinTransform\_Xxx**) a skupinu povolit (spuštěním bloku **MC\_GroupEnable**).

Další důležité administrativní bloky pro řízení stavu skupiny os jsou **MC\_GroupReset** (smaže chybu, tj. přepíná skupinu ze stavu **ErrorStop** do stavu **Standstill**), **MC\_GroupDisable** (přepíná skupinu os do stavu **PowerOff**) a **MC\_UngroupAllAxis** (ruší všechna nastavení, tj. odebere osy, odebere kinematickou transformaci, přepne do stavu **PowerOff**).

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

Víme, že blok **RM\_AxisSpline** může být v rychlejší úloze než blok **RM\_Axis** a pak se musí provádět interpolace polohy a rychlosti. Podobně bloky **RM\_Axis** mohou být v rychlejší úloze než **RM\_AxesGroup** a pak je potřeba provádět interpolaci. Interpolaci je potřeba provádět v rychlejší úloze (tj. v blocích **RM\_Axis**), ale režim interpolace se zadává v bloku **RM\_AxesGroup** parametrem **InterpolationMode**.

Vektortové parametry **min** a **max** jsou nepovinné. Pokud jsou zadány, zadávají se ve stejném formátu jako poloha skupiny (tj. např. parametr **Position** bloku **MC\_MoveLinearAbsolute**). Tento formát je definován kinematickou transformací, tj. blokem **MC\_SetKinTransform\_Lin** nebo podobným (tj. **MC\_SetKinTransform\_Xxx**). Pohyb je zastaven (a blok **RM\_AxesGroup** přejde do stavu **ErrorStop**, pokud hrozí překročení hodnoty v kterékoliv souřadnici. rotační souřadnice se ale nekontrolují.

Poznámka: U omezení na rychlost otáčení (parametr **VelRot**) je základní jednotka radián za sekundu. Připojená kinematická transformace však může změnit jednotku pro



zadávaní úhlu (např. na úhlové stupně nebo otáčky) a pak se rychlost zadává v těchto jednotkách za sekundu. Stejně pravidlo platí pro zrychlení a jerk, tj. parametry `AccRot` a `JerkRot`.

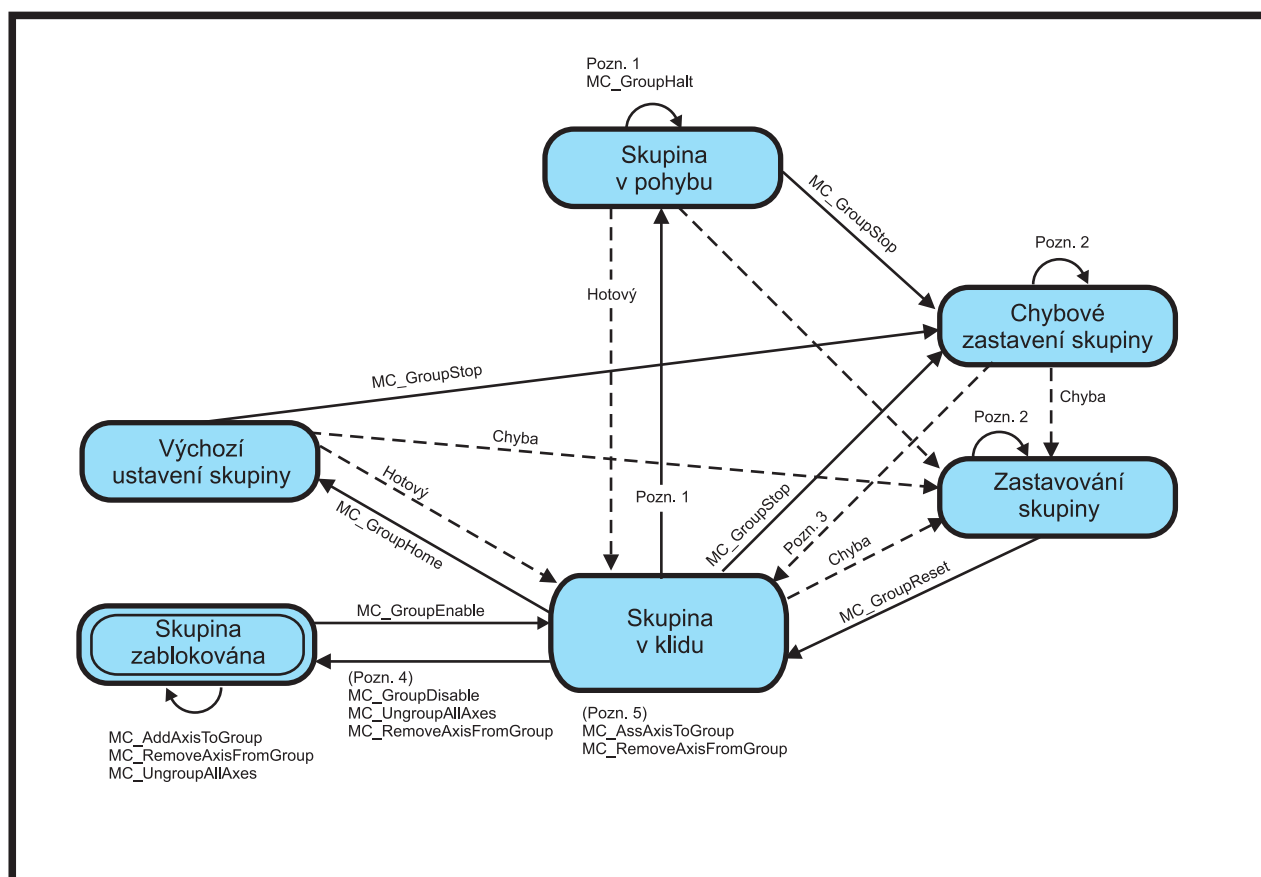
## Parametry

<code>Velocity</code>	Maximální povolená rychlost [unit/s]	<code>Double</code> (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	<code>Double</code> (F64)
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<code>Double</code> (F64)
<code>VelRot</code>	Maximální povolená rychlost otáčení [rotunit/s]	<code>Double</code> (F64)
<code>AccRot</code>	Maximální povolené zrychlení v otáčení [rotunit/s <sup>2</sup> ]	<code>Double</code> (F64)
<code>JerkRot</code>	Maximální povolená změna zrychlení v otáčení [rotunit/s <sup>3</sup> ]	<code>Double</code> (F64)
<code>InterpolationMode</code>	Algoritmus pro interpolaci <span style="float: right;">⊙2</span>	<code>Long</code> (I32)
	1 ..... linear (poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0, tj. rychlost je po částech konstantní funkce se skoky)	
	2 ..... cubic spline (poloha je polynom 3. řádu vypočtený na základě polohy a rychlosti na začátku a konci intervalu, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	3 ..... quintic spline (poloha je polynom 5. řádu vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	4 ..... cubic Bspline approximation (poloha je polynom 3. řádu vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem, proložená funkce nemusí přesně procházet zadanými body, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	5 ..... quintic Bspline approximation (poloha je polynom 5. řádu vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem, proložená funkce nemusí přesně procházet zadanými body, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	6 ..... all linear (poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti)	
	7 ..... all cubic (poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy)	
	8 ..... rezervováno pro pozdější použití	
	9 ..... rezervováno pro pozdější použití	
<code>min</code>	Minimální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu <code>ErrorStop</code> <span style="float: right;">⊙[]</span>	<code>Double</code> (F64)
<code>max</code>	Maximální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu <code>ErrorStop</code> <span style="float: right;">⊙[]</span>	<code>Double</code> (F64)

## Výstupy

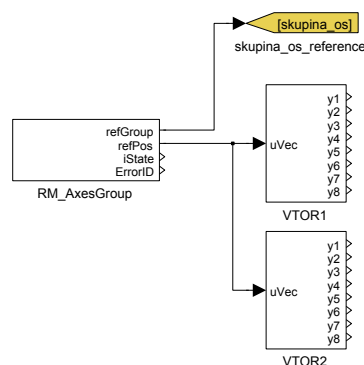
refGroup	Odkaz na skupinu os	Reference
refPos	Vektor generovaných poloh a rychlostí v MCS	Reference
iState	Stav skupiny	Long (I32)
	0 ..... Disabled (skupina blokována)	
	1 ..... Standby (skupina připravena)	
	2 ..... Homing (hledání výchozí polohy)	
	6 ..... Moving (koordinovaný pohyb)	
	7 ..... Stopping (zastavování nebo dočasné blokování)	
	8 ..... Error stop (zastavování nebo blokování skupiny po chybě)	
ErrorID	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

## Stavový diagram skupiny os



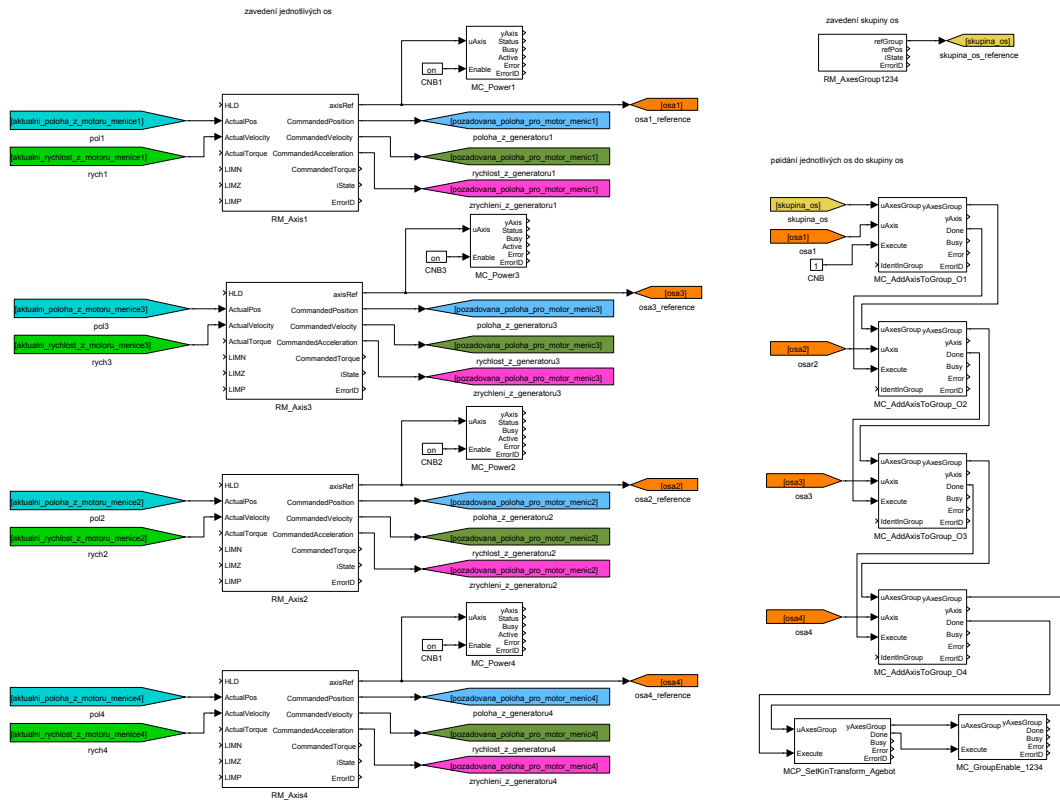
## Čtení z výstupu refPos

Z výstupu `refPos` bloku `RM_AxesGroup` lze číst aktuální polohu a rychlost (stejně jako pomocí bloků `MC_GroupReadActualPosition`, `MC_GroupReadActualVelocity`; zrychlení zde není vyčíslováno a musí se použít `MC_GroupReadActualAcceleration`). Výstup `refPos` je vektorový, pro získání konkrétních hodnot je nutné použít blok `VTOR` z knihovny `MA-TRIX`. Tento blok má 8 výstupů. Pokud je souřadnic více než 8, vyřeší se to paralelním připojením dalšího bloku `VTOR` (viz. Obr. níže) do kterého je nutné zadat jako parametr správný offset (od jakého indexu se mají dát data na výstup). Např. robot AGEBOT (jeho sestavení jako skupiny os viz. níže) má čtyři osy. Bude mít tedy čtyři souřadnice pro polohu a rychlost. Jedním blokem `VTOR` získáme informaci o poloze a rychlosti (výstupy  $y_1 - y_4$  a  $y_5 - y_8$  bloku `VTOR1`).



## Sestavení skupiny os

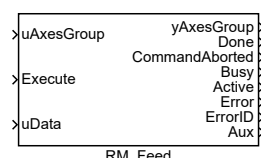
Pro správné spuštění bloku generujícího víceosý pohyb je nutné definovat jednotlivé osy `RM_Axis` a postupně je všechny přidat do skupiny os pomocí bloku `MC_AddAxisToGroup`. Dále je třeba nastavit kinematickou transformaci (dle typu stroje jeden z bloků `MC_SetKinTransform_XXX`). Nakonec je třeba aktivovat skupinu os blokem `MC_GroupEnable`. Pro pohyby v PCS je nutné ještě přidat kartézskou transformaci (blok `MC_SetCartesianTransforms`). Zpět do výchozího stavu (tj. před přiřazení os do skupiny) se skupina dostane spuštěním bloku `MC_UngroupAllAxis`. Příklad sestavení osy je uveden na obrázku níže.



