Funkční bloky systému REXYGEN
Referenční příručka

REX Controls s.r.o.

Verze 3.0
15.12.2023
Plzeň

Obsah

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**
Kapitola popisuje zejména bloky sloužící pro konfiguraci struktury a časování jednotlivých objektů zařazovaných do systému reálného času řídicího systému REXYGEN (programu RexCore).

3 **INOUT – Bloky vstupů a výstupů systému REXYGEN**
Tato podknihovna vstupně-výstupních bloků opět obsahuje převážně bloky určené jen pro systém REXYGEN a zprostředkovávající hlavně vazbu mezi řídicími úlohami a vstupně-výstupními ovladači.

4 **MATH – Matematické bloky**
Podknihovna popisuje většinou jednoduché bloky pro matematické operace a základní matematické funkce.

5 **ANALOG – Zpracování analogových signálů**
Mezi bloky pro zpracování analogových signálů patří integrátor, derivátor, dopravní zpoždění, vlečný průměr, komparátory a selektory, filtry. Velmi zajímavým blokem je rozběhová jednotka `AVS`.

6 **GEN – Generátory signálů**
Kapitola popisuje bloky generující analogové i logické testovací signály.

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**
Kapitola popisuje bloky pro kombinační i sekvenční logické řízení od jednoduchých logických operací (negace, součet, součin), až po sekvenční logický automat `ATMT`, implementující standard SCF (Sequential Function Charts, dříve Grafcet).

10 **ARC – Archivace dat**
Mezi bloky pro archivaci dat v systému REXYGEN patří bloky pro generování alarmů a bloky pro záznam trendů přímo na cílovém zařízení.

12 **PARAM – Práce s parametry**
Bloky této podknihovny umožňují pracovat s parametry konfigurace systému REXYGEN zejména ukládat a nahrávat parametry nebo je vzdáleně modifikovat.

13 **MODEL – Modely dynamických systémů**
Systém REXYGEN může být využit i pro tvorbu matematických modelů dynamických systémů běžících v reálném čase. Bloky této podknihovny byly vyvinuty právě pro takové účely.

14 **MATRIX – Práce s maticovými a vektorovými daty**
Tato podknihovna obsahuje bloky pro práci s vektorovými a maticovými signály v systému REXYGEN.

20 **MC_SINGLE – Řízení pohybu v jedné ose**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu v jedné ose.

21 **MC_MULTI – Řízení pohybu ve více osách**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu ve více osách.

22 **MC_COORD – Koordinované řízení pohybu**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro koordinované řízení pohybu.

16 **SPEC – Speciální bloky**
Do skupiny speciálních bloků patří v současné době dva zajímavé bloky. Prvním je blok REXLANG, umožňující překlad a interpretaci uživatelských algoritmů vytvořených v jazyce velmi podobném jazyku C (syntaxe většiny příkazů jazyka REXLANG je totožná se syntaxí jazyka C). Druhým blokem je blok RDC, umožňující v reálném čase komunikaci mezi dvěma systémy REXYGEN.

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:

`<jmeno>` [jm] Podrobný popis vstupu (výstupu, parametru) `<jmeno>`. Matematický symbol jm 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. [⊙`<def>`] [↓`<min>`] [↑`<max>`] `<typ>`

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 přípustou 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[1].

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.

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:

[1] Přesný rozsah typu `Large` je -9223372036854775808 až 9223372036854775807.

`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, 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`, `ptype1`, `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 `'_'` (podtržítko). 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 (OLE for Process Control), viz [1] a obsahují jednobajtovou informaci, jejíž struktura je uvedena v tabulce 1.2

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].

Číslo bitu	7	6	5	4	3	2	1	0
Váha bitu	128	64	32	16	8	4	2	1
Bitová pole	**Kvalita**		**Substatus**				**Omezení**	
	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

Kapitola 2

EXEC – Konfigurace exekutivy reálného času

Obsah

ALARMS – Hromadná definice alarmů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok se umisťuje 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:

`id` ... Identifikátor alarmu, přes který se odkazuje blok ALM i záznamy v archivu.

`level` ... Položka level v archivu (pokud je ukládání do archivu povoleno).

`archives` ... 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.

`group` ... Rezervováno pro budoucí použití. Jedná se o další hodnotu (nebo bitové pole) kterou lze použít k filtraci alarmů ve vizualizaci.

`name` ... Název alarmu; může být použit jako identifikátor, proto by měl být jednoznačný.

`description` ... 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 angličnina, 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>]]`,

přičemž 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 exponencialním 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 uspořádany 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

`afile`	Název `.csv` souboru s daty	`String`

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

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

Výstup

next	Výstup sloužící pro zřetězování archivů připojením na vstup prev následujícího archivu	Long (I32)

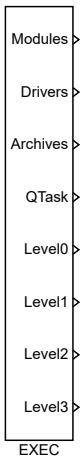
Parametry

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

EXEC – Exekutiva reálného času

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `EXEC` tvoří základ tzv. *hlavního souboru projektu* ve formátu `.mdl`, kterým se konfigurují 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`, nejnižšší 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

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

Parametry

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

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

HMI – * Konfigurace visualizace

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

IncludeHMI	Zahrnout soubory HMI do projektu	⊙on	Bool
HmiDir	Výstupní adresář pro soubory vizualizace (HMI)	⊙hmi	String
SourceDir	Zdrojový adresář	⊙hmisrc	String
GenerateWebWatch	Vygenerovat WebWatch vizualizaci z MDL souborů	⊙on	Bool
GenerateRexHMI	Při překladu projektu vygenerovat HMI ze SVG a JS souborů	⊙on	Bool
RedirectToHMI	Webserver bude automaticky přesměrovávat na stránku s HMI	⊙on	Bool
Compression	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

`Title`	Název projektu	`String`
`Author`	Autor projektu	`String`
`Description`	Stručný popis projektu	`String`
`Customer`	Informace o zákazníkovi	`String`

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 Studiosystému 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

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

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 nevyhovyje, 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	⊙1	`Long (I32)`
`stack`	Velikost zásobníku (v bytech)	⊙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 umisťuje 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 překladu 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 vypouštěn, 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

`prev`	Vstup sloužící pro připojení prvního modulu na výstup `Modules` bloku EXEC nebo k připojení na výstup `next` předchozího modulu	`Long (I32)`

Výstup

`next`	Výstup sloužící pro zřetězování modulů připojením na vstup `prev` 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

TRG	Spuštění zvolené akce		Bool

Výstupy

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

Parametr

action	Systémová funkce	⊙1	Long (I32)
	1 ..... restartovat systém		
	2 ..... vypnout systém		
	3 ..... zastavit systém (HALT)		
	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

`CompileParams`	Parametry příkazového řádku programu REXYGEN Compiler		`String`
`SourcesOnTarget`	Uložit zdrojové soubory na cílové zařízení	⊙`on`	`Bool`
`TargetURL`	Výchozí adresa cílového zařízení (typicky IP adresa)		`String`
`LibraryPath`	Cesta ke knihovnám bloků. Může být absolutní nebo relativní vůči adresáři s projektem.		`String`

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

prev	Vstup, sloužící pro k připojení k výstupu QTask bloku EXEC	Long (I32)

Parametry

factor	Násobek času tick bloku EXEC určující periodu úlohy (factor * tick)	⊙1	Long (I32)
stack	Velikost zásobníku (v bytech)	⊙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

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).

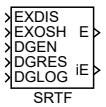
Parametr

ts	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 spouštění 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 `bname` `2` ..... interní chyba systému REXYGEN (nesprávné ukazatele) `3` ..... příznak se nepodařilo nastavit (timeout)	`Long (I32)`

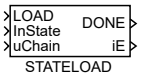
Parametr

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

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

Parametry

`filename`	Název souboru pro načtení stavu	`String`
`blocks`	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	`String`
`depth`	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. ↓0 ↑65535	`Long (I32)`

`mask`	Vyberte, které objekty jsou načteny. Každý bit čísla znamená: • 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	`Long (I32)`
`LoadOnInit`	Načtení během inicializace konfigurace ⊙on	`Bool`
`STRICT`	Pokud je nastaveno, soubor je zkontrolován vůči aktuální konfiguraci a data jsou odmítnuta, pokud nedochází ke shodě ⊙on	`Bool`

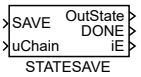
Výstupy

`DONE`	Stav načten	`Bool`
`iE`	Kód chyby při selhání	`Error`

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

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

Parametry

`filename`	Název souboru, kam se ukládá	`String`
`blocks`	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.	`String`
`depth`	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	`Long (I32)`

`mask`	Vyberte, které objekty se mají ukládat. Každý bit čísla představuje: • 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	`Long (I32)`
`SaveOnExit`	Pokud je nastaveno, soubor se uloží při ukončení konfigurace. ⊙`on`	`Bool`

Výstupy

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

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

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

Výstupy

`VALID`	Platná (aktuální) výstupní data	`Bool`
`sEvent`	Archivní položka (JSON formát)	`String`
`sVal`	Hodnota archivní položky (pro text)	`String`
`iVal`	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

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

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 `Level3` 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 `Level3` (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

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

Výstup

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

Parametry

`factor`	Faktor spouštění, násobek periody `tick*ntick<i>` bloku `i`-té výpočetní úrovně bloku EXEC určující periodu úlohy (`factor * tick* ntick<i>`) ⊙1	`Long (I32)`
`start`	Číslo tiku periody dané výpočetní úrovně, na kterém má být úloha spuštěna ↓0 ↑`ntick<i>`	`Long (I32)`
`stop`	Číslo tiku periody dané výpočetní úrovně, do kterého má být úloha dokončena ↓`start+1` ↑`ntick<i>`	`Long (I32)`
`stack`	Velikost zásobníku (v bytech) ⊙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) doplněným příponou `.mdl`.	`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

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

Výstupy

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

Parametry

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

`factor`	Násobek parametru `tick` bloku `EXEC` určující periodu spouštění úlohy ovladače	↓1 ⊙10	`Long (I32)`
`stack`	Velikost zásobníku úlohy ovladače v bytech	↓1024 ⊙10240	`Long (I32)`
`pri`	Priorita úlohy ovladače	↓1 ↑31 ⊙3	`Long (I32)`
`cpu`	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

`Source`	Zdrojový adresář	`String`
`Target`	Cílový adresář	`String`
`Compression`	Aktivovat kompresi dat	`Bool`

Kapitola 3

INOUT – Bloky vstupů a výstupů systému REXYGEN

Obsah

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.

Vstup

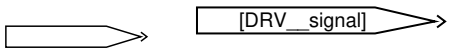
`u`	Vstupní signál	`Any`

Parametry

`Format`	Formát zobrazované hodnoty ⊙1 `Best fit` stejné jako `long`, ale pro extrémě malá nebo velká čísla jako `long_e`; základní formát pro desetinná čísla `short` běžné zobrazení desetiného čísla s nejvýše 3 ciframi za desetinnou čárkou (tečkou) `long ..` běžné zobrazení desetiného čísla s nejvýše s plným počtem desetinných míst (tj. až 15) `short_e` exponenciální (vědecké) zobrazení desetiného čísla s nejvýše 3 ciframi za desetinnou čárkou (tečkou) `long_e` exponenciální (vědecké) zobrazení desetiného čísla s nejvýše s plným počtem desetinných míst (tj. až 15) `bank ..` běžné zobrazení desetiného čísla s 2 ciframi za desetinnou čárkou (tečkou) `dec ...` běžné zobrazení celého čísla (v desítkové soustavě); základní formát pro celá čísla `hex ...` celé číslo v hexadecimálním formátu (šestnáctková soustava) `bin ...` celé číslo v binárním formátu (dvojková soustava) `oct ...` celé číslo v oktalovém formátu (osmičková soustava)	`Long (I32)`
`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 překladu 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 překladu 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

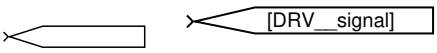
`value`	Signál z I/O ovladače nebo bloku `Goto`. Typ výstupu je určen typem signálu, který je na vlajku přiveden.	`Any`

Parametr

`GotoTag`	Odkaz na parametr `GotoTag` bloku `Goto`, se kterým má být blok `From` propojen nebo odkaz na vstupní signál ovladače systému REXYGEN, který má být přiveden na výstup bloku.	`String`

`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__DO1`. 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

`value`	Signál odesílaný do I/O ovladače nebo bloku `From`. V případě napojení na I/O ovladač systému REXYGEN, je typ vstupu určen ovladačem z parametru `GotoTag`.	`Any`

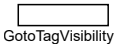
Parametry

`GotoTag`	Odkaz na parametr `GotoTag` bloku `From`, se kterým má být blok `Goto` propojen, nebo odkaz na výstupní signál ovladače systému REXYGEN, jehož hodnota je pak určena vstupem bloku.	`String`
`TagVisibility`	Viditelnost (dostupnost) daného bloku uvnitř `.mdl` souboru. Určuje podmínky pro umístění bloku `Goto` a k němu odpovídajícímu bloku `From` tak, aby byly vzájemně dostupné: ⊙`local` `local` oba bloky se musí nacházet ve stejném subsystému `global` bloky mohou být umístěny kdekoliv v daném `.mdl` souboru `scoped` bloky musí být umístěny ve stejném subsystému nebo v jakékoliv hierarchické úrovni pod umístěním bloku `GotoTagVisibility` se stejným parametrem `GotoTag`	`String`

`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 překladu projektu překladačem REXYGEN Compiler a ve výsledné binární konfiguraci není obsažen, protože v reálném čase nevykonává žádnou činnost.

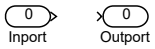
Parametr

`GotoTag`	Odkaz na parametr `GotoTag` bloku `Goto`, jehož viditelnost je dána umístěním tohoto bloku `GotoTagVisibility`	`String`

`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 přečíslovány. Pozor, pokud jsou přečíslovány 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`

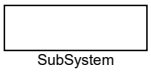
Parametry

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

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.

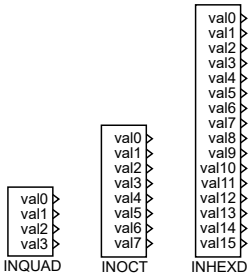
Parametry

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 popsaný 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`.

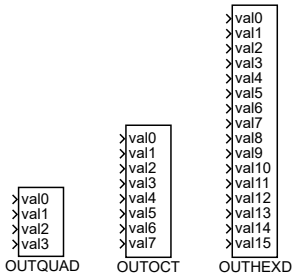
Výstupy

`val`*i*	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 popsaný způsob 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__DO. 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.

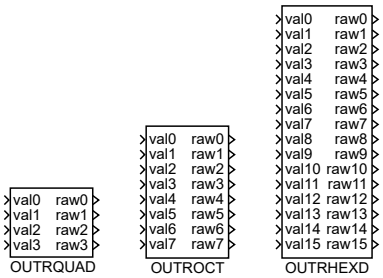
Vstupy

`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`

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 řidicího algoritmu o výsledku zápisu dvojím 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`

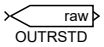
Výstupy

`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 řidicího algoritmu o výsledku zápisu dvojím 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

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

Výstup

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

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

iq	Základní příznaky kvality, viz tab. 1.2, str. 20	Long (I32)
is	Doplňující příznaky kvality, viz [1]	Long (I32)
il	Příznaky dosažení mezních úrovní, viz [1]	Long (I32)

Výstup

iqf	Bitová kombinace vstupních signálů iq, is a il	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 vstupného 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

iqf	Příznaky kvality, které mají být dekomponovány na složky iq, is a il	Long (I32)

Výstupy

iq	Příznaky základního typu kvality, viz tabulku 1.2, str. 20	Long (I32)
is	Doplňující příznaky kvality, viz [1]	Long (I32)
il	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 GU(Good if Uncertain) jsou vstupní signály v bloku VIN dále zpracovány následujícím způsobem:

- Pro GU = 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 GU = 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

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

Výstupy

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

Parametr

GU	Přípustnost kvality UNCERTAIN off ... kvalita UNCERTAIN je nepřípustná on .... kvalita UNCERTAIN je přípustná	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

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

Výstup

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

Kapitola 4

MATH – Matematické bloky

Obsah

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,

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

Vstup

u	Analogový vstupní signál	Double (F64)

Výstupy

y	Absolutní hodnota vstupního signálu	Double (F64)
sgn	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.