## RM\_Feed – MC „krmič“

Symbol bloku

Licence: **COORDINATED MOTION**



### Popis funkce

Tento blok slouží pro předání předem vygenerované trajektorie do infrastruktury pro koordinované řízení pohybu. Data se předávají pomocí matice přivedené na vstup **uData**. V základním režimu, kdy **VelFactor=1** je v každé periodě úlohy nastavena hodnota z dalšího sloupce matice. Hodnoty musí být ve sloupci v pořadí: číslo předávané na výstup **aux**, vektor polohy, dále nepovinně vektor rychlosti, nepovinně vektor zrychlení, nepovinně vektor sil/momentů. Pokud vektor rychlosti a/nebo zrychlení není zadán, dopočítává se diferencí. Momenty v celém systému řízení pohybu slouží jen jako dopředná vazba pro zpětnovazební regulátor servoměniče, takže pokud není znám generuje se 0 (což ostatně dělá i většina ostatních pohybových bloků). Vektor polohy/rychlosti/zrychlení se zadává ve stejném tvaru jako pro ostatní pohybové bloky, např. **MC\_MoveLinearAbsolute**. Pokud **VelFactor** není 1, hodnoty polohy/rzchlosti/zrychlení jsou interpolovány lineárně, přičemž rychlost a zrychlení musí být násobeno příslušným faktorem (aby rychlost stále odpovídala derivaci polohy a zrychlení derivaci rychlosti).

Význam ostatních parametrů je stejný jako pro jiné pohybové bloky. Z principu funkce však musí být odstartován ze správné polohy, jinak dojde k chybě -707 (skok v poloze).

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Execute</b>	Náběžná hrana aktivuje blok	Bool
<b>uData</b>	Odkaz na matici s daty	Reference

### Parametry

<b>VelFactor</b>	Faktor rychlosti. Pro nominální rychlost (tak jak byla data vygenerována) volíme 1, pro zpomalení pohybu čísla menší než 1 a pro zrychlení čísla větší než 1	Double (F64) ↓0.01 ↑100.0 ⊙1.0
<b>Relative</b>	Pokud je zatrženo, poloha v datech se přičte k poloze na startu (tj. první sloupec v matici by měl být 0)	Bool

<b>CoordSystem</b>	Volba souřadného systému	↓1 ↑3 ⊙2	Long (I32)
	1 ..... ACM		
	2 ..... MCS		
	3 ..... PCS		
<b>BufferMode</b>	Režim převzetí osy	↓1 ↑6 ⊙1	Long (I32)
	1 ..... Aborting (nový blok se spustí okamžitě)		
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)		
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)		
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)		
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)		
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
<b>TransitionMode</b>	Režim míchání pohybu	↓0 ↑15 ⊙1	Long (I32)
	1 ..... TMNone (xx)		
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)		
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)		
	4 ..... TMCornerDistance (xx)		
	5 ..... TMMaxCornerDeviation (xx)		
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)

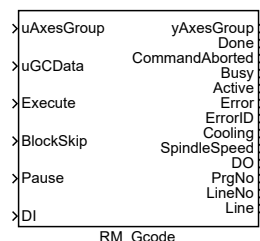
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	Reference
<b>Done</b>	Příznak dokončení algoritmu	Bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	Bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	Bool
<b>Active</b>	Příznak, že blok řídí osu	Bool
<b>Error</b>	Příznak chyby	Bool
<b>ErrorID</b>	Výsledek poslední operace	Error
<b>Aux</b>	První hodnota v datech pro pomocné účely	Double (F64)

## RM\_Gcode – CNC řízení pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

G-code původně vznikl pro řízení obráběcích strojů pomocí počítače kolem roku 1950 (tzv. technologie NC a CNC). Tehdy se zadával pomocí děrné pásky nebo děrného štítku, proto je poměrně úsporný. V současnosti je to stále dominantní způsob zadávání sekvence pohybu (tj. programu) pro obráběcí stroje, ale i některé další typy strojů, např. řezacích plotrů, 3D tiskáren. Ačkoliv byl G-code standardizován již v 60. letech s drobnými úpravami v 80. letech, výrobci strojů používají různá rozšíření, aby bylo možné využít schopnosti jejich stroje. Dále jsou drobné rozdíly podle typu stroje (frézy mají některé příkazy odlišné od soustruhů nebo 3D tiskáren). Základní struktura příkazu/parametru je sekvence písmeno následovaná číslem. Příkazy na jedné řádce se považují za zadané současně, tj. nezávisí na pořadí. Příkazy/parametry, které na řádce nejsou se považují za nezměněné od minulého výskytu. Další informace k G-code např. <https://en.wikipedia.org/wiki/G-code>. V dnešní době se G-code prakticky vždy generuje z CAD programu.

V základní podobě se soubory s G-code umísťují do adresáře určeného parametrem `BaseDir` na cílové platformě. Soubory musí mít název <čtyřciferné číslo>.nc. Číslo programu, který se má spustit se pak zadává v parametru `MainFile`, přičemž program může volat další podprogramy (v G-code se identifikují číslem, tj. soubor ve stejném formátu i ve stejném adresáři jako hlavní program). Např. pokud `MainFile = 1`, otevírá se soubor `0001.nc`.

Druhá možnost je na vstup `uGCData` připojit speciální blok, který generuje řádky G-code z jiných dat.

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>uGCData</code>	Odkaz na speciální blok generující G-code	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>BlockSkip</code>	Přeskočení bloku v G-code (pokud program přijde na řádku, kde je znak lomítka a je <code>BlockSkip=true</code> , tak se řádka přeskočí)	Bool





<code>workOffsets</code>	Sady počátečních souřadnic	⊙[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	Double (F64)
<code>toolOffsets</code>	Sady korekcí délky nástroje	⊙[0 0 0]	Double (F64)
<code>cutterOffsets</code>	Poloměry nástroje	⊙[0 0 0]	Double (F64)

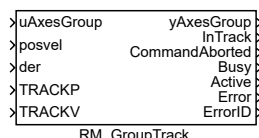
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os		Reference
<code>Done</code>	Příznak dokončení algoritmu		Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku		Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil		Bool
<code>Active</code>	Příznak, že blok řídí osu		Bool
<code>Error</code>	Příznak chyby		Bool
<code>ErrorID</code>	Výsledek poslední operace i . . . . . obecná chyba systému REXYGEN		Error
<code>Cooling</code>	Chlazení zapnuto		Bool
<code>SpindleSpeed</code>	Rychlost otáčení vřetene		Double (F64)
<code>D0</code>	Celé číslo, které slouží jako bitové pole výstupních logických signálů, které lze v G-code nastavovat		Long (I32)
<code>PrgNo</code>	Číslo právě vykonávaného programu		Long (I32)
<code>LineNo</code>	Číslo právě vykonávané řádky programu (parametr N v G-code)		Long (I32)
<code>Line</code>	Právě prováděná řádka G-kódu		String

## RM\_GroupTrack – \* Sledování polohy/rychlosti

Symbol bloku

Licence: [COORDINATED MOTION](#)



### Popis funkce

Blok **RM\_GroupTrack** slouží pro sledování externího signálu. Je to obdoba bloku **RM\_Track** pro koordinovaný pohyb. Blok se aktivuje nastavením vstupu **TRACKP = 1** nebo **TRACKV = 1**. V prvním případě se blok snaží co nejrychleji (ovšem s respektováním maximální zadané rychlosti, zrychlení, popřípadě jerku nastavené parametry **Velocity**, **Acceleration**, **Jerk**) dosáhnou polohy přivedené na vstup **posvel**. Poloha se může v každém kroku algoritmu měnit a blok se stále přepočítává trajektorii a snaží se sledovat vstupní signál. Vstupní vektor má stejný formát jako poloha v bloku **MC\_MoveLinearAbsolute** a podobných (tj. přesený formát definuje kinematická transformace). Ve druhém případě se na vstup **posvel** přivádí vektor rychlosti a blok se opět snaží řídit skupinu os, aby co nejpřesněji odpovídala požadované hodnotě, ale byly respektovány zadaná omezení. Na vstup **der** se přivádí derivace signálu přivedeného na **posvel** (tj. v prvním režimu to odpovídá rychlosti, ve druhém zrychlení). Protože poloha/rychlost je vektorová veličina, je i její derivace vektor. Pokud hodnota derivace není známa, ponechává se vstup nepřipojen a derivace je odhadována z předchozího průběhu signálu (diferencí z několika posledních hodnot podle parametru **iLen**). Pokud je ale derivace známa, je vhodné ji připojit, protože to zlepšuje přesnost sledování trajektorie. Protože trajektorie, která není dopředu známa nelze přesně sledovat, nelze u tohoto bloku přesně definovat trajektorii podobně jako u bloku **MC\_MoveDirectAbsolute**. Omezení na rychlost/zrychlení/jerk jsou kontrolována v jednotlivých složkách a celková hodnota ve směru trajektorie může být o něco větší. Zajímavý je režim **CoordSystem = TCS**, kde se zadává rychlost v souřadnicích spojených s koncovým efektem robota. Tento režim se velmi hodí pro ruční navádění robota a zcela pochopitelně funguje jen v rychlostním režimu. Navíc se v tomto režimu kontroluje, jestli nedojde k překročení rychlosti motorů (např. při pohybu v blízkosti singulárních bodů) a pokud ano, je pohyb zpomalen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

### Vstup

<b>uAxesGroup</b>	Odkaz na skupinu os	Reference
<b>posvel</b>	Vektor požadované polohy nebo rychlosti	Reference
<b>der</b>	Vektor požadované derivace vstupu posvel	Reference
<b>TRACKP</b>	Sledování polohy	Bool

TRACKV Sledování rychlosti Bool

## Parametr

Velocity Maximální povolená rychlost [unit/s] Double (F64)  
 Acceleration Maximální povolené zrychlení [unit/s<sup>2</sup>] Double (F64)  
 Jerk Maximální povolená změna zrychlení [unit/s<sup>3</sup>] Double (F64)  
 CoordSystem Volba souřadného systému ⊙2 Long (I32)  
     1 ..... ACS  
     2 ..... MCS  
     3 ..... PCS  
     4 ..... TCS  
 LimitMode Význam limitů rychlosti/zrychlení/změny zrychlení ⊙1 Long (I32)  
     1 ..... Relativní (k defaultním hodnotám)  
     2 ..... Absolutní [unit/s unit/s<sup>2</sup> ...]  
 iLen Počet vzorků pro odhad rychlosti /zrychlení ↓-1 ↑99 Long (I32)

## Výstup

yAxesGroup Odkaz na skupinu os Reference  
 InTrack Příznak dosažení sledované polohy / rychlosti Bool  
 CommandAborted Příznak přerušení funkce bloku Bool  
 Busy Příznak, že algoritmus ještě neskončil Bool  
 Active Příznak, že blok řídí osu Bool  
 Error Příznak chyby Bool  
 ErrorID Výsledek poslední operace Error  
     i ..... kód chyby REXYGENu



## Kapitola 23

# CanDrv – Komunikace po sběrnici CAN

### Obsah

---

<a href="#">CanItem – Další přijatá zpráva sběrnice CAN</a>	790
<a href="#">CanRecv – Přijetí zprávy sběrnice CAN</a>	791
<a href="#">CanSend – Odeslání zprávy na sběrnici CAN</a>	793

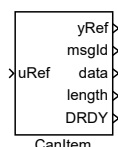
---

Knihovna CanDrv je určena pro zpracování komunikace po sběrnici CAN (Controller Area Network) v systému REXYGEN. Obsahuje [CanItem](#) pro správu datových položek CAN, [CanRecv](#) pro příjem zpráv ze sběrnice a [CanSend](#) pro odesílání zpráv. Tato knihovna poskytuje zásadní nástroje pro efektivní a spolehlivou komunikaci přes síť CAN, usnadňuje výměnu dat a řídicích příkazů mezi různými komponentami systému.

## CanItem – Další přijatá zpráva sběrnice CAN

Symbol bloku

Licence: [CANDRV](#)



### Popis funkce

Tento blok se používá ve spojení s blokem `CanRecv`. Vstup `uRef` tohoto bloku musí být připojen k výstupu `itemRef` bloku `CanRecv` (buď přímo nebo nepřímo připojením na výstup `yRef` již připojeného bloku).

Tento blok zobrazuje starší zprávy sběrnice CAN, které prošly filtrem v připojeném bloku `CanRecv`.