Vstupy

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

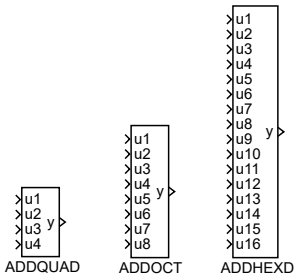
Výstup

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 nl 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+...+u16. Pokud bude například nl=2,5,7, tak bude realizována funkce y = u1 − u2 + u3 + u4 − u5 + u6 − u7 + ... + u16.

Pro jednoduché operace sčítání a odečítání můžete použít bloky ADD a SUB.

Vstupy

u1..u16	Analogové vstupní signály	Double (F64)

Výstup

y	Výsledná hodnota	Double (F64)

Parametr

nl	Seznam signálů, které se místo přičítání odečítají. 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 decimálně 157, binárně 10011101.	Long (I32)

CNB – Booleovská (logická) konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNB slouží pro zadání Booleovské (logické) konstanty.

Výstup

Y	Logický výstupní signál	Bool

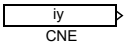
Parametr

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

Parametry

yenum	Konstanta ze seznamu	⊙1: option A	String
pupstr	Definice seznamu konstant ⊙1: option A|2: option B|3: option C		String

Výstup

iy	Celočíselný výstupní signál	Long (I32)

CNI – Celočíselná konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNI slouží pro zadání celočíselné konstanty.

Výstup

iy	Celočíselný výstupní signál		Long (I32)

Parametr

icn	Celočíselná konstanta	⊙1	Long (I32)
vtype	Typ hodnoty výstupu, může nabývat hodnot:	⊙4	Long (I32)
	2 ..... Byte (rozsah 0 ... 255)		
	3 ..... Short (rozsah -32768 ... 32767)		
	4 ..... Long (rozsah -2147483648 ... 2147483647)		
	5 ..... Word (rozsah 0 ... 65536)		
	6 ..... DWord (rozsah 0 ... 4294967295)		
	10 .... Large (rozsah -9223372036854775808...9223372036854775807)		

CNR – Reálná konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNR slouží pro zadání reálné konstanty.

Výstup

y	Analogový výstupní signál		Double (F64)

Parametr

ycn	Reálná konstanta	⊙1.0	Double (F64)

DIF – Blok diference

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).

Vstup

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

Výstup

y	Diference vstupního signálu	Double (F64)

Parametr

ISSF	Nulový výstup při spuštění off ... v prvním cyklu bude na výstupu y = u on .... v prvním cyklu bude výstup y = 0	Bool

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.

Vstupy

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

Výstupy

y	Podíl vstupních signálů	Double (F64)
E	Indikátor chyby – dělení nulou	Bool

Parametr

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

EAS – Rozšířené sčítání a odečítání

Symbol bloku

Licence: STANDARD

Popis funkce

Blok EAS sčítá vstupní analogové signály u1, u2, u3 a u4 s příslušnými vahami a, b, c a d. Výstup y je pak dán vztahem

$$y = a * u1 + b * u2 + c * u3 + d * u4 + y0.$$

Vstupy

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)
u3	Třetí analogový vstup bloku	Double (F64)
u4	Čtvrtý analogový vstup bloku	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

a	Váhový koeficient pro vstup u1	⊙1.0	Double (F64)
b	Váhový koeficient pro vstup u2	⊙1.0	Double (F64)
c	Váhový koeficient pro vstup u3	⊙1.0	Double (F64)
d	Váhový koeficient pro vstup u4	⊙1.0	Double (F64)
y0	Aditivní konstanta (bias)		Double (F64)

EMD – Rozšířené násobení a dělení

Symbol bloku

Licence: STANDARD

Popis funkce

Blok EMD slouží k násobení a dělení vstupních analogových signálů u1, u2, u3 a u4 s příslušnými vahami a, b, c a d. Výstup y je pak dán vztahem

$$y = \frac{(a * u1 + a0)(b * u2 + b0)}{(c * u3 + c0)(d * u4 + d0)}. \tag{4.1}$$

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

Vstupy

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)
u3	Třetí analogový vstup bloku	Double (F64)
u4	Čtvrtý analogový vstup bloku	Double (F64)

Výstupy

y	Analogový výstupní signál	Double (F64)
E	Indikátor chyby - dělení nulou	Bool

Parametry

a	Váhový koeficient pro vstup u1	⊙1.0	Double (F64)
a0	Aditivní konstanta pro vstup u1		Double (F64)
b	Váhový koeficient pro vstup u2	⊙1.0	Double (F64)
b0	Aditivní konstanta pro vstup u2		Double (F64)
c	Váhový koeficient pro vstup u3	⊙1.0	Double (F64)
c0	Aditivní konstanta pro vstup u3		Double (F64)
d	Váhový koeficient pro vstup u4	⊙1.0	Double (F64)
d0	Aditivní konstanta pro vstup u4		Double (F64)
yerr	Náhradní hodnota pro případ chyby	⊙1.0	Double (F64)

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 určená parametrem ifn.
Tabulka funkcí bloku FNX:

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

Vstup

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

Výstupy

`y`	Výsledek vybrané funkce		`Double (F64)`
`E`	Příznak chyby		`Bool`

Parametry

`ifn`	Typ funkce (viz tabulka výše)	⊙1	`Long (I32)`
`yerr`	Náhradní hodnota pro případ chyby		`Double (F64)`

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 určená parametrem ifn.
Tabulka funkcí bloku FNXY:

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

Funkce atan2 vrací funkční hodnotu v intervalu $\langle -\pi, \pi \rangle$. 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 nesplňuje 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 $y = yerr$.

Vstupy

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

Výstupy

y	Výsledek vybrané funkce	Double (F64)

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

Parametry

`ifn`	Typ funkce (viz tabulka výše) `1` ..... atan2 `2` ..... fmod `3` ..... pow	⊙`1`	`Long (I32)`
`yerr`	Náhradní hodnota pro případ chyby		`Double (F64)`

GAIN – Násobení konstantou

Symbol bloku

Licence: STANDARD

Popis funkce

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

$$y = ku.$$

Vstup

u	Analogový vstupní signál		Double (F64)

Výstup

y	Analogový výstupní signál		Double (F64)

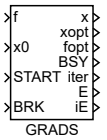
Parametr

k	Zesílení	⊙1.0	Double (F64)

GRADS – Gradientní optimalizace

Symbol bloku

Licence: ADVANCED



Popis funkce

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

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

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

Vstupy

`f`	Hodnota minimalizované funkce $f(.)$ v bodě `x`	`Double (F64)`
`x0`	Startovní bod optimalizace	`Double (F64)`
`START`	Spouštěcí signál (reaguje na náběžnou hranu)	`Bool`
`BRK`	Signál pro předčasné přerušení	`Bool`

Výstupy

`x`	Aktuální hodnota optimalizované proměnné `x`	`Double (F64)`
`xopt`	Výsledná optimální hodnota proměnné `x`	`Double (F64)`
`fopt`	Výsledná optimální hodnota funkce $f(\mathtt{x}, v)$	`Double (F64)`
`BSY`	Indikátor probíhající optimalizace	`Bool`
`iter`	Číslo aktuální iterace	`Long (I32)`
`E`	Příznak chyby	`Bool`
`iE`	Kód chyby 1 ..... $\mathtt{x} \notin < \mathtt{xmin}, \mathtt{xmax} >$ 2 ..... $\mathtt{x} = \mathtt{xmin}$ nebo $\mathtt{x} = \mathtt{xmax}$	`Long (I32)`

Parametry

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

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).

Vstupy

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

Výstupy

n	Celočíselný součet vstupních signálů	Long (I32)
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	Bool

Parametry

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	Long (I32)
SAT	Kontrola přetečení off ... přetečení se nekontroluje on .... přetečení se kontroluje		Bool

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).

Vstupy

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

Výstupy

n	Celočíselný rozdíl vstupních signálů	Long (I32)
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	Bool

Parametry

vtype	Číselný typ 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	Long (I32)
SAT	Kontrola přetečení off ... přetečení se nekontroluje on .... přetečení se kontroluje		Bool

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).