Pokud je k jednomu bloku `CanRecv` připojeno (přímo i nepřímo) více bloků `CanItem`, zprávy jsou zobrazovány podle pořadí vykonávání bloků `CanItem`, takže 1. blok `CanItem` zobrazuje předposlední přijatou zprávu (poslední je přímo v bloku `CanRecv`), 2. blok `CanItem` zobrazuje 3. od konce přijatou zprávu, atd. Proto se doporučuje připojovat vždy následující blok `CanItem` na `yRef` předcházejícího bloku `CanItem` aby bylo jasné pořadí.

Dokud nepřijde dostatek zpráv, blok zobrazuje na výstupech náhradní hodnoty `msgId` = -1 a `length` = -1.

Výstup `DRDY` = on pokud zobrazovaná zpráva (t.j. hodnoty na výstupech `msgId`, `data`, `length`) přišla po sběrnici CAN během poslední periody (t.j. po minulém spuštění bloku). V opačném případě je zpráva starší (tj. už zpracovaná a tedy neplatná), ale ponechává se na výstupech pro snazší kontrolu a ladění aplikace.

### Vstup

<code>uRef</code>	Odkaz na další přijaté packety	Reference
-------------------	--------------------------------	-----------

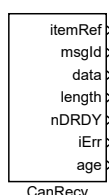
### Výstupy

<code>yRef</code>	Odkaz na další přijaté packety	Reference
<code>msgId</code>	Číslo zprávy (COB-ID)	Long (I32)
<code>data</code>	Data zprávy (max. 8 bajtů, nejnižší bajt první)	Large (I64)
<code>length</code>	Počet datových bajtů zprávy	↓0 ↑8 Long (I32)
<code>DRDY</code>	Přijata zpráva během poslední periody	Bool

## CanRecv – Přijetí zprávy sběrnice CAN

Symbol bloku

Licence: [CANDRV](#)



### Popis funkce

Blok slouží k přijetí zprávy na sběrnici CAN. Blok přijímá jen zprávy které odpovídají filtru (parametry `filterId`, `filterIdMask`, `filterLength`, `RTR`, `EXT`).

Počet zpráv, které prošly filtrem za periodu (tj. od minulého spuštění bloku) určuje výstup `nDRDY`.

Poslední přijatá zpráva je zobrazena na výstupech `msgId`, `data`, `length`. Starší zprávy (s ohledem na parametr `nmax`) jsou dostupné pomocí bloku `CanItem` navázaného na výstup `itemRef`.

Pro správnou funkci blok musí být napojen na ovladač `CanDrv`, který je v režimu `simpleCAN` (tj. `NodeMode=256`). To se provede pojmenováním bloku dle vzoru `<DRV>__<signal>` (stejně jako u bloků `Goto`, `OUTSTD`, `OUTQAD`, apod.), tj. název bloku musí začínat názvem driveru a dvěma podtržítky následované názvem signálu, přičemž název signálu může být v tomto případě libovolný.

Blok umožňuje přijímat zprávy s krátkým (11bitů) i dlouhým (29 bitů) číslem zprávy (řídí se parametrem `EXT`) a také zprávy pro vyžádání zprávy (parametr `RTR`). FD režim (který umožňuje zprávy s až 64 bajty) není podpořen.

### Parametry

<code>filterId</code>	Číslo zprávy, které jsou přijaty tímto blokem	↓0 ↑536870911	Long (I32)
<code>filterIdMask</code>	Označuje platné bity v parametru <code>filterId</code>	↓0 ↑536870911	Long (I32)
<code>filterLength</code>	Kolik bajtů dat musí mít zpráva, aby byla akceptována tímto blokem (-1 nefiltruje se podle délky dat)	↓-1 ↑8	Long (I32)
<code>RTR</code>	Příznak žádost o zprávu (RequestToSend)	⊙on	Bool
<code>EXT</code>	Rozšířený formát čísla zprávy (29bitů)	⊙on	Bool
<code>timeout</code>	Pokud během této doby nepříjde packet, je indikována chyba [s]	↓0.0	Double (F64)
<code>nmax</code>	Maximální počet zpráv přijatých blokem během jedné periody	↓1 ↑255	Long (I32)

## Výstupy

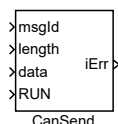
<code>itemRef</code>	Odkaz na další přijaté packety		Reference
<code>msgId</code>	Číslo zprávy (COB-ID)		Long (I32)
<code>data</code>	Data zprávy (max. 8 bajtů, nejnižší bajt první)		Large (I64)
		↓-9.22337E+18 ↑9.22337E+18	
<code>length</code>	Počet datových bajtů zprávy	↓0 ↑8	Long (I32)
<code>nDRDY</code>	Počet přijatých zpráv v aktuální periodě tasku	↑255	Word (U16)
<code>iErr</code>	Kód chyby		Error
<code>age</code>	Čas od poslední přijaté zprávy [s]	↓0.0	Double (F64)



## CanSend – Odeslání zprávy na sběrnici CAN

Symbol bloku

Licence: [CANDRV](#)



### Popis funkce

Blok slouží k odeslání zprávy po sběrnici CAN. Zpráva je určena pomocí vstupů `msgId`, `data`, `length` a parametrů `RTR`, `EXT`. Zpráva se odešle jen pokud je vstup `RUN = on`.

Pro správnou funkci blok musí být napojen na ovladač `CanDrv`, který je v režimu `simpleCAN` (tj. `NodeMode=256`). To se provede pojmenováním bloku dle vzoru `<DRV>__<signal>` (stejně jako u bloků `Goto`, `OUTSTD`, `OUTQAD`, apod.), tj. název bloku musí začínat názvem driveru a dvěma podtržítka následované názvem signálu, přičemž název signálu může být v tomto případě libovolný.

Blok umožňuje posílat zprávy s krátkým (11bitů) i dlouhým (29 bitů) číslem zprávy (řídí se parametrem `EXT`) a také vyžádat si poslání zprávy (parametr `RTR`). FD režim (který umožňuje zprávy s až 64 bajty) není podpořen.

### Vstupy

<code>msgId</code>	Číslo zprávy (COB-ID)	↓0 ↑536870911	Long (I32)
<code>length</code>	Počet datových bajtů zprávy	↓0 ↑8	Long (I32)
<code>data</code>	Data zprávy (max. 8 bajtů, nejnižší bajt první)	↓-9.22337E+18 ↑9.22337E+18	Large (I64)
<code>RUN</code>	Povolení odeslání zprávy		Bool

### Parametry

<code>RTR</code>	Příznak žádost o zprávu (RequestToSend)	⊙on	Bool
<code>EXT</code>	Rozšířený formát čísla zprávy (29bitů)	⊙on	Bool

### Výstup

<code>iErr</code>	Kód chyby	Error
-------------------	-----------	-------



## Kapitola 24

# OpcUaDrv – Komunikace pomocí OPC UA

### Obsah

---

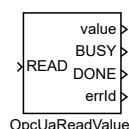
<a href="#">OpcUaReadValue – Čtení hodnoty protokolem OPC UA . . . . .</a>	<b>796</b>
<a href="#">OpcUaServerValue – Vystavení hodnoty v podobě OPC UA uzlu .</a>	<b>798</b>
<a href="#">OpcUaWriteValue – Zápis hodnoty protokolem OPC UA . . . . .</a>	<b>800</b>

---

Knihovna `OpcUaDrv` se specializuje na propojení se servery OPC UA (Open Platform Communications Unified Architecture) pro průmyslovou automatizaci. Skládá se ze tří klíčových bloků: `OpcUaReadValue`, `OpcUaServerValue` a `OpcUaWriteValue`. Blok `OpcUaReadValue` je určen pro čtení dat ze serverů, takže je stěžejní pro sběr dat v automatizovaných systémech. Blok `OpcUaWriteValue` umožňuje zápis dat na servery, což umožňuje řízení a provádění příkazů. Blok `OpcUaServerValue` navíc usnadňuje monitorování a správu hodnot serveru. Tato knihovna slouží jako kritický nástroj pro bezproblémovou komunikaci a interakci se servery OPC UA, čímž rozšiřuje možnosti automatizačních systémů.

## OpcUaReadValue – Čtení hodnoty protokolem OPC UA

Symbol bloku

Licence: [ADVANCED](#)

### Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál `OpCuaDrv` ovladače [14].

Blok `OpCuaReadValue` slouží pro čtení hodnoty OPC UA uzlu prostřednictvím spojení, které udržuje ovladač `OpCuaDrv` v módu OPC UA Klient.

První dva parametry bloku jsou `NodeId` a `NodeId_type`. `NodeId%type` určuje, jaký typ identifikátoru je očekáván v parametru `NodeId`. Pokud vybraný typ jedním z typů `string`, `numeric` nebo `guid`, pak by parametr `NodeId` měl obsahovat identifikátor OPC UA uzlu definovaného na serveru s prefixem indexu jmenného prostoru deklarovaného v konfiguraci ovladače odděleného dvojtečkou (např. `1:myNode`).

Pokud je vybrán typ `cesta`, pak by měl parametr `NodeId` obsahovat cestu k požadovanému uzlu ve stromové struktuře serveru. Každá část cesty se skládá z atributu `BrowseName` uzlu opět s prefixem indexu jmenného prostoru z konfigurace ovladače (např. `/1:myDevice/1:myNode`). Cesta je relativní ke složce `Objects` ve stromové struktuře serveru.

Parametr `type` definuje očekávaný datový typ OPC UA uzlu. Blok konvertuje atribut `value` uzlu na specifikovaný datový typ a nastaví hodnotu na svůj výstup `value` v případě úspěchu a nebo nastaví výstup `errId` na příslušný chybový kód.

### Vstup

READ	Povolení běhu algoritmu	Bool
------	-------------------------	------

### Parametry

NodeId	OPC UA Node Id	String
NodeId_type	Typ Node Id	⊙1 Long (I32)
	1 . . . . .	string
	2 . . . . .	numeric
	3 . . . . .	guid
	4 . . . . .	cesta

type	Očekávaný typ příchozích dat	⊙1 Long (I32)
	1 ..... Bool	
	2 ..... Byte (U8)	
	3 ..... Short (I16)	
	4 ..... Long (I32)	
	5 ..... Word (U16)	
	6 ..... DWord (U32)	
	7 ..... Float (F32)	
	8 ..... Double (F64)	
	10 .... Large (I64)	
	12 .... String	

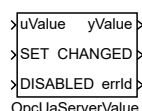
## Výstupy

value	Výstupní signál	Any
BUSY	Příznak probíhající operace	Bool
DONE	Příznak dokončení transakce	Bool
errId	Kód chyby	Error

## OpCuaServerValue – Vystavení hodnoty v podobě OPC UA uzlu

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál `OpCuaDrv` ovladače [14].

Blok `OpCuaServerValue` slouží pro vystavení OPC UA uzlu prostřednictvím `OpCuaDrv` ovladače v módu OPC UA Server.

První dva parametry bloku jsou `NodeId` a `NodeId_type`. `NodeId%type` určuje, jaký typ identifikátoru je očekáván v parametru `NodeId`. Parametr `NodeId` určuje identifikátor uzlu, pod kterým bude uzel vystavený na serveru.

Vstupní signál `DISABLE` určuje, zda bude uzel zveřejněn na serveru nebo ne. Pokud je vstupní signál `SET` nastaven na hodnotu `on`, hodnota ze vstupního signálu `uValue` je nastavena atributu `value` OPC UA uzlu.

Pokud je hodnota parametru `READONLY` nastavena na `off`, hodnota atributu `value` může být nastavena také prostřednictvím OPC UA protokolu z prostředí mimo algoritmus aplikace.

Výstupní signál `yValue` je nastavena v každém tiku na hodnotu OPC UA uzlu. Parametr `type` určuje datový typ atributu `value` OPC UA uzlu, datový typ vstupu `uValue` i datový typ výstupu `yValue`.

### Vstupy

<code>uValue</code>	Vstupní signál	Any
<code>SET</code>	Překopírování hodnoty vstupu do hodnoty OPC UA uzlu	Bool
<code>DISABLE</code>	Deaktivace OPC UA uzlu	Bool

### Parametry

<code>NodeId</code>	OPC UA Node Id	String
<code>NodeId_type</code>	Typ id OPC UA uzlu	⊙1 String
	1 ..... string	
	2 ..... numeric	
	3 ..... guid	

<b>type</b>	Datový typ hodnoty	⊙1	Long (I32)
	1 ..... Bool		
	2 ..... Byte (U8)		
	3 ..... Short (I16)		
	4 ..... Long (I32)		
	5 ..... Word (U16)		
	6 ..... DWord (U32)		
	7 ..... Float (F32)		
	8 ..... Double (F64)		
	10 .... Large (I64)		
	12 .... String		
<b>BrowseName</b>	Atribut 'browse name' OPC UA uzlu		String
<b>Description</b>	Popiska OPC UA uzlu		String
<b>DisplayName</b>	Zobrazované jméno uzlu OPC UA uzlu		String
<b>READONLY</b>	Nastavení hodnoty OPC uzlu jako pouze pro čtení	⊙on	Bool

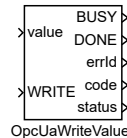
## Výstupy

<b>yValue</b>	Výstupní signál		Any
<b>CHANGED</b>	Indikace zápisu hodnoty přes OPC UA protocol		Bool
<b>errId</b>	Kód chyby		Error

## OpcUaWriteValue – Zápis hodnoty protokolem OPC UA

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál `OpCuaDrv` ovladače [14].

Blok `OpCuaWriteValue` slouží pro zápis hodnoty OPC UA uzlu prostřednictvím spojení, které udržuje ovladač `OpCuaDrv` v módu OPC UA Klient.

První dva parametry bloku jsou `NodeId` a `NodeId_type`. `NodeId_type` určuje, jaký typ identifikátoru je očekáván v parametru `NodeId`. Pokud vybraný typ jedním z typů **string**, **numeric** nebo **guid**, pak by parametr `NodeId` měl obsahovat identifikátor OPC UA uzlu definovaného na serveru s prefixem indexu jmenného prostoru deklarovaného v konfiguraci ovladače odděleného dvojtečkou (např. `1:myNode`).

Pokud je vybrán typ **cesta**, pak by měl parametr `NodeId` obsahovat cestu k požadovanému uzlu ve stromové struktuře serveru. Každá část cesty se skládá z atributu `BrowseName` uzlu opět s prefixem indexu jmenného prostoru z konfigurace ovladače (např. `/1:myDevice/1:myNode`). Cesta je relativní ke složce *Objects* ve stromové struktuře serveru.

Parametr `type` definuje očekávaný datový typ OPC UA uzlu. Vstupní signál `value` je převeden na zvolený datový typ a jeho hodnota je poté zapsána na atribut `value` OPC UA uzlu.

Po dokončení operace zápisu výsledný stavový kód operace definovaný standardem OPC UA je zapsán na výstup bloku `code` a jeho textová interpretace je nastavena na výstup bloku `status`.

### Vstupy

<code>value</code>	Vstupní signál	<code>Any</code>
<code>WRITE</code>	Povolení běhu algoritmu	<code>Bool</code>

### Parametry

<code>NodeId</code>	OPC UA Node Id	<code>String</code>
---------------------	----------------	---------------------



<b>NodeId_type</b>	Typ Node Id	⊙1	Long (I32)
	1 . . . . . string		
	2 . . . . . numeric		
	3 . . . . . guid		
	4 . . . . . cesta		
<b>type</b>	Datový typ posílané hodnoty	⊙1	Long (I32)
	1 . . . . . Bool		
	2 . . . . . Byte (U8)		
	3 . . . . . Short (I16)		
	4 . . . . . Long (I32)		
	5 . . . . . Word (U16)		
	6 . . . . . DWord (U32)		
	7 . . . . . Float (F32)		
	8 . . . . . Double (F64)		
	10 . . . . . Large (I64)		
	12 . . . . . String		

## Výstupy

<b>BUSY</b>	Příznak probíhající operace	Bool
<b>DONE</b>	Příznak dokončení transakce	Bool
<b>errId</b>	Kód chyby	Error
<b>code</b>	OPC UA výsledný stavový kód operace	DWord (U32)
<b>status</b>	OPC UA stavový kód v textové reprezentaci	String



## Příloha A

# Typy licencí

Z hlediska licencování existuje několik verzí runtime modulu **RexCore**, které poskytují maximální flexibilitu pro jednotlivé projekty. Níže uvedená tabulka porovnává jednotlivé varianty.

Funkční bloky systému **REXYGEN** jsou licencovány po skupinách. Funkční bloky ze skupiny **STANDARD** lze použít vždy, použití ostatních bloků je podmíněno aktivováním příslušné licence.

	RexCore DEMO	RexCore Starter	RexCore Plus	RexCore Professional	RexCore Ultimate
<i>Funkční bloky</i>					
STANDARD	●	●	●	●	●
ADVANCED	●	–	●	●	●
REXLANG	●	–	●	●	●
MOTION CONTROL	●	–	○	○	●
COORDINATED MOTION	●	–	○	○	●
AUTOTUNING	–	–	○	○	●
MATRIX	●	–	○	○	●
<i>I/O ovladače</i>					
Základní I/O ovladače	●	●	●	●	●
Další I/O ovladače	●	○	○	●	●

(● ... included, ○ ... optional, – ... not available)

Podrobné informace o licencování jednotlivých funkčních bloků jsou uvedeny v příloze [B](#).



## Příloha B

# Seznam funkčních bloků a jejich licencování

Aby bylo dosaženo maximální flexibility pro různé projekty, jsou funkční bloky systému REXYGEN licencovány po skupinách. Tabulka níže ukazuje příslušnost funkčních bloků k jednotlivým licenčním skupinám. Bližší informace o možnostech licencování jsou uvedeny v příloze A.

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ABS_	•	
ABSR0T		ADVANCED
ACD	•	
ADD	•	
ADDHEXD	•	
ADDOCT	•	
ADDQUAD	•	
AFLUSH	•	
ALB	•	
ALBI	•	
ALN	•	
ALNI	•	
ARS	•	
AND_	•	
ANDHEXD	•	
ANDOCT	•	
ANDQUAD	•	
ANLS	•	
ARC	•	
ARLY	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ASW		ADVANCED
ATMT	•	
AVG	•	
AVS		ADVANCED
BDHEXD	•	
BDOCT	•	
BINS	•	
BIS	•	
BISR	•	
BITOP	•	
BMHEXD	•	
BMOCT	•	
BPF	•	
CanItem		CANDRV
CanRecv		CANDRV
CanSend		CANDRV
CDELSSM		ADVANCED
CMP	•	
CNA	•	
CNB	•	
CNDR	•	
CNE	•	
CNI	•	
CNR	•	
CNS	•	
CONCAT	•	
COND		
COUNT	•	
CSSM		ADVANCED
DATE_	•	
DATETIME	•	
DDELSSM		ADVANCED
DEL	•	
DELM	•	
DER	•	
DFIR		ADVANCED
DIF_	•	
Display	•	
DIV	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
DSSM		ADVANCED
EAS	•	
EATMT		ADVANCED
EDGE_	•	
EKF		MODEL
EMD	•	
EPC		ADVANCED
EQ	•	
EVAR	•	
EXEC	•	
FIND	•	
FLCU		ADVANCED
FNX	•	
FNXY	•	
FOPDT	•	
FRID		ADVANCED
From	•	
GAIN	•	
GETPA	•	
GETPB	•	
GETPI	•	
GETPR	•	
GETPS	•	
Goto	•	
GotoTagVisibility	•	
GRADS		ADVANCED
HMI	•	
HTTP		ADVANCED
HTTP2		ADVANCED
I3PM		ADVANCED
IADD	•	
IDIV	•	
IMOD	•	
IMUL	•	
INFO	•	
INHEXD	•	
INOCT	•	
Inport	•	
INQUAD	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
INSTD	•	
INTE	•	
INTSM	•	
IODRV	•	
IOTASK	•	
ISSW	•	
ISUB	•	
ITOI	•	
ITOS	•	
KDER		ADVANCED
LC	•	
LEN	•	
LIN	•	
LLC	•	
LPBRK	•	
LPF	•	
MC_AccelerationProfile		MOTION CONTROL
MC_AddAxisToGroup		COORDINATED MOTION
MC_CamIn		MOTION CONTROL
MC_CamOut		MOTION CONTROL
MC_CombineAxes		MOTION CONTROL
MC_GearIn		MOTION CONTROL
MC_GearInPos		MOTION CONTROL
MC_GearOut		MOTION CONTROL
MC_GroupContinue		COORDINATED MOTION
MC_GroupDisable		COORDINATED MOTION
MC_GroupEnable		COORDINATED MOTION
MC_GroupHalt		COORDINATED MOTION
MC_GroupInterrupt		COORDINATED MOTION
MC_GroupReadActualAcceleration		COORDINATED MOTION
MC_GroupReadActualPosition		COORDINATED MOTION
MC_GroupReadActualVelocity		COORDINATED MOTION
MC_GroupReadError		COORDINATED MOTION
MC_GroupReadStatus		COORDINATED MOTION
MC_GroupReset		COORDINATED MOTION
MC_GroupSetOverride		COORDINATED MOTION
MC_GroupSetPosition		COORDINATED MOTION
MC_GroupStop		COORDINATED MOTION
MC_Halt		MOTION CONTROL

*Seznam pokračuje na další stránce...*



Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MC_HaltSuperimposed		MOTION CONTROL
MC_Home		MOTION CONTROL
MC_MoveAbsolute		MOTION CONTROL
MC_MoveAdditive		MOTION CONTROL
MC_MoveCircularAbsolute		COORDINATED MOTION
MC_MoveCircularRelative		COORDINATED MOTION
MC_MoveContinuousAbsolute		MOTION CONTROL
MC_MoveContinuousRelative		MOTION CONTROL
MC_MoveDirectAbsolute		COORDINATED MOTION
MC_MoveDirectRelative		COORDINATED MOTION
MC_MoveLinearAbsolute		COORDINATED MOTION
MC_MoveLinearRelative		COORDINATED MOTION
MC_MovePath		COORDINATED MOTION
MC_MovePath_PH		COORDINATED MOTION
MC_MoveRelative		MOTION CONTROL
MC_MoveSuperimposed		MOTION CONTROL
MC_MoveVelocity		MOTION CONTROL
MC_PhasingAbsolute		MOTION CONTROL
MC_PhasingRelative		MOTION CONTROL
MC_PositionProfile		MOTION CONTROL
MC_Power		MOTION CONTROL
MC_ReadActualPosition		MOTION CONTROL
MC_ReadAxisError		MOTION CONTROL
MC_ReadBoolParameter		MOTION CONTROL
MC_ReadCartesianTransform		COORDINATED MOTION
MC_ReadParameter		MOTION CONTROL
MC_ReadStatus		MOTION CONTROL
MC_Reset		MOTION CONTROL
MC_SetCartesianTransform		COORDINATED MOTION
MC_SetOverride		MOTION CONTROL
MC_Stop		MOTION CONTROL
MC_TorqueControl		MOTION CONTROL
MC_UngroupAllAxes		COORDINATED MOTION
MC_VelocityProfile		MOTION CONTROL
MC_WriteBoolParameter		MOTION CONTROL
MC_WriteParameter		MOTION CONTROL
MCP_AccelerationProfile		MOTION CONTROL
MCP_CamIn		MOTION CONTROL
MCP_CamTableSelect		MOTION CONTROL

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MCP_CombineAxes		MOTION CONTROL
MCP_GearIn		MOTION CONTROL
MCP_GearInPos		MOTION CONTROL
MCP_GroupHalt		COORDINATED MOTION
MCP_GroupInterrupt		COORDINATED MOTION
MCP_GroupSetOverride		COORDINATED MOTION
MCP_GroupSetPosition		COORDINATED MOTION
MCP_GroupStop		COORDINATED MOTION
MCP_Halt		MOTION CONTROL
MCP_HaltSuperimposed		MOTION CONTROL
MCP_Home		MOTION CONTROL
MCP_MoveAbsolute		MOTION CONTROL
MCP_MoveAdditive		MOTION CONTROL
MCP_MoveCircularAbsolute		COORDINATED MOTION
MCP_MoveCircularRelative		COORDINATED MOTION
MCP_MoveContinuousAbsolute		MOTION CONTROL
MCP_MoveContinuousRelative		MOTION CONTROL
MCP_MoveDirectAbsolute		COORDINATED MOTION
MCP_MoveDirectRelative		COORDINATED MOTION
MCP_MoveLinearAbsolute		COORDINATED MOTION
MCP_MoveLinearRelative		COORDINATED MOTION
MCP_MovePath		COORDINATED MOTION
MCP_MovePath_PH		COORDINATED MOTION
MCP_MoveRelative		MOTION CONTROL
MCP_MoveSuperimposed		MOTION CONTROL
MCP_MoveVelocity		MOTION CONTROL
MCP_PhasingAbsolute		MOTION CONTROL
MCP_PhasingRelative		MOTION CONTROL
MCP_PositionProfile		MOTION CONTROL
MCP_SetCartesianTransform		COORDINATED MOTION
MCP_SetKinTransform_Arm		COORDINATED MOTION
MCP_SetKinTransform_Schunk		COORDINATED MOTION
MCP_SetKinTransform_UR		COORDINATED MOTION
MCP_SetOverride		MOTION CONTROL
MCP_Stop		MOTION CONTROL
MCP_TorqueControl		MOTION CONTROL
MCP_VelocityProfile		MOTION CONTROL
MCU	•	
MDL	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MDLI	•	
MID	•	
MINMAX	•	
MODULE	•	
MP	•	
MqttPublish		MQTTDRV
MqttSubscribe		MQTTDRV
MUL	•	
MVD	•	
NOT_	•	
NSCL	•	
NSSM		MODEL
OpcUaReadValue		ADVANCED
OpcUaServerValue		ADVANCED
OpcUaWriteValue		ADVANCED
OR_	•	
ORHEXD	•	
OROCT	•	
ORQUAD	•	
OSD	•	
OSCALL	•	
OUTHEXD	•	
OUTOCT	•	
Outport	•	
OUTQUAD	•	
OUTRHEXD		ADVANCED
OUTROCT		ADVANCED
OUTRQUAD		ADVANCED
OUTRSTD		ADVANCED
OUTSTD	•	
PARA	•	
PARB	•	
PARE	•	
PARI	•	
PARR	•	
PARS	•	
PGAVR		
PGBAT		
PGBUS		