Vstupy

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

Výstupy

n	Celočíselný součin vstupních signálů	Long (I32)
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	Bool

Parametry

vtype	Číselný typ 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	Long (I32)
SAT	Kontrola přetečení off ... přetečení se nekontroluje on .... přetečení se kontroluje		Bool

IDIV – Celočíselné dělení

Symbol bloku

Licence: STANDARD

Popis funkce

Blok IDIV dělí dva vstupní celočíselné signály n = i1 ÷ i2, kde ÷ 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.

Vstupy

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

Výstupy

n	Celočíselný podíl vstupních signálů	Long (I32)
E	Příznak chyby – dělení nulou	Bool

Parametr

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙1	Long (I32)
nerr	Náhradní hodnota pro případ chyby	⊙1	Long (I32)

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.

Vstupy

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

Výstupy

n	Zbytek po celočíselném dělení	Long (I32)
E	Příznak chyby – dělení nulou	Bool

Parametr

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙1	Long (I32)
nerr	Náhradní hodnota pro případ chyby	⊙1	Long (I32)

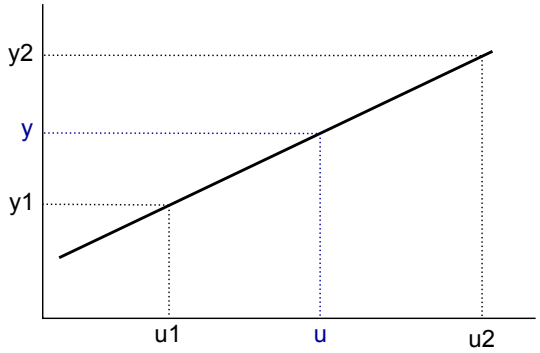
LIN – Lineární interpolace

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].



Vstup

u	Analogový vstupní signál		Double (F64)

Výstup

y	Analogový výstupní signál		Double (F64)

Parametry

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)

MUL – Násobení dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

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

Vstupy

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:

$$\mathtt{y} = \mathtt{a}_0 + \mathtt{a}_1\mathtt{u} + \mathtt{a}_2\mathtt{u}^2 + \mathtt{a}_3\mathtt{u}^3 + \mathtt{a}_4\mathtt{u}^4 + \mathtt{a}_5\mathtt{u}^5 + \mathtt{a}_6\mathtt{u}^6 + \mathtt{a}_7\mathtt{u}^7 + \mathtt{a}_8\mathtt{u}^8.$$

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

Vstup

u	Analogový vstupní signál	Double (F64)

Parametry

ai	Koeficient i-té mocniny vstupu, $i = 0, 1, \ldots, 8$	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 = on a na výstupu y je náhradní hodnota yerr.

Vstup

u	Analogový vstupní signál		Double (F64)

Výstupy

y	Analogový výstupní signál		Double (F64)
E	Indikátor chyby – dělení nulou		Bool

Parametr

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

REL – Relační operace dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

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

Vstupy

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

Výstup

`Y`	Logický výstupní signál indikující platnost relace	`Bool`

Parametr

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

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:

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

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

Vstup

r	Analogový vstupní signál	Double (F64)

Parametry

vtype	Typ hodnoty výstupu, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	Long (I32)
SAT	Detekce přetečení	⊙on	Bool

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

$$y = u^2.$$

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 = on a výstup y je nastaven na hodnotu parametru yerr.

Vstup

u	Analogový vstupní signál	Double (F64)

Výstupy

y	Odmocnina ze vstupní hodnoty	Double (F64)
E	Příznak chyby off ... bez chyby on .... odmocňování záporného čísla	Bool

Parametr

yerr	Náhradní hodnota při odmocňování záporného čísla	⊙1.0	Double (F64)

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.

Vstupy

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 chápe vstupní (kladné) číslo jako celé číslo se znaménkem v reprezentaci dvojkový doplněk, tj. běžná reprezentace v procesorech. Například číslo -1 je v 8-bitové reprezentaci je stejné s číslem 255 a v 16-bitové reprezentaci s číslem 65535. Kolika bitová reprezentace se předpokládá určuje parametr `bits`.

Hlavní použití tohoto bloku je situace, kdy hodnota z ovladače obsahuje více signálů, které se získávají vymaskováním (typicky blokem INTSM nebo BITOP). Výstupem tohoto vymaskování je vždy kladné číslo, ale pokud je význam signálu z ovladače číslo se znaménkem, použije s k získání správné hodnoty tento blok.

Protože procesory vyskytující se na trhu ukládají vícebajtová čísla různě (obvyklejší je tzv. little-endian, kde na nižší adrese je výnamnější bajt, ale vyskytují se i procesory s tzv. big-endian formátem, kde je to maopak), blok umožňuje i prohodit pořadí bajtů, pokud to není ošetřeno v ovladači. K tomu slouží parametr `SWAP`.

POZOR!!! Prohození pořadí bajtů (tj. nastavit `SWAP=on`) řeší problém s rozdílným pořadí bajtů v procesoru obvykle jen pro případ `bits=16` nebo `bits=32`.

Vstup

`u`	Vstupní signál bez znaménka	↓-9.22337E+18 ↑9.22337E+18	`Large (I64)`

Parametry

`bits`	Počet platných bitů ve vstupním signálu	↓2 ↑64 ⊙16	`Long (I32)`
`SWAP`	Otočení pořadí bajtů vstupu		`Bool`

Výstup

`i`	Zkonvertovaný (se znaménkem) vstupní signál	`Large (I64)`

Kapitola 5

ANALOG – Zpracování analogových signálů

Obsah

ABSROT – Zpracování dat z absolutního snímače 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° až 355° (popř. -175° až +175°), ale potřebujeme řídit pohyb o několik otáček. Blok předpokládá spojitý signál, takže při přechodu z například 355° na 5° předpokládá, že nastala další otáčka a úhel je ve skutečnosti 365°.

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.

Vstupy

`u`	Signál z absolutního snímače polohy	`Double (F64)`
`R1`	Reset bloku (nastavení výstupu do základního intervalu)	`Bool`

Výstupy

`y`	Vypočtená poloha	`Double (F64)`
`irev`	Počet dokončených otáček (překročení hranic intervalu)	`Long (I32)`
`MPI`	Indikátor středové polohy	`Bool`
`OLI`	Indikátor polohy mimo rozsah senzoru	`Bool`

Parametry

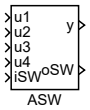
`lolim`	Dolní mez pro údaj z čidla	⊙`-3.14159265`	`Double (F64)`
`hilim`	Horní mez pro údaj z čidla	⊙`3.14159265`	`Double (F64)`
`tol`	Rozsah pro indikaci středové pozice	⊙`0.5`	`Double (F64)`
`hys`	Hystereze pro indikaci středové pozice		`Double (F64)`

`RESR`	Příznak pro resetování počítadla otáček
	`off` ... resetovat pouze vypočtenou polohu `y`
	`on` .... vynulovat i počítadlo otáček `irev`
			`Bool`

ASW – Přepínač s automatickou volbou vstupu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok ASW ukládá na výstup y hodnotu jednoho ze vstupů vstup u1, ..., u4 nebo jeden z parametrů p1, ..., p4. 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 p1, pro iSW $= 3$ je na výstupu y hodnota u3 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 u1, ..., u4 nebo parametrů p1, ..., p4, který se naposledy změnil. Pokud se změní více hodnot najednou, pak se použije hodnota podle následujícího pořadí p4, p3, p2, p1, u4, u3, u2, u1. 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 p1, ..., p4 (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 nadřízených 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 u1, ..., u4 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).

Vstupy

u1..u4	Analogové vstupní signály, ze kterých se vybírá ten aktivní	Double (F64)
iSW	Volba aktivního signálu nebo parametru	Long (I32)

Výstupy

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

Parametry

`delta`	Práh pro detekci změny	⊙`1e-06`	`Double (F64)`
`p1..p4`	Parametry, ze kterých se vybírá ten aktivní		`Double (F64)`

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

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

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

Vstup

u	Vstupní signál filtru	Double (F64)

Výstup

y	Filtrovaný výstupní signál	Double (F64)

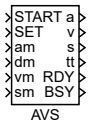
Parametr

n	Počet vzorků, ze kterých se provádí výpočet vlečného průměru ↓1 ↑10000000 ⊙10	Long (I32)
nmax	Maximální velikost parametru n (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙100	Long (I32)

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.