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
PGCB		
PGENG		
PGGEN		
PGGS		
PGINV		
PGLOAD		
PGMAINS		
PGSENS		
PGSG		
PGSIM		
PGSOLAR		
PGWIND		
PIDAT		AUTOTUNING
PIDE		ADVANCED
PIDGS		ADVANCED
PIDMA		AUTOTUNING
PIDU	•	
PIDUI		ADVANCED
PJROCT	•	
PJSEXOCT	•	
PJSEXOCT	•	
PJSOCT	•	
POL	•	
POUT	•	
PRBS	•	
PRGM	•	
PROJECT	•	
PSMPC		ADVANCED
PWM	•	
PYTHON		REXLANG
QFC		ADVANCED
QFD		ADVANCED
QTASK	•	
QP_MPC2QP		ADVANCED
QP_UPDATE		ADVANCED
QP_OASES		ADVANCED
QCEDPOPT		ADVANCED
RDC		ADVANCED
REC	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
REGEXP		ADVANCED
REL	•	
REPLACE	•	
REXLANG		REXLANG
RLIM	•	
RLY	•	
RM_AxesGroup		COORDINATED MOTION
RM_Axis		MOTION CONTROL
RM_AxisOut		MOTION CONTROL
RM_AxisSpline		MOTION CONTROL
RM_DirectTorque		MOTION CONTROL
RM_DirectVelocity		MOTION CONTROL
RM_DriveMode		MOTION CONTROL
RM_Feed		COORDINATED MOTION
RM_Gcode		COORDINATED MOTION
RM_GroupTrack		COORDINATED MOTION
RM_HomeOffset		MOTION CONTROL
RM_Track		MOTION CONTROL
RS	•	
RTOI	•	
RTOS	•	
RTOV	•	
S_AND		
S_BC		
S_CMP		
S_CTS		
S_LB		
S_NOT		
S_OR		
S_PULS		
S_PV		
S_RS		
S_SEL		
S_SELVAL		
S_SR		
S_SUMC		
S_TDE		
S_TDR		
S_TLATCH		

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
S_VALB		
S_VALC		
S10F2		ADVANCED
SAI		ADVANCED
SAT	•	
SC2FA		AUTOTUNING
SCU	•	
SCUV	•	
SEL	•	
SELHEXD	•	
SELOCT	•	
SELQUAD	•	
SELSOCT	•	
SELU	•	
SETPA	•	
SETPB	•	
SETPI	•	
SETPR	•	
SETPS	•	
SG	•	
SGI	•	
SGSLP		ADVANCED
SHIFTOCT	•	
SHLD	•	
SILO	•	
SILOS	•	
SINT	•	
SLEEP	•	
SMHCC		ADVANCED
SMHCCA		AUTOTUNING
SMTF		ADVANCED
SOPDT	•	
SPIKE		ADVANCED
SQR	•	
SQRT_	•	
SR	•	
SRTF		ADVANCED
SSW	•	
STEAM	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
STOR	•	
SUB	•	
SubSystem	•	
SWR	•	
SWU	•	
SWVMR	•	
TASK	•	
TIME	•	
TIMER_	•	
TIODRV	•	
TRND	•	
TRNDV	•	
TSE	•	
UTOI	•	
VDEL	•	
VIN		ADVANCED
VOUT		ADVANCED
VTOR	•	
WASM		REXLANG
WSCH	•	
WWW	•	
ZV4IS		ADVANCED
DFIR		ADVANCED
PGSIM		
PGMAINS		
PGBUS		
PGLOAD		
PGGEN		
PGCB		
PGSENS		
PGENG		
PGAVR		
PGSG		
PGINV		
PGSOLAR		
PGWIND		
PGBAT		
PGGS		
CanSend		CANDRV

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
CanRecv		CANDRV
CanItem		CANDRV
MqttPublish		MQTTDRV
MqttSubscribe		MQTTDRV
EKF		MODEL
NSSM		MODEL
RM_HomeOffset		MOTION CONTROL
PARE	•	
EQ	•	
PYTHON		REXLANG
WASM		REXLANG
RM_DriveMode		MOTION CONTROL
RM_DirectTorque		MOTION CONTROL
RM_DirectVelocity		MOTION CONTROL
COND		
TESTS		
S_CMPT		
S_RCK		
S_POR		
OpcUaReadValue		
OpcUaWriteValue		
OpcUaServerValue		
STEAM	•	
PJSEXOCT	•	
BISR	•	
DP2M		
MBAL		
MOFN		
TB1		
TB2		
TB3		
TB6		
VAC		
OSD	•	
CNT	•	
CNDT	•	
CONCAT_DT	•	
SPLIT_DT	•	
STR2DT	•	

*Seznam pokračuje na další stránce...*



Jméno bloku	Potřebná licence	
	STANDARD	Jiná
DT2STR	•	
WEEK	•	
T2STR	•	
TZ2UTC	•	
UTC2TZ	•	
SYSLOG	•	
SYSEVENT	•	
ALM	•	
ALARMS	•	
TRIM	•	



## Příloha C

# Chybové kódy systému REXYGEN

### Kódy úspěšných operací

- 0 ..... V pořádku
- 1 ..... Nepravda
- 2 ..... První hodnota je větší
- 3 ..... Druhá hodnota je větší
- 4 ..... Parametr byl změněn
- 5 ..... V pořádku, na serveru neprovedena žádná transakce
- 6 ..... Příliš velká hodnota
- 7 ..... Příliš malá hodnota
- 8 ..... Operace probíhá
- 9 ..... Upozornění ovladače systému REXYGEN
- 10 ..... V archivu nejsou další položky
- 11 ..... Položka je pole
- 12 ..... Ukončeno
- 13 ..... Konec souboru
- 14 ..... Parametr pravděpodobně nesprávný

### Obecné chybové kódy

- 100 .... Nedostatek paměti
- 101 .... Předpoklad nesplněn (Assertion failure)
- 102 .... Překročení času (timeout)
- 103 .... Obecná chyba vstupní proměnné
- 104 .... Nesprávná verze konfigurace
- 105 .... Není implementováno
- 106 .... Nesprávný parametr
- 107 .... Chyba služeb COM/OLE
- 108 .... Chyba modulu systému REXYGEN - některý ovladač nebo blok není nainstalován nebo licencován
- 109 .... Chyba ovladače systému REXYGEN

- 110 .... Úlohu operačního systému se nepodařilo vytvořit
- 111 .... Chyba volání funkce operačního systému
- 112 .... Nesprávná verze operačního systému
- 113 .... Přístup odmítnut operačním systémem
- 114 .... Perioda bloku nebyla nastavena
- 115 .... Selhala inicializace
- 116 .... Probíhá výměna konfigurace systému REXYGEN
- 117 .... Nesprávné cílové zařízení konfigurace
- 118 .... Přístup odmítnut systémem REXYGEN
- 119 .... Blok nebo jiný objekt není nainstalován nebo licencován
- 120 .... Kontrolní součty se liší
- 121 .... Objekt již existuje
- 122 .... Objekt neexistuje
- 123 .... Systémový uživatel nemá přiřazenou žádnou skupinu řídicího systému REXY-  
GEN
- 124 .... Špatné heslo
- 125 .... Špatné uživatelské jméno nebo heslo
- 126 .... Cílové zařízení není kompatibilní
- 127 .... Zdroj nelze použít, neboť je uzamčen jiným modulem
- 128 .... Text není platný v kódování UTF8
- 129 .... Spuštění exekutivy není povoleno
- 130 .... Dosaženo k překročení maximálního počtu nějakého objektu
- 131 .... Došlo ke oříznutí textu
- 132 .... Nedostatečný buffer pro požadovanou operaci
- 133 .... Vykonávání bloku pozastaveno kvůli běhové chybě

#### Registrace tříd, chybové kódy symbolů a validačních procedur

- 200 .... Neregistrovaná třída
- 201 .... Třída už byla registrována
- 202 .... Nedostatek místa v registru
- 203 .... Index registru mimo rozsah
- 204 .... Nesprávný kontext
- 205 .... Nesprávný identifikátor
- 206 .... Nesprávný příznak vstupu
- 207 .... Nesprávná maska vstupu
- 208 .... Nesprávný druh objektu
- 209 .... Nesprávný typ proměnné
- 210 .... Nesprávný pracovní prostor objektu
- 211 .... Symbol nebyl nalezen
- 212 .... Symbol je nejednoznačný
- 213 .... Chyba kontroly rozsahu
- 214 .... Nedostatek místa pro hledání
- 215 .... Zápis do proměnné určené pouze pro čtení není dovolen
- 216 .... Data nejsou připravena

- 217 .... Hodnota mimo přípustný rozsah
- 218 .... Chyba připojení vstupu
- 219 .... Nalezena smyčka typů UNKNOWN
- 220 .... Chyba při překladu jazyka REXLANG

### Kódy pro streamy a souborový systém

- 300 .... Přetečení streamu
- 301 .... Podtečení streamu
- 302 .... Vysílací chyba streamu
- 303 .... Přijímací chyba streamu
- 304 .... Chyba při posílání dat na cílové zařízení (download)
- 305 .... Chyba při posílání dat z cílového zařízení (upload)
- 306 .... Chyba vytvoření souboru
- 307 .... Chyba otvírání souboru
- 308 .... Chyba zavření souboru
- 309 .... Chyba čtení souboru
- 310 .... Chyba zápisu do souboru
- 311 .... Nesprávný formát
- 312 .... Chyba při komprimaci souborů
- 313 .... Chyba během extrahování souborů

### Chyby komunikace

- 400 .... Chyba síťové komunikace
- 401 .... Komunikace už byla inicializována
- 402 .... Komunikace úspěšně ukončena
- 403 .... Nečekané zavření komunikace
- 404 .... Neznámý příkaz
- 405 .... Neočekávaný příkaz
- 406 .... Nečekané zavření komunikace, pravděpodobně 'příliš mnoho klientů'
- 407 .... Překročení časového limitu pro komunikaci (timeout)
- 408 .... Cílové zařízení nebylo nalezeno
- 409 .... Spojení selhalo
- 410 .... Konfigurace systému REXYGEN byla změněna
- 411 .... Běh exekutivy systému REXYGEN se ukončuje
- 412 .... Běh exekutivy systému REXYGEN byl ukončen
- 413 .... Spojení odmítnuto
- 414 .... Cílové zařízení není dostupné
- 415 .... Cílové zařízení nebylo nalezeno v záznamu DNS
- 416 .... Chyba při čtení ze soketu
- 417 .... Chyba zápisu do soketu
- 418 .... Chybná operace na soketu
- 419 .... Rezervováno pro soket 1
- 420 .... Rezervováno pro soket 2
- 421 .... Rezervováno pro soket 3

- 422 .... Rezervováno pro soket 4
- 423 .... Rezervováno pro soket 5
- 424 .... Nelze vytvořit kontext SSL
- 425 .... Nelze načíst certifikát
- 426 .... Chyba při vyjednávání spojení SSL
- 427 .... Chyba verifikace certifikátu
- 428 .... Rezervováno pro SSL 2
- 429 .... Rezervováno pro SSL 3
- 430 .... Rezervováno pro SSL 4
- 431 .... Rezervováno pro SSL 5
- 432 .... Relace odmítnuta
- 433 .... STARTTLS odmítnuto
- 434 .... Ověřovací metoda odmítnuta
- 435 .... Ověření selhalo
- 436 .... Chyba operace vysílání
- 437 .... Chyba operace přijímání
- 438 .... Komunikační příkaz selhal
- 439 .... Vyrovňovací paměť pro příjem je příliš malá
- 440 .... Vyrovňovací paměť pro vysílání je příliš malá
- 441 .... Špatná hlavička
- 442 .... Server HTTP vrátil chybu
- 443 .... Server HTTP vrátil přesměrování
- 444 .... Nepřípustná blokuující operace
- 445 .... Neplatná operace
- 446 .... Komunikace ukončena
- 447 .... Připojování přerušeno

#### Kódy numerických chyb

- 500 .... Obecná numerická chyba
- 501 .... Dělení nulou
- 502 .... Přetečení numerického zásobníku
- 503 .... Neplatná numerická instrukce
- 504 .... Neplatná numerická adresa
- 505 .... Nesprávný numerický typ
- 506 .... Neinicializovaná numerická hodnota
- 507 .... Přetečení/podtečení numerického argumentu
- 508 .... Numerická chyba kontroly rozsahu
- 509 .... Nesprávný rozsah indexů vektoru/matice
- 510 .... Číselná hodnota příliš blízká nule

#### Kódy archivního systému

- 600 .... Chyba prohledávání archivu
- 601 .... Fatální chyba archivního semaforu
- 602 .... Archiv byl smazán

- 603 .... Archiv byl rekonstruován ze záložních proměnných
- 604 .... Archiv byl rekonstruován z normálních proměnných
- 605 .... Chyba kontrolního součtu archivu
- 606 .... Chyba integrity archivu
- 607 .... Byla změněna velikost archivu
- 608 .... Byla překročena povolená velikost archivu

### Kódy bloků pro řízení pohybu

- 700 .... MC - Neplatný parametr
- 701 .... MC - Mimo rozsah
- 702 .... MC - Pozice není dosažitelná
- 703 .... MC - Neplatný stav osy
- 704 .... MC - Překročen limit momentu
- 705 .... MC - Překročen časový limit
- 706 .... MC - Překročena hraniční pozice
- 707 .... MC - Skoková změna pozice nebo rychlosti
- 708 .... MC - Base axis error or invalid state
- 709 .... MC - Pohyb zastaven vstupem FAULT
- 710 .... MC - Pohyb zastaven polohou mimo rozsah osy
- 711 .... MC - Pohyb zastaven z důvodu překročení maximální rychlosti osy
- 712 .... MC - Pohyb zastaven z důvodu překročení maximálního zrzchlení osy
- 713 .... MC - Pohyb zastaven koncovým spínačem
- 714 .... MC - Pohyb zastaven z důvodu překročení maximální odchylky polohy (LAG)
- 715 .... MC - Osa deaktivována během pohybu
- 716 .... MC - Chyba generování přechodové křivky
- 717 .... MC - Chyba servoměniče
- 718 .... MC - nepoužito
- 719 .... MC - nepoužito
- 720 .... MC - Obecná chyba
- 721 .... MC - Není implementováno
- 722 .... MC - Příkaz ukončen
- 723 .... MC - Rozdílná perioda osy a bloku
- 724 .... MC - Blok čeká na převzetí osy

### Kódy licencovacího systému

- 800 .... Nepodařila se identifikace síťového rozhraní
- 801 .... Nepodařila se identifikace CPU
- 802 .... Nepodařila se identifikace HDD
- 803 .... Neplatný kód zařízení
- 804 .... Neplatný licenční klíč
- 805 .... Licence nenalezena

### Kódy spojené s webserverem

- 900 .... Příliš rozsáhlý požadavek na webový server
- 901 .... Příliš rozsáhlá odpověď webového serveru
- 902 .... Neplatný formát
- 903 .... Neplatný parametr

### Kódy spojené s knihovnou RexVision

- 1000 ... Výsledek není vyhodnocen
- 1001 ... Nelze nalézt hledaný objekt/vzor
- 1002 ... Zadanému kritériu hledání vyhovuje více objektů

### Kódy spojené se standardem FMI

- 1100 ... Nepodařilo se alokovat kontext FMI
- 1101 ... Nesprávná verze FMU
- 1102 ... Chyba parsování souboru XML pro FMI
- 1103 ... Vyžadován pouze druh FMI pro Model Exchange
- 1104 ... Vyžadován pouze druh FMI pro Co-Simulation
- 1105 ... Nepodařilo se zavést FMU
- 1106 ... Nepodařilo se vytvořit instanci FMU
- 1107 ... Nepodařilo se ukončit instanci FMU
- 1108 ... Selhal reset FMU
- 1109 ... Selhalo nastavení experimentu FMU
- 1110 ... Selhalo zahájení inicializačního módu FMU
- 1111 ... Selhalo ukončení inicializačního módu FMU
- 1112 ... Chyba získání seznamu proměnných FMU
- 1113 ... Chyba čtení reálné proměnné z FMU
- 1114 ... Chyba zápisu reálné proměnné do FMU
- 1115 ... Chyba čtení celočíselné proměnné z FMU
- 1116 ... Chyba zápisu celočíselné proměnné do FMU
- 1117 ... Chyba čtení booleovské proměnné z FMU
- 1118 ... Chyba zápisu booleovské proměnné do FMU
- 1119 ... Chyba provedení simulačního kroku FMU
- 1120 ... FMU má příliš mnoho vstupů
- 1121 ... FMU má příliš mnoho výstupů
- 1122 ... FMU má příliš mnoho parametrů



## Příloha D

# Speciální signály systému REXYGEN

Dále je uveden seznam speciálních signálů, které lze číst uvnitř REXYGENu. Podrobnosti o jejich použití najdete v příkladu *0001\_Special\_Signals*.

Parameter	Popis	Data Type	Možnosti
perf	Frekvence výkonnostního čítače	LARGE	EXEC
period	Perioda	DOUBLE	LEVEL, TASK, SEQ, IODRV, QTASK, BLOCK, ARCHIVE
nblocks	Počet bloků	SHORT	TASK, SEQ, QTASK
stack	Velikost zásobníku	LONG	TASK, IODRV, QTASK
exfac	Faktor provedení tasku	DWORD	TASK, IODRV, QTASK
start	Start tick	DWORD	TASK
stop	Stop tick	DWORD	TASK
ntasks	Počet tasků	SHORT	LEVEL, IODRV, EXEC
ntick	Počet ticků levelu	DWORD	LEVEL
pri	Priorita levelu	SHORT	LEVEL, IODRV, QTASK
tick	Hlavní tick	LARGE	EXEC
nlevels	Počet levelů	SHORT	EXEC
nmodules	Počet modulů	SHORT	EXEC
ndrivers	Počet ovladačů	SHORT	EXEC
narchives	Počet archivů	SHORT	EXEC
nqtasks	Počet rychlých tasků	SHORT	EXEC
tcomp	Čas kompilace [ns od epochy]	LARGE	EXEC
tdnld	Čas downloadu [ns od epochy]	LARGE	EXEC
bufsize	Velikost bufferu (archiv)	LONG	ARCHIVE
timesize	Velikost indexového bufferu (archiv)	LONG	ARCHIVE
daysize	Velikost denního bufferu (archiv)	LARGE	ARCHIVE
errblk	Index bloku s chybou	SHORT	TASK, SEQ, QTASK
errno	Kód chyby	SHORT	TASK, SEQ, QTASK, IODRV
status	Kód stavu (ovladač)	LONG	IODRV
over	Počet překryvů/kolizí (qtask)	LARGE	QTASK, TASK, IODRV
excnt	Počet spuštění tasku	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tlast	Číslo posledního provedení [ticky výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tmin	Minimální čas provedení [ticky výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tmax	Maximální čas provedení [ticky výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tsum	Součet provedení [ns nebo ticky]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tavg	Průměrný čas provedení (tsum/excnt)[ticky výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
dstart	Časový údaj o zahájení [ticky výkonnostního čítače]	LARGE	IODRV, TASK
dstop	Časový údaj o ukončení [ticky výkonnostního čítače]	LARGE	IODRV, TASK
tover	Overtime [ticky výkonnostního čítače]	LARGE	TASK
dmstart	Maximální start delay [ticky výkonnostního čítače]	LARGE	IODRV, TASK
dmstop	Maximální stop delay [ticky	LARGE	IODRV, TASK

# Literatura

- [1] OPC Foundation. *Data Access Custom Interface Specification Version 3.00*. OPC Foundation, P.O. Box 140524, Austin, Texas, USA, 2003.
- [2] Schlegel Miloš. Fuzzy regulátor: tutoriál.
- [3] Miloš Schlegel, Pavel Balda, and Milan Štětina. Robustní PID autotuner: momentová metoda. *Automatizace*, 46(4):242–246, 2003.
- [4] M. Schlegel and P. Balda. Diskretizace spojitého lineárního systému (in Czech). *Automatizace*, 11, 1987.
- [5] J. Königsmarková and M. Schlegel. Identification of n-link inverted pendulum on a cart. <https://ieeexplore.ieee.org/document/7976186>, 2017.
- [6] Python Software Foundation. Python documentation. <https://docs.python.org/3/>, 2019.
- [7] Martin Reinecke. PocketFFT. <https://gitlab.mpcdf.mpg.de/mtr/pocketfft>, 2019.
- [8] Paul N. Swarztrauber. FFTPACK Version 4 – netlib.org. <https://www.netlib.org/fftpack>, 1985.
- [9] Paul N. Swarztrauber. Vectorizing the ffts\*\*this chapter was written while the author was visiting the scientific computing division at the national bureau of standards. In GARRY RODRIGUE, editor, *Parallel Computations*, pages 51–83. Academic Press, 1982.
- [10] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.
- [11] Wikipedia. Welch’s method. [https://en.wikipedia.org/wiki/Welch’s\\_method](https://en.wikipedia.org/wiki/Welch’s_method).
- [12] REX Controls s.r.o.. *Ovladač MQTTDrv systému REXYGEN – Uživatelská příručka*, 2020. →.
- [13] OASIS. MQTT Version 3.1.1. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>, 2014.

- [14] *Ovladač OpcUaDrv systému REXYGEN – Uživatelská příručka.*

# Rejstřík

- úloha
  - doba provádění, 57
  - priorita, 57
  - rychlá, 46
  - standardní, 57
- čítání pulsů
  - obousměrné, 275
- časování
  - řízení, 299
- časovač
  - systémový, 41
  - týdenní, 302
- šířka pásma, 150
- šířková modulace, 235
- TODO
  - SRTF DGLOG, 49
- ABS, 18, 25, 86, 87
- ABS\_, 805
- ABSR0T, 25, 129, 805
- ACD, 315, 805
- ACLEAR, 323
- ADD, 18, 25, 86, 88, 89, 805
- ADDHEXD, 25, 89, 805
- ADDOCT, 88, 89, 124, 805
- ADDQUAD, 89, 805
- AFLUSH, 322, 805
- alarm
  - číselná hodnota, 310
  - logická hodnota, 307
- ALARMS, 29, 309, 310, 817
- alarmy, 309
- ALB, 29, 307, 805
- ALBI, 29, 307, 805
- ALM, 29, 309, 817
- ALMI, 29, 309
- ALN, 29, 310, 805
- ALNI, 29, 310, 805
- AND, 18, 25, 263–265
- AND\_, 805
- ANDHEXD, 25, 264, 265, 805
- ANDOCT, 264, 265, 805
- ANDQUAD, 264, 265, 805
- ANLS, 18, 183, 184, 805
- aplikace
  - řídícího systému REXYGEN, 34
- ARC, 32, 35, 308, 312, 316, 319, 321–323, 805
- architektura
  - otevřená, 43
- archiv, 32, 306
  - alarmů, 32
  - konfigurace, 32
  - na disku, 306
  - trendů, 32
  - událostí, 32
  - v paměti RAM, 306
  - v zálohované paměti, 306
- archivace
  - delta kritérium, 315
- ARLY, 197, 805
- ARS, 313, 805
- ASW, 18, 25, 128, 131, 806
- ATMT, 18, 25, 263, 267, 277, 346, 356, 360, 806
- AVG, 18, 25, 128, 133, 806
- AVS, 25, 134, 806
- AVSI, 25, 135
- běh úloh, 48
- BDHEXD, 25, 270, 277, 806
- BDOCT, 18, 263, 270, 277, 806
- Besselův filtr, 150

- binární posloupnost
  - generátor, 186, 188
- BINS, 18, 183, 186, 806
- BIS, 18, 183, 188, 190, 806
- BISR, 18, 183, 189, 806, 816
- BITOP, 18, 25, 125, 263, 272, 806
- blok
  - formát popisu, 22
  - komunikační, 475
  - parametry, 23
  - popis funkce, 22
  - symbol, 22
  - výstupu, 22
  - volně programovatelný, 480
  - volně programovatelný v jazyce Python, 508
  - vstupy, 22
- BMHEXD, 25, 273, 277, 806
- BMOCT, 18, 263, 273, 277, 806
- BPF, 18, 25, 128, 138, 806
- BSFIFO, 20, 515, 516
- BSGET, 20, 515, 518, 520, 521
- BSGETOCT, 518, 521
- BSGETOCTV, 520, 522
- BSGETV, 20, 515, 520, 522
- BSSET, 20, 515, 518, 521
- BSSETOCT, 518, 521
- BSSETOCTV, 520, 522
- BSSETV, 20, 515, 520, 522
- Butterworthův filtr, 150
  
- CanItem, 22, 789, 790, 806, 816
- CanRecv, 22, 789, 791, 806, 816
- CanSend, 22, 789, 793, 806, 815
- CDELSSM, 19, 368, 369, 806
- cesta
  - úplná, 48
- chyba
  - fatální, 46
- CMP, 25, 139, 806
- CNA, 400, 806
- CNB, 25, 91, 806
- CNDR, 25, 140, 806
- CNDT, 816
- CNE, 92, 806
- CNI, 25, 93, 806
- CNR, 25, 94, 806
- CNS, 19, 325, 326, 806
- CNT, 816
- CONCAT, 19, 325, 327, 806
- CONCAT\_DT, 816
- COND, 806, 816
- COUNT, 18, 25, 263, 275, 806
- CSSM, 19, 367, 372, 806
  