Vstupy

START	Spouštěcí signál (náběžná hrana off→on), start generování trajektorie	Bool
SET	Inicializace/výpočet trajektorie podle aktuálních vstupů	Bool
am	Maximální povolené zrychlení [m/s^2]	Double (F64)
dm	Maximální povolené zpomalení [m/s^2]	Double (F64)
vm	Maximální povolená rychlost [m/s]	Double (F64)
sm	Žádaná konečná poloha [m] (počáteční poloha je 0)	Double (F64)

Výstupy

a	Zrychlení [m/s^2]	Double (F64)
v	Rychlost [m/s]	Double (F64)
s	Poloha [m]	Double (F64)
tt	Čas [s]	Double (F64)
RDY	Příznak připravenosti (určuje, zda může být spuštěno generování trajektorie)	Bool
BSY	Příznak probíhající operace (určuje, zda v daném okamžiku je či není generována trajektorie)	Bool

BPF – Filtr: pásmová propusť

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 ξ 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.

Vstup

u	Vstupní signál filtru	Double (F64)
R1	Reset (stav jako po initu)	Bool
HLD	Pozastavení vykonávání bloku	Bool

Výstup

y	Filtrovaný výstupní signál	Double (F64)

Parametry

fm	Střed pásma propustnosti [Hz]	⊙1.0	Double (F64)
xi	Součinitel relativního tlumení (doporučená hodnota 0.5 až 1)	⊙0.707	Double (F64)
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav		Bool

CMP – Komparátor s hysterezí

Symbol bloku

Licence: STANDARD

Popis funkce

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

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

kde

$$e_k = u1_k - u2_k$$

a

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

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.

Vstupy

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

Výstup

Y	Logický výstupní signál	Bool

Parametr

hys	Hystereze	↓0.0 ⊙0.5	Double (F64)

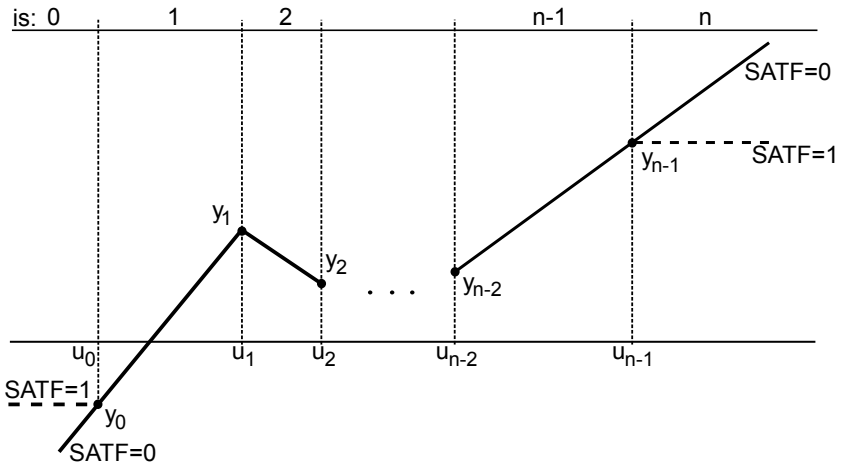
CNDR – Kompenzátor složité nelinearity

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 $\mathtt{u} < u_0$ a $\mathtt{u} > u_{n-1}$ je výstup definován v závislosti na parametru SATF.

Vstup

u	Analogový vstupní signál	Double (F64)

Výstupy

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

Parametry

nmax	Alokovaná velikost polí up, yp	↓4 ⊙10	Long (I32)
SATF	Saturace v koncových uzlech off ... signál není omezen aktivní on .... saturační meze jsou	⊙on	Bool

`up`	Vektor rostoucích u souřadnic uzlů ⊙`[0.0 3.9 3.9 9.0 14.5 20.0]`	`Double (F64)`
`yp`	Vektor y souřadnic uzlů ⊙`[0.0 0.0 15.8 38.4 72.0 115.0]`	`Double (F64)`

DEL – Dopravní zpoždění s inicializací

Symbol bloku

Licence: STANDARD

Popis funkce

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

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

Jestliže po spuštění nebo restartu (R1: off→on→off) dosud není zapamatovaných n minulých vzorků (RDY = off), potom

$$y_k = y_0,$$

kde y_0 je inicializační vstup bloku.

Vstupy

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool
y0	Počáteční hodnota výstupu	Double (F64)

Výstupy

y	Zpožděný vstupní signál	Double (F64)
RDY	Příznak připravenosti signalizující, že paměťový buffer je již naplněn vstupními vzorky	Bool

Parametr

n	Zpoždění (počet vzorků) – příslušné časové zpoždění je $n \cdot T_S$, kde T_S je perioda spouštění bloku ↓0 ↑10000000 ⊙10	Long (I32)
nmax	Maximální velikost parametru n (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙100	Long (I32)

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.

Vstup

u	Analogový vstupní signál	Double (F64)

Výstup

y	Zpožděný vstupní signál	Double (F64)

Parametr

del	Časové zpoždění [s]	⊙1.0	Double (F64)
nmax	Délka vyrovnávací paměti pro dopravní zpoždění del. Používá se pro interní alokaci paměti.	↓10 ↑10000000 ⊙100	Long (I32)

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

Symbol bloku

Licence: STANDARD

Popis funkce

Blok DER prokládá posledních n + 1 vzorků (n ≤ $N-1$, 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 = on výstupy y a z podle vztahů:

Derivace:	y	=	a
Filtrace:	z	=	b, pro $t_p = 0$
Predikce:	z	=	$at_p + b$, pro $t_p > 0$
Postdikce:	z	=	$at_p + b$, pro $t_p < 0$

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

Vstupy

u	Analogový výstupní signál	Double (F64)
RUN	Povolení běhu algoritmu off ... sledování (z=u) on .... filtrace (y – odhad derivace, z – odhad u v čase t_p)	Bool
tp	Časový okamžik pro predikci/filtraci (tp = 0 je v aktuálním okamžiku vzorkování)	Double (F64)

Výstupy

y	Odhad derivace vstupního signálu u	Double (F64)
z	Predikovaný/filtrovaný výstupní signál	Double (F64)
RDY	Příznak připravenosti (blok má k dispozici n + 1 vzorků)	Bool

Parametr

n	Počet vzorků pro lineární interpolaci (je použito n + 1 vzorků); 1 ≤ n ≤ $nmax$ ↓1 ↑10000000 ⊙10	Long (I32)
nmax	Maximální velikost parametru n (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙100	Long (I32)

EVAR – Vlečná střední hodnota a směrodatná odchylka

Symbol bloku

Licence: STANDARD

Popis funkce

Blok **EVAR** počítá střední hodnotu `mu` (μ) a směrodatnou odchylku `si` (σ) z posledních `n` vzorků vstupního signálu `u` podle vztahů

$$\begin{aligned} \mu_k &= \frac{1}{\mathtt{n}} \sum_{i=0}^{n-1} \mathtt{u}_{k-i} \\ \sigma_k &= \sqrt{\frac{1}{\mathtt{n}} \sum_{i=0}^{n-1} \mathtt{u}_{k-i}^2 - \mu_k^2} \end{aligned}$$

kde k značí aktuální okamžik vzorkování.

Vstup

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

Výstupy

`mu`	Střední hodnota vstupního signálu	`Double (F64)`
`si`	Směrodatná odchylka vstupního signálu	`Double (F64)`

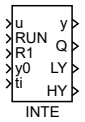
Parametr

`n`	Počet vzorků pro výpočet statistických ukazatelů ↓2 ↑10000000 ⊙100	`Long (I32)`
`nmax`	Maximální velikost parametru `n` (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙200	`Long (I32)`

INTE – Řízený integrátor

Symbol bloku

Licence: STANDARD



Popis funkce

Blok INTE realizuje řízený integrátor s proměnnou integrační časovou konstantou ti a indikací dvou úrovní výstupu ymin a ymax. Je-li RUN = on a R1 = off, potom

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

kde hodnota C = y0. Je-li RUN = off a R1 = off, je výstup y zmrazen na jeho poslední hodnotu před sestupnou hranou vstupu RUN. Je-li R1 = on, potom je výstup y resetován na počáteční hodnotu y0. 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.