- DATE, 19, 295, 296
- DATE\_, 297, 806
- DATETIME, 19, 295–297, 301, 806
- DDELSSM, 19, 368, 374, 806
- DEL, 18, 25, 26, 128, 142, 806
- DELM, 25, 143, 806
- delta kritérium, 315
- DER, 25, 144, 806
- DFIR, 19, 367, 376, 806, 815
- DIF, 25, 95
- DIF\_, 806
- Display, 64, 806
- DIV, 18, 25, 26, 86, 96, 806
- doba
  - provádění úlohy, 57
- dopravní zpoždění, 382, 394
- DP2M, 816
- DSSM, 19, 367, 377, 807
- DT2STR, 817
  
- EAS, 25, 97, 807
- EATMT, 25, 277, 807
- EDGE, 25, 168, 280
- EDGE\_, 807
- EKF, 19, 367, 379, 392, 807, 816
- EMD, 25, 98, 807
- EPC, 20, 44, 467, 468, 807
- EQ, 25, 281, 807, 816
- EVAR, 25, 145, 807
- EXEC, 17, 28, 32, 34, 39–41, 43, 46, 47, 57–60, 154, 299, 807
- exekutiva
  - reálného času, 34

- externí program, 468
- FFT, 20, 515, 523
- filtr
  - šířka pásma, 150
  - Besselův, 150
  - Butterworthův, 150
  - nelineární, 173
- filtrace
  - číslíková vstupních signálů, 46
- FIND, 19, 325, 328, 807
- FLCU, 18, 196, 198, 807
- FMUCS, 19, 367
- FMUINFO, 19, 367
- FNX, 18, 25, 86, 100, 807
- FNXY, 18, 25, 86, 102, 807
- FOPDT, 19, 367, 382, 807
- Fourierova transformace
  - rychlá, 523
- frekvenční charakteristika, 210
- FRID, 200, 807
- From, 65, 67–69, 807
- funkce
  - operačního systému, 44
- GAIN, 25, 104, 807
- generátor
  - časových funkcí, 229
  - binární posloupnosti, 186, 188
  - po částech lineární funkce, 184
  - signálu, 193
- GETPA, 19, 70, 343, 344, 347, 354, 807
- GETPB, 346, 807
- GETPI, 346, 488, 489, 807
- GETPR, 346, 360, 807
- GETPS, 19, 343, 348, 807
- Goto, 65–67, 69, 807
- GotoTagVisibility, 68, 69, 807
- GRADS, 105, 807
- hierarchie, 72
- HMI, 17, 28, 37, 807
- hodnota
  - implicitní, 23
  - maximální, 23
  - minimální, 23
  - náhradní, 96, 98, 100, 102, 109, 110, 118, 123
- HTTP, 471, 807
- HTTP2, 20, 467, 473, 807
- I3PM, 202, 807
- IADD, 18, 25, 86, 107, 807
- identifikace
  - modelu se třemi parametry, 202
- IDIV, 18, 25, 86, 109, 807
- IMOD, 18, 25, 86, 110, 807
- IMUL, 18, 25, 86, 111, 807
- INFO, 38, 807
- INHEXD, 74, 807
- inicializace
  - pořadí modulů, 39
  - pořadí ovladačů, 39
  - rychlé úlohy, 46
- INOCT, 74, 807
- Inport, 70, 72, 347, 357, 637, 807
- INQUAD, 74, 807
- INSTD, 65, 74, 76, 808
- INTE, 25, 146, 172, 808
- INTSM, 25, 125, 282, 808
- IODRV, 35, 39, 65, 67, 808
- IOTASK, 41, 49, 59, 344, 346, 355, 356, 369, 372, 808
- IPEN2, 383, 386
- IPEN2pu, 383, 386
- IPEN3, 383, 386
- IPEN3pu, 383, 386
- ISSW, 25, 283, 808
- ISUB, 25, 113, 808
- ITOI, 25, 284, 808
- ITOS, 19, 325, 329, 808
- jmenovatel, 98
- KDER, 25, 148, 808
- kompatibilia
  - REXYGEN a Simulink, 42
- kompenzátor
  - derivační, 204
  - integračně-derivační, 205

- komprese, 315
- konfigurace
  - archivy, 34
  - moduly, 34
  - systému REXYGEN, 34
  - výpočetní úloha, 34
  - vstupně-výstupní ovladače, 34
- krokový regulátor, 247, 250
  
- LC, 204, 808
- LEN, 19, 325, 330, 808
- LIN, 18, 25, 86, 115, 808
- LLC, 205, 808
- LPBRK, 42, 131, 808
- LPF, 25, 150, 808
- LPI, 206
  
- MB\_DASUM, 401
- MB\_DAXPY, 402
- MB\_DCOPY, 403
- MB\_DDOT, 404
- MB\_DGEMM, 20, 405
- MB\_DGEMV, 406
- MB\_DGER, 20, 407
- MB\_DNRM2, 408
- MB\_DROT, 409
- MB\_DSCAL, 410
- MB\_DSWAP, 411
- MB\_DTRMM, 20, 412
- MB\_DTRMV, 413
- MB\_DTRSV, 414
- MBAL, 816
- MC\_AccelerationProfile, 21, 540, 569, 808
- MC\_AddAxisToGroup, 21, 685, 713, 776, 779, 808
- MC\_CamIn, 21, 644, 648, 659, 663, 678, 681, 808
- MC\_CamOut, 21, 644, 659, 663, 808
- MC\_CombineAxes, 21, 643, 665, 808
- MC\_CombineAxis, 676, 678, 681
- MC\_GearIn, 21, 643, 668, 671, 676, 678, 681, 808
- MC\_GearInPos, 21, 644, 671, 676, 678, 681, 808
- MC\_GearOut, 21, 643, 668, 676, 808
- MC\_GroupContinue, 714, 723, 808
- MC\_GroupDisable, 21, 685, 715, 776, 808
- MC\_GroupEnable, 21, 685, 716, 776, 779, 808
- MC\_GroupHalt, 21, 685, 717, 808
- MC\_GroupInterrupt, 714, 723, 808
- MC\_GroupReadActualAcceleration, 724, 779, 808
- MC\_GroupReadActualPosition, 21, 685, 725, 779, 808
- MC\_GroupReadActualVelocity, 22, 685, 726, 779, 808
- MC\_GroupReadError, 22, 685, 727, 808
- MC\_GroupReadStatus, 22, 685, 728, 808
- MC\_GroupReset, 730, 776, 808
- MC\_GroupSetOverride, 731, 808
- MC\_GroupSetPosition, 734, 808
- MC\_GroupStop, 718, 736, 808
- MC\_Halt, 21, 541, 573, 808
- MC\_HaltSuperimposed, 575, 809
- MC\_Home, 21, 540, 576, 809
- MC\_MoveAbsolute, 21, 540, 578, 585, 614, 641, 672, 809
- MC\_MoveAdditive, 581, 809
- MC\_MoveCircularAbsolute, 21, 685, 739, 809
- MC\_MoveCircularRelative, 21, 685, 744, 809
- MC\_MoveContinuousAbsolute, 584, 809
- MC\_MoveContinuousRelative, 588, 809
- MC\_MoveDirectAbsolute, 21, 685, 749, 786, 809
- MC\_MoveDirectRelative, 21, 685, 753, 809
- MC\_MoveLinearAbsolute, 21, 685, 718, 721, 731, 737, 757, 776, 781, 809
- MC\_MoveLinearRelative, 21, 685, 761, 809
- MC\_MovePath, 21, 685, 765, 809
- MC\_MovePath\_PH, 768, 809
- MC\_MoveRelative, 21, 540, 589, 592, 595, 809
- MC\_MoveSuperimposed, 575, 595, 678, 681, 809
- MC\_MoveVelocity, 21, 540, 598, 809
- MC\_PhasingAbsolute, 21, 644, 678, 809



MC\_PhasingRelative, 21, 644, 681, 809  
 MC\_PositionProfile, 21, 541, 601, 641, 647, 809  
 MC\_Power, 21, 540, 605, 809  
 MC\_ReadActualPosition, 21, 541, 606, 809  
 MC\_ReadAxisError, 21, 541, 607, 809  
 MC\_ReadBoolParameter, 608, 809  
 MC\_ReadCartesianTransform, 22, 685, 770, 809  
 MC\_ReadParameter, 21, 541, 609, 809  
 MC\_ReadStatus, 611, 809  
 MC\_Reset, 21, 541, 613, 730, 809  
 MC\_SetCartesianTransform, 22, 685, 770, 772, 809  
 MC\_SetCartesianTransforms, 779  
 MC\_SetKinTransform\_Lin, 739, 744, 749, 753, 757, 761, 776  
 MC\_SetOverride, 614, 809  
 MC\_SetPosition, 576  
 MC\_Stop, 21, 541, 616, 809  
 MC\_TorqueControl, 618, 629, 809  
 MC\_UngroupAllAxes, 775, 809  
 MC\_UngroupAllAxis, 776, 779  
 MC\_VelocityProfile, 21, 541, 621, 809  
 MC\_WriteBoolParameter, 625, 809  
 MC\_WriteParameter, 21, 541, 626, 809  
 MCP\_AccelerationProfile, 542, 569, 809  
 MCP\_CamIn, 645, 647, 659, 809  
 MCP\_CamTableSelect, 21, 644, 647, 659, 660, 809  
 MCP\_CombineAxes, 649, 665, 810  
 MCP\_GearIn, 651, 668, 810  
 MCP\_GearInPos, 653, 671, 810  
 MCP\_GroupHalt, 687, 717, 810  
 MCP\_GroupInterrupt, 688, 723, 810  
 MCP\_GroupSetOverride, 689, 731, 810  
 MCP\_GroupSetPosition, 690, 734, 810  
 MCP\_GroupStop, 691, 736, 810  
 MCP\_Halt, 544, 573, 810  
 MCP\_HaltSuperimposed, 545, 575, 810  
 MCP\_Home, 546, 576, 810  
 MCP\_MoveAbsolute, 548, 578, 810  
 MCP\_MoveAdditive, 550, 581, 810  
 MCP\_MoveCircularAbsolute, 692, 739, 810  
 MCP\_MoveCircularRelative, 694, 744, 810  
 MCP\_MoveContinuousAbsolute, 552, 584, 810  
 MCP\_MoveContinuousRelative, 554, 588, 810  
 MCP\_MoveDirectAbsolute, 696, 749, 810  
 MCP\_MoveDirectRelative, 698, 753, 810  
 MCP\_MoveLinearAbsolute, 700, 757, 810  
 MCP\_MoveLinearRelative, 702, 761, 810  
 MCP\_MovePath, 704, 765, 810  
 MCP\_MovePath\_PH, 706, 810  
 MCP\_MoveRelative, 556, 592, 810  
 MCP\_MoveSuperimposed, 558, 595, 810  
 MCP\_MoveVelocity, 559, 598, 810  
 MCP\_PhasingAbsolute, 655, 678, 810  
 MCP\_PhasingRelative, 657, 681, 810  
 MCP\_PositionProfile, 561, 601, 810  
 MCP\_SetCartesianTransform, 708, 772, 810  
 MCP\_SetKinTransform\_Arm, 709, 810  
 MCP\_SetKinTransform\_Schunk, 810  
 MCP\_SetKinTransform\_UR, 711, 810  
 MCP\_SetOverride, 563, 614, 810  
 MCP\_Stop, 564, 616, 810  
 MCP\_TorqueControl, 565, 618, 810  
 MCP\_VelocityProfile, 567, 621, 810  
 MCU, 208, 260, 810  
 MDL, 19, 367, 388, 389, 810  
 MDLI, 19, 367, 389, 811  
 metoda nejmenších čtverců, 144  
 MID, 19, 325, 331, 811  
 MINMAX, 25, 152, 811  
 ML\_DGEBAK, 415  
 ML\_DGEBAL, 416  
 ML\_DGEBRD, 20, 417  
 ML\_DGECON, 418  
 ML\_DGEES, 419  
 ML\_DGEES , 20  
 ML\_DGEEV, 20, 420  
 ML\_DGEHRD, 421  
 ML\_DGELQF, 422  
 ML\_DGELSD, 423  
 ML\_DGEQRF, 20, 424  
 ML\_DGESDD, 20, 425  
 ML\_DLACPY, 426  
 ML\_DLANGE, 427  
 ML\_DLASET, 428