Vstupy

u	Analogový vstupní signál	Double (F64)
RUN	Povolení běhu algoritmu off ... integrace je pozastavena, on ... integrace probíhá	Bool
R1	Reset bloku, inicializace výstupu integrátoru na hodnotu y0	Bool
y0	Počáteční hodnota výstupu	Double (F64)
ti	Integrační časová konstanta	Double (F64)

Výstupy

y	Výstup integrátoru	Double (F64)
Q	Příznak probíhající integrace	Bool

`LY`	Příznak dosažení spodní úrovně (`y` < `ymin`)		`Bool`
`HY`	Příznak dosažení horní úrovně (`y` > `ymax`)		`Bool`

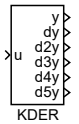
Parametry

`ymin`	Nastavení dolní úrovně	⊙`-1.0`	`Double (F64)`
`ymax`	Nastavení horní úrovně	⊙`1.0`	`Double (F64)`
`SAT`	Omezení výstupu, když je dosažena krajní úroveň		`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 `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 ω_0. Naopak frekvenční spektrum šumů by mělo být co možná nejdále od frekvence ω_0. Pro vyšší potlačení šumů je nutné volit nižší zlomovou frekvenci ω_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.

Vstup

`u`	Vstupní signál filtru	`Double (F64)`

Výstupy

`y`	Filtrovaný vstupní signál	`Double (F64)`
`dy`	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)`

Parametry

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

LPF – Filtr: dolní propusť

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 fb a ξ = xi jsou parametry bloku. Frekvence fb [Hz] určuje šířku pásma filtru a parametr xi součinitel relativního tlumení filtru. 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.

Vstup

u	Vstupní signál filtru	Double (F64)
R1	Reset (stav jako po initu)	Bool
HLD	Pozastavení vykonávání bloku	Bool

Výstup

y	Filtrovaný výstupní signál	Double (F64)

Parametry

fb	Šířka pásma [Hz] (filtr propouští frekvence v intervalu $\langle 0, \texttt{fb}\rangle$ – útlum na frekvenci fb je 3 dB, na $10 \cdot$ fb přibližně 40 dB); pro správnou funkci filtru musí platit $f_b < \frac{1}{10T_S}$, kde T_S je perioda spouštění bloku ⊙1.0	Double (F64)
xi	Součinitel relativního tlumení (doporučená hodnota 0,5 až 1) ⊙0.707	Double (F64)
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav	Bool

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 = off a minimum a maximum se hledá mezi dostupnými vzorky.

Vstupy

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

Výstupy

ymin	Nalezená minimální hodnota vstupního signálu	Double (F64)
ymax	Nalezená maximální hodnota vstupního signálu	Double (F64)
RDY	Příznak připravenosti (buffer je naplněn)	Bool

Parametry

n	Počet prvků pro výpočet minima a maxima (délka bufferu) ↓1 ↑10000000 ⊙100	Long (I32)
nmax	Maximální velikost parametru n (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙200	Long (I32)

NSCL – Kompenzátor jednoduché nelinearity

Symbol bloku

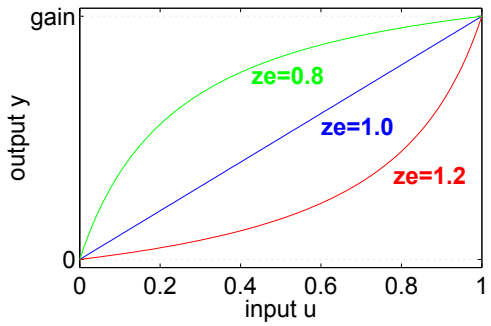
Licence: STANDARD

Popis funkce

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

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

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



Vstup

u	Analogový vstupní signál	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

gain	Zesílení	⊙1.0	Double (F64)
ze	Tvarovací parametr	⊙1.0	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).

Vstupy

u	Vstupní signál	Any

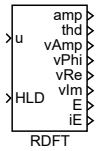
Výstupy

y	Zpožděný vstupní signál	Any

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.

Vstupy

u	Analogový vstupní signál	Double (F64)
HLD	Pozastavení funkce bloku	Bool

Výstupy

amp	Amplituda základní frekvence (určená parametrem freq)	Double (F64)
thd	Celkové harmonické zkreslení, podíl základní a vyšších harmonických (jen pokud nharm ≥ 1)	Double (F64)
vAmp	Vektor amplitud pro zadané frekvence	Reference
vPhi	Vektor fázových posunů pro zadané frekvence	Reference
vRe	Vektor reálných částí pro zadané frekvence	Reference
vIm	Vektor imaginárních částí pro zadané frekvence	Reference
E	Příznak chyby	Bool
iE	Kód chyby	Error
	i ..... obecná chyba systému REXYGEN	

Parametry

`freq`	Základní frekvence	↓1e-09 ↑1e+09 ⊙1.0	Double (F64)
`nper`	Počet period signálu na kterých provádět výpočet	↓1 ↑10000 ⊙10	Long (I32)
`nharm`	Počet monitorovaných harmonických frekvencí	↓0 ↑16 ⊙3	Long (I32)
`ifrunit`	Jednotky pro frekvenci 1 ..... Hz 2 ..... rad/s	↓1 ↑2 ⊙1	Long (I32)
`iphunit`	Jednotky pro fázový posun 1 ..... stupně 2 ..... radiány	↓0 ↑2 ⊙1	Long (I32)
`nmax`	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙8192	Long (I32)
`freq2`	Vektor uživatelem definovaných frekvencí	⊙[2.0 3.0 4.0]	Double (F64)

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 tp a tn podle následujících vztahů:

maximální nárůst za sekundu:	$1/\texttt{tp}$
maximální pokles za sekundu:	$-1/\texttt{tn}$

Vstup

u	Vstupní signál filtru	Double (F64)

Výstup

y	Filtrovaný výstupní signál	Double (F64)

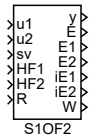
Parametry

tp	Časová konstanta určující maximální růst	⊙2.0	Double (F64)
tn	Časová konstanta určující maximální pokles (pozor, $\texttt{tn} > 0$)	⊙2.0	Double (F64)

S1OF2 – Výběr jednoho ze dvou analogových vstupů

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok S1OF2 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 S1OF2 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

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

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:

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

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.

Vstupy

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

Výstupy

`y`	Analogový výstupní signál	`Double (F64)`
`E`	Indikátor neplatnosti výstupního signálu `y` `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`, viz výstup `iE1`	`Long (I32)`
`W`	Varování (neplatný vstupní signál) `off` ... oba vstupní signály jsou platné `on` .... alespoň jeden vstupní signál je neplatný	`Bool`

Parametry

`nb`	Počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálů `u1` a `u2` ⊙10	`Long (I32)`

`nc`	Počet vzorků pro testování neměnnosti (viz blok SAI, podmínka `F2`) ⊙`10`	`Long (I32)`
`nbits`	Počet bitů A/D převodníku vstupního modulu (zdroje signálů `u1` a `u2`) ⊙`12`	`Long (I32)`
`nr`	Počet vzorků pro testování variability (viz blok SAI, podmínka `F3`) ⊙`10`	`Long (I32)`
`prate`	Maximální předpokládaná procentuální změna vstupu `u1` (`u2`) z celkového rozsahu `vmax` − `vmin` za `nr` vzorků vstupu `u1` (`u2`), viz blok SAI ⊙`10.0`	`Double (F64)`
`nv`	Počet vzorků pro testování překročení rozsahu (viz blok SAI, podmínka `F4`) ⊙`1`	`Long (I32)`
`vmin`	Spodní omezení na vstupní signál ⊙`-1.0`	`Double (F64)`
`vmax`	Horní omezení na vstupní signál ⊙`1.0`	`Double (F64)`
`nd`	Počet vzorků pro vyhodnocování odchýlení (vnitřní příznak `D`, pro `nd` = 0 je vždy `D` = `off`) ⊙`5`	`Long (I32)`
`pdev`	Maximální povolená procentuální odchylka signálů `u1` a `u2` z celkového rozsahu `vmax` − `vmin` ⊙`10.0`	`Double (F64)`
`mode`	Způsob výpočtu výstupu při platnosti obou vstupů (`E1` = `off`, `E2` = `off` a `D` = `off`) ⊙`1` `1` ..... průměr, `y` = $\frac{\texttt{u1}+\texttt{u2}}{2}$ `2` ..... minimum, `y` = min(`u1`, `u2`) `3` ..... maximum, `y` = max(`u1`, `u2`)	`Long (I32)`

SAI – Zabezpečený analogový vstup

Symbol bloku

Licence: ADVANCED

Popis funkce

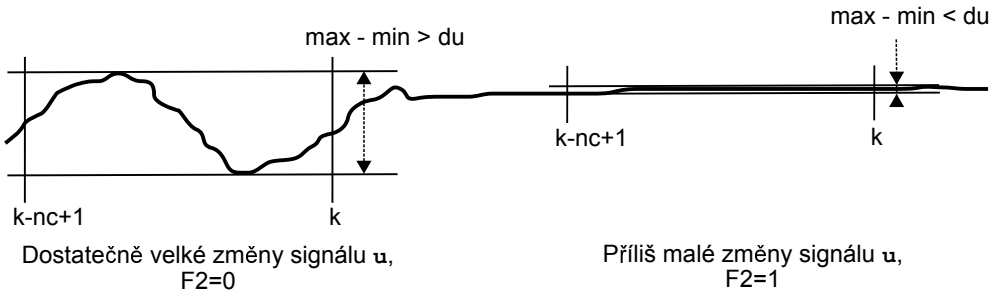
Blok `SAI` testuje vstupní signál `u` s cílem rozpoznání jeho platnosti. Vstupní signál `u` se považuje za neplatný (výstup `E` = `on`) v následujících případech:

`F1`: Hardwarová chyba. Vstupní signál `HWF` = `on`.

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

$$\texttt{du} = \begin{cases} \frac{\texttt{vmax}-\texttt{vmin}}{2^{\texttt{nbits}}}, & \text{pro } \texttt{nbits} \in \{8, 9, ..., 16\} \\ 0, & \text{pro } \texttt{nbits} \notin \{8, 9, ..., 16\}, \end{cases}$$

kde `vmin` a `vmax` jsou po řadě dolní a horní mez vstupu `u` a `nbits` je počet bitů příslušného A/D převodníku. Situace, kdy je splněna podmínka příliš malé změny `u`, je zobrazena na následujícím obrázku:



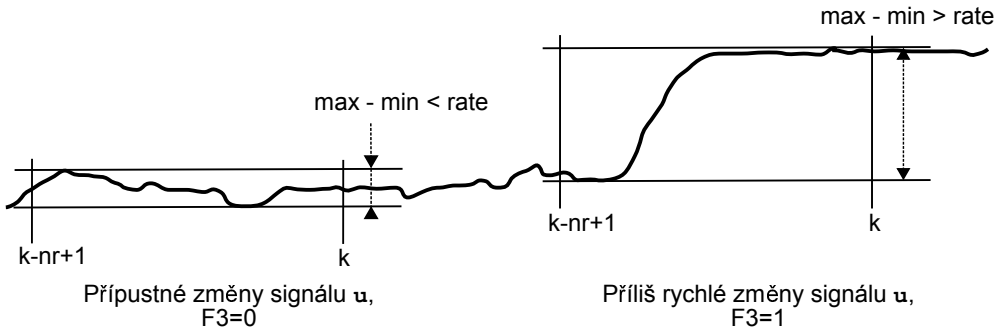
Jestliže je nastaveno `nc` = 0, potom podmínka `F2` není splněna nikdy.

`F3`: Vstupní signál `u` se mění příliš rychle. Posledních `nr` vzorků vstupu `u` filtrovaného filtrem `SPIKE` neleží v intervalu délky `rate`,

$$\texttt{rate} = \texttt{prate}\frac{\texttt{vmax} - \texttt{vmin}}{100},$$

kde `prate` vyjadřuje dovolenou procentuální změnu signálu `u` z celkového rozsahu během `nr` vzorků. V bloku je zařazený `SPIKE` filtr s pevnými parametry `mingap` = (`vmax` − `vmin`)/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, 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 ⟨`vmin`, `vmax`⟩. 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`→`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` ≥ 5 z důvodu odeznění počátečních podmínek SPIKE filtru.

Vstupy

`u`	Analogový vstupní signál	`Double (F64)`
`sv`	Náhradní hodnota při neplatném signálu `u`	`Double (F64)`

`HWF`	Příznak hardwarové chyby `off` ... vstupní modul signálu pracuje normálně `on` .... došlo k hardwarové chybě vstupního modulu	`Bool`
`R`	Vynulování vnitřních příznaků chyb `F1`–`F4`	`Bool`

Výstupy

`y`	Analogový výstupní signál	`Double (F64)`
`yf`	Výstupní signál `y` filtrovaný SPIKE algoritmem	`Double (F64)`
`E`	Indikátor neplatnosti výstupního signálu `off` ... výstup je platný `yf` = `sv` `on` .... výstup není platný, `y` =	`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)`

Parametry

`nb`	Počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálu `u`	⊙`10`	`Long (I32)`
`nc`	Počet vzorků pro testování neměnnosti (podmínka `F2`)	⊙`10`	`Long (I32)`
`nbits`	Počet bitů A/D převodníku vstupního modulu	⊙`12`	`Long (I32)`
`nr`	Počet vzorků pro testování variability (podmínka `F3`)	⊙`10`	`Long (I32)`
`prate`	Maximální předpokládaná procentuální změna vstupu `u` z celkového rozsahu (`vmax` − `vmin`) za `nr` vzorků vstupu `u`	⊙`10.0`	`Double (F64)`
`nv`	Počet vzorků pro testování překročení rozsahu (podmínka `F4`)	⊙`1`	`Long (I32)`
`vmin`	Spodní omezení na vstupní signál `u`	⊙`-1.0`	`Double (F64)`
`vmax`	Horní omezení na vstupní signál `u`	⊙`1.0`	`Double (F64)`

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 SW*n*.

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:

iSW	SW1	SW2	y
0	off	off	u1
1	off	on	u2
2	on	off	u3
3	on	on	u4

Vstupy

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)
u3	Třetí analogový vstup bloku	Double (F64)
u4	Čtvrtý analogový vstup bloku	Double (F64)
iSW	Selektor aktivního signálu, který je aktivní, je-li BINF = off	Long (I32)
SW1	Binární vstup pro výběr, který je aktivní, je-li BINF = on	Bool
SW2	Binární vstup pro výběr, který je aktivní, je-li BINF = on	Bool

Výstup

y	Zvolený signál	Double (F64)

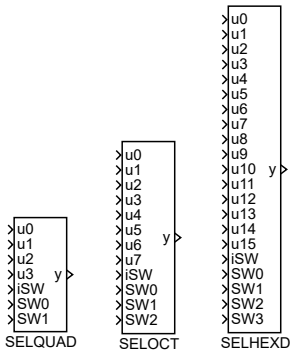
Parametr

BINF	Výběr pomocí binárních vstupů off ... zakázáno (výběr přes iSW) on .... povoleno (výběr přes SW1 a SW2)	Bool

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

Vstupy

u0..15	Analogové vstupní signály	Double (F64)

`iSW`	Selektor aktivního signálu	`Long (I32)`
`SW0..3`	Binární vstupy pro výběr	`Bool`

Výstup

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

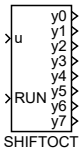
Parametr

`BINF`	Výběr pomocí binárních vstupů `off` ... zakázáno (výběr přes `iSW`) `on` .... povoleno (výběr přes `SW`n)	`Bool`

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:

$$\begin{aligned} \mathtt{y}_i &= \mathtt{y}_{i-1},\ i = 1..7, \\ \mathtt{y0} &= \mathtt{u}, \end{aligned}$$

tedy hodnota na každém z výstupů `y0` až `y6` je posunuta na výstup v pořadí následující, a hodnota vstupu `u` je přenesena na výstup `y0`.