- ML\_DTRSYL, 429
- model
  - druhého řádu s dopravním zpožděním, 394
  - FOPDT, 382
  - procesu, 388
  - procesu s proměnnými parametry, 389
  - prvního řádu s dopravním zpožděním, 382
  - SOPDT, 394
  - stavový
    - diskrétní, 377
    - diskrétní s dopravním zpožděním, 374
    - spojitý, 372
    - spojitý s dopravním zpožděním, 369
- modul, 43
  - rozšiřující, 39
  - rozšiřující systému REXYGEN, 43
- modulace
  - šířková, 235
- MODULE, 35, 39, 43, 811
- MOFN, 816
- MOSS, 20, 515, 525
- MP, 18, 183, 190, 811
- MqttPublish, 21, 533, 534, 811, 816
- MqttSubscribe, 21, 533, 536, 811, 816
- MUL, 18, 25, 86, 116, 811
- MVD, 19, 368, 390, 811
- MX\_AT, 430
- MX\_ATSET, 431
- MX\_CNADD, 432
- MX\_CNMUL, 433
- MX\_CTODPA, 434
- MX\_DIM, 435
- MX\_DIMSET, 436
- MX\_DSAGET, 437
- MX\_DSAREF, 438
- MX\_DSASET, 439
- MX\_DTRNSP, 20, 440
- MX\_DTRNSQ, 441
- MX\_FILL, 20, 442
- MX\_MAT, 20, 443
- MX\_RAND, 20, 444
- MX\_REFCOPY, 445
- MX\_SLFS, 446
- MX\_VEC, 20, 449
- MX\_WRITE, 450
- NOT, 18, 25, 263, 286
- NOT\_, 811
- NSCL, 25, 153, 811
- NSSM, 379, 391, 811, 816
- OPC server, 478
- OpcUaReadValue, 22, 795, 796, 811, 816
- OpcUaServerValue, 22, 795, 798, 811, 816
- OpcUaWriteValue, 22, 795, 800, 811, 816
- operační systém, 44
- operace
  - binární, 119
  - bitová, 272
- OR, 18, 25, 263, 287, 288
- OR\_, 811
- ORHEXD, 25, 287, 288, 811
- OROCT, 287, 288, 811
- ORQUAD, 287, 288, 811
- OSCALL, 44, 470, 811
- OSD, 25, 154, 811, 816
- OUTHEXD, 76, 78, 811
- OUTOCT, 76, 78, 811
- Outport, 70, 72, 73, 347, 357, 637, 811
- OUTQUAD, 76, 78, 811
- OUTRHEXD, 78, 811
- OUTROCT, 78, 811
- OUTRQUAD, 78, 811
- OUTRSTD, 79, 811
- OUTSTD, 67, 74, 76, 811
- ovladač
  - konfigurační data, 39
  - pořadí inicializace, 39
  - soubor s příponou .rio, 39
  - systém REXYGEN, 39
  - uživatelská dokumentace, 41
  - vstupně-výstupní, 39
  - vstupně-výstupní s úlohami, 59
- pásmo propustnosti, 138
- překlad
  - program REXYGEN Compiler, 42

- překladač REXYGEN Compiler, 34
- přepínač
  - vstupu pro vysledování, 260
- PARA, 349, 811
- PARAM, 495
- parametr
  - tick, 34
  - nastavitelný ze vstupu, 351
  - vzdáleně nastavovaný, 354, 356
  - vzdáleně získávaný, 344, 346
- PARB, 351, 811
- PARE, 350, 811, 816
- PARI, 351, 811
- PARR, 19, 170, 343, 351, 811
- PARS, 353, 811
- PGAVR, 811, 815
- PGBAT, 811, 815
- PGBUS, 811, 815
- PGCB, 812, 815
- PGENG, 812, 815
- PGGEN, 812, 815
- PGGS, 812, 815
- PGINV, 812, 815
- PGLOAD, 812, 815
- PGMAINS, 812, 815
- PGSENS, 812, 815
- PGSG, 812, 815
- PGSIM, 812, 815
- PGSOLAR, 812, 815
- PGWIND, 812, 815
- PID
  - PID regulátor, 223
  - s autotunerem, 210
  - s momentovým autotunerem, 217
  - s přepínáním parametrů, 215
  - s parametry na vstupech, 226
  - se statikou, 213
- PIDAT, 18, 196, 210, 812
- PIDE, 213, 812
- PIDGS, 18, 196, 215, 812
- PIDMA, 18, 196, 217, 247, 478, 812
- PIDU, 210, 213, 215, 223, 226, 247, 250, 260, 812
- PIDUI, 226, 812
- PJROCT, 19, 325, 332, 812
- PJSEXOCT, 336, 812, 816
- PJSOCT, 332, 334, 336, 812
- pořadí
  - inicializace úloh, 57
  - inicializace modulů, 43
  - spouštění úloh, 57
  - zavádění modulů, 43
- POL, 18, 25, 86, 117, 812
- posloupnost
  - binární pseudonáhodná, 191
- POUT, 228, 812
- PRBS, 18, 183, 191, 812
- prediktivní řízení, 231
- PRGM, 229, 812
- priorita
  - úloh, 57
  - logická, 34, 39, 46
  - závislost na operačním systému, 34
- program
  - REXYGEN Compiler, 42
  - REXYGEN Studio, 32, 39, 46, 57, 59, 70, 345, 347, 354, 357
  - REXYGEN Studio, příznak Enable, 48
  - REXYGEN Studio, tlačítko Halt/Run, 48
  - REXYGEN Studio, tlačítko RESET, 48
  - externí, 468
  - týdenní, 302
- programovatelný blok pro Python, 508
- PROJECT, 45, 812
- projekt
  - hlavní soubor, 34, 39
- protokol
  - UDP/IP, 475
- prvek
  - třístavový, 261
- PSD, 20, 515, 527
- PSMPC, 231, 812
- pulz, 228
  - ručně generovaný, 190
- pulzní výstup, 228
- PWM, 216, 222, 225, 227, 235, 255, 812
- PYTHON, 508, 518, 812, 816

- QCEDPOPT, 20, 455, 456, 812  
 QFC, 80, 81, 487, 812  
 QFD, 78–81, 487, 812  
 QP\_MPC2QP, 20, 455, 458, 812  
 QP\_OASES, 20, 455, 460, 812  
 QP\_UPDATE, 20, 455, 463, 812  
 QTASK, 17, 28, 34, 35, 41, 46, 49, 57, 369, 372, 812  
  
 Rate monotonic scheduling, 35  
 RDC, 20, 467, 475, 478, 812  
 RDFT, 25, 155  
 režie  
     jádra řídicího systému, 34  
 REC, 25, 118, 812  
 REGEXP, 19, 325, 337, 813  
 regulátor  
     fuzzy, 198  
     krokový s polohovou zpětnou vazbou, 247  
     krokový s rychlostním vstupem, 250  
     PID, 223  
     PID s autotunerem, 210  
     PID s momentovým autotunerem, 217  
     PID s přepínáním parametrů, 215  
     PID s parametry na vstupech, 226  
     PID se statikou, 213  
     prediktivní, 231  
     s klouzavým režimem, 255  
     stavový s frekvenčním autotunerem, 240  
 REL, 25, 119, 813  
 relé  
     s hysterezí, 237  
     s předstihem, 197  
 REPLACE, 19, 325, 338, 813  
 REXLANG, 20, 313, 467, 480, 518, 813  
 RLIM, 25, 157, 813  
 RLY, 237, 813  
 RM\_AxesGroup, 713, 715, 716, 727, 776, 813  
 RM\_Axis, 576, 606–609, 614, 625–627, 634, 636, 637, 713, 715, 731, 776, 779, 813  
 RM\_AxisGroup, 775  
 RM\_AxisOut, 634, 813  
 RM\_AxisSpline, 70, 135, 627, 635, 776, 813  
 RM\_DirectTorque, 813, 816  
 RM\_DirectVelocity, 813, 816  
 RM\_DriveMode, 813, 816  
 RM\_Feed, 781, 813  
 RM\_Gcode, 783, 813  
 RM\_GroupTrack, 786, 813  
 RM\_HomeOffset, 640, 813, 816  
 RM\_Track, 641, 786, 813  
 rozvrh  
     týdenní, 302  
 RS, 25, 290, 293, 813  
 RTOI, 25, 120, 813  
 RTOS, 19, 325, 339, 813  
 RTOV, 451, 469, 813  
 rychlá Fourierova transformace, 523  
 rychlá smyčka, 42  
  
 S10F2, 26, 158, 814  
 S\_AND, 813  
 S\_BC, 813  
 S\_CMP, 813  
 S\_CMPT, 816  
 S\_CTS, 813  
 S\_LB, 813  
 S\_NOT, 813  
 S\_OR, 813  
 S\_POR, 816  
 S\_PULS, 813  
 S\_PV, 813  
 S\_RCK, 816  
 S\_RS, 813  
 S\_SEL, 813  
 S\_SELVAL, 813  
 S\_SR, 813  
 S\_SUMC, 813  
 S\_TDE, 813  
 S\_TDR, 813  
 S\_TLATCH, 813  
 S\_VALB, 814  
 S\_VALC, 814  
 SAI, 26, 158, 159, 161, 814  
 SAT, 238, 814  
 saturace výstupu, 238

- SC2FA, 240, 814
- SCU, 216, 222, 225, 227, 247, 250, 814
- SCUV, 216–218, 222–225, 227, 250, 814
- SEL, 164, 814
- selektor
  - aktivního regulátoru, 254
- SELHEXD, 25, 164, 166, 814
- SELOCT, 164, 166, 814
- SELQUAD, 164, 166, 814
- SELSOCT, 340, 814
- SELU, 254, 814
- servoventil, 390
- SETPA, 19, 70, 343, 344, 354, 357, 814
- SETPB, 356, 814
- SETPI, 356, 488, 814
- SETPR, 19, 343, 356, 360, 814
- SETPS, 19, 343, 358, 814
- SG, 18, 183, 193, 814
- SGI, 193, 814
- SGSLP, 346, 357, 359, 363, 814
- SHIFTOCT, 25, 168, 814
- SHLD, 25, 170, 351, 814
- SILO, 19, 343, 361, 363, 814
- SILOS, 19, 343, 365, 814
- simulace
  - běh v reálném čase, 47
  - parametry, 47
- Simulink, 42, 47, 475
- SINT, 25, 146, 172, 814
- SLEEP, 47, 814
- SMHCC, 255, 814
- SMHCCA, 258, 814
- SMTP, 20, 467, 501, 814
- SOPDT, 19, 367, 394, 814
- součinitel relativního tlumení, 138
- SPIKE, 25, 161, 162, 173, 814
- SPLIT\_DT, 816
- SQR, 18, 25, 86, 122, 814
- SQRT, 18, 25, 86, 123
- SQRT\_, 814
- SR, 25, 291, 814
- SRTF, 48, 814
- SSW, 25, 175, 814
- STATELOAD, 50
- STATESAVE, 50, 52
- stavový model, 372, 377
  - s dopravním zpožděním, 369, 374
- STEAM, 20, 467, 503, 814, 816
- STOR, 19, 325, 341, 815
- STR2DT, 816
- SUB, 18, 25, 86, 89, 124, 815
- subsystém, 72
  - archivační, 305
- SubSystem, 66, 68, 72, 74, 76, 815
- SWR, 25, 176, 815
- SWU, 260, 815
- SWVMR, 453, 815
- SYSEVENT, 54, 817
- SYSLOG, 56, 817
- system
  - druhého řádu, 394
  - prvního řádu, 382
- T2STR, 817
- týdenní časovač, 302
- třístavový výstup, 261
- TASK, 17, 28, 34, 35, 41, 46, 49, 57, 59, 369, 372, 815
- task
  - quick, 46
- TB1, 816
- TB2, 816
- TB3, 816
- TB6, 816
- TC, 19, 295, 299
- TESTS, 816
- TIME, 19, 295, 297, 301, 815
- TIMER, 18, 25, 263, 292
- TIMER\_, 815
- TIODRV, 35, 41, 59, 815
- trajektorie
  - časově optimální, 134
- transformace
  - Fourierova, rychlá, 523
- trend
  - záznam, 317, 320
- TRIM, 342, 817
- TRND, 317, 320, 815

TRNDV, 320, 815  
TSE, 247, 250, 261, 815  
typ  
    parametr, 23  
    výstup, 23  
    vstup, 23  
typy  
    proměnných, 23  
TZ2UTC, 817  
  
UART, 20, 467, 490, 505  
UTC2TZ, 817  
UTOI, 25, 125, 815  
  
výkonová spektrální hustota, 527  
výkonové spektrum, 527  
výstup  
    pulzní, 228  
VAC, 816  
VDEL, 25, 177, 815  
ventil  
    s motorizovaným pohonem, 390  
VIN, 26, 78, 79, 81, 82, 487, 815  
VOUT, 80, 83, 487, 815  
VTOR, 155, 454, 469, 779, 815  
  
WASM, 815, 816  
WEEK, 817  
WSCH, 19, 295, 302, 815  
WWW, 61, 815  
  
zásobník  
    velikost, 39  
záznam dat, 317, 320  
zadávání  
    ruční, 208  
zpětná vazba, 42  
zpoždění  
    dopravní, 382, 394  
ZV4IS, 25, 178, 815