Blok pracuje s libovolným datovým typem signálu přivedeného na vstup `u`. Požadovaný datový typ je třeba nastavit parametrem `vtype`. Výstupy `y0` až `y7` jsou pak shodného datového typu.

Pokud potřebujete posun dat v registru provádět na základě triggeru, vložte před vstup `RUN` blok EDGE_.

Vstupy

`u`	Vstupní hodnota registru	`Any`
`RUN`	Povoluje posun výstupů	`Bool`

Výstupy

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

Parametry

`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)	

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 y0 a má prioritu před vstupem SETH.

Zvažte též použití bloku PARR, který může být rovněž použit pro uložení číselné hodnoty.

Vstupy

u	Analogový vstupní signál		Double (F64)
SETH	Vstup pro nastavení a podržení výstupní hodnoty		Bool
R1	Reset bloku, R1 = on → y = y0		Bool

Výstup

y	Analogový výstupní signál		Double (F64)

Parametr

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

SINT – Jednoduchý integrátor

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SINT realizuje diskrétní integrátor popsaný diferenční rovnicí

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

kde T_S je perioda spouštění bloku a ti 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 ymin, ymax, potom je výstup i stav integrátoru příslušně omezen.

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

Vstup

u	Analogový vstupní signál	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

ti	Integrační časová konstanta	⊙1.0	Double (F64)
y0	Počáteční hodnota výstupu		Double (F64)
ymax	Horní saturační mez	⊙1.0	Double (F64)
ymin	Dolní saturační mez	⊙-1.0	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.

Vstup

u	Vstupní signál filtru	Double (F64)

Výstup

y	Filtrovaný výstupní signál	Double (F64)

Parametry

`mingap`	Minimální velikost tolerančního okénka	⊙`0.01`	`Double (F64)`
`q`	Rychlost adaptace tolerančního okénka filtru	↓`1.0` ⊙`2.0`	`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).

Vstupy

u1	První vstup bloku	Any
u2	Druhý vstup bloku	Any
SW	Přepínací signál off ... je zvolen první vstupní signál, y = u1 on .... je zvolen druhý vstupní signál, y = u2	Bool

Výstup

y	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 vybraný 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.

Vstupy

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

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

t1	Časová konstanta omezovače strmosti, nabíhání na hodnotu u1 ⊙1.0	Double (F64)
t2	Časová konstanta omezovače strmosti, nabíhání na hodnotu u2 ⊙1.0	Double (F64)
y0	Počáteční hodnota výstupu, ze které se vysledovává před prvním přepnutím	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 spouštění bloku. Jestliže po spuštění bloku není dosud k dispozici n posledních vzorků, potom je výstup y nastaven na inicializační hodnotu y0.

Vstupy

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

Výstup

y	Zpožděný vstupní signál	Double (F64)

Parametr

y0	Počáteční/náhradní hodnota výstupu	Double (F64)
nmax	Délka vyrovnávací paměti pro dopravní zpoždění d (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙1000	Long (I32)

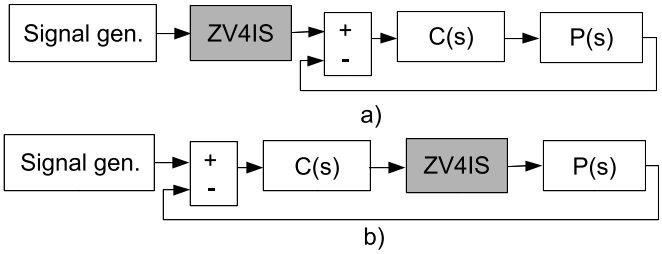
ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací

Symbol bloku

Licence: ADVANCED

Popis funkce

Blok ZV4IS realizuje funkci frekvenčního filtru typu pásmová zádrž. 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 opotřebení 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 $P(s)$). Záporem 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.

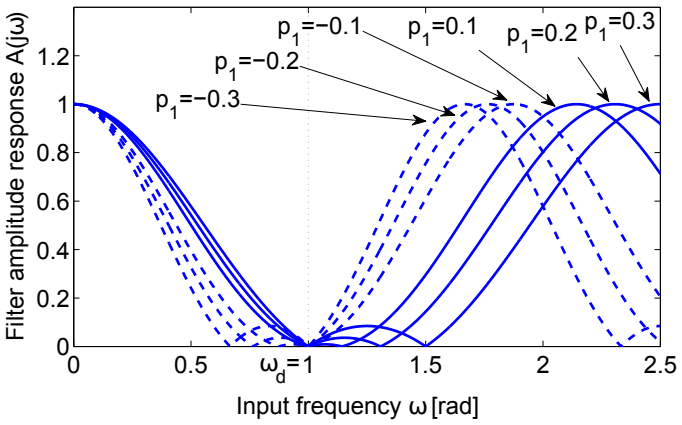
Vlastní algoritmus filtrace lze popsat v časové oblasti vztahem

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

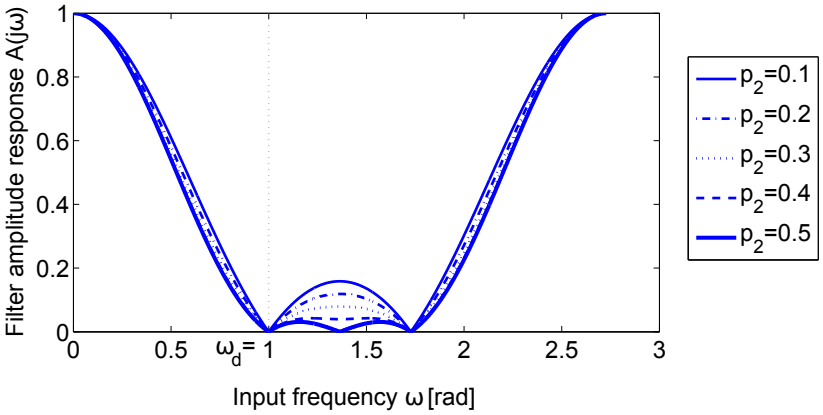
Filtr má tedy strukturu sumy vážených dopravních zpoždění kde zesílení $A_1..A_4$ a hodnoty zpoždění $t_1..t_4$ závisí na volbě typu filtru, frekvenci a tlumení kmitavého módu systému. Výhodou uvedené struktury oproti klasickým dynamickým notch filtrům užívaným v regulační technice je konečná impulzní 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 `ipar` pak udává typ tvarovacího filtru, pro `ipar` = `1` je použit jeden z deseti základních filtrů, které se volí parametrem `istype`. 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).

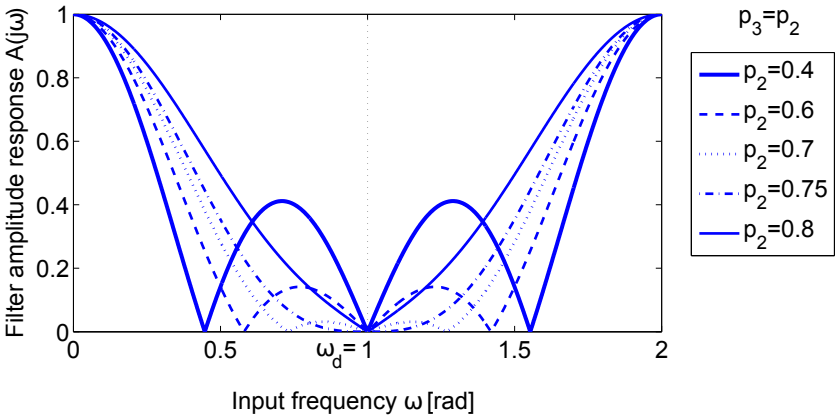
Pro jemné ladění filtru lze použít kompletní parametrizaci volbou `ipar` = `2`, pro kterou se zpřístupní volné parametry `p_alpha`,`p_a2` a `p_a3`. Ty určují tvar frekvenční charakteristiky filtru a lze je použít pro nalezení optimálního kompromisu mezi robustností a zavedeným zpožděním.



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).



Parametr necitlivosti `p_a2` určuje šířku a úroveň útlumu nepropustného pásma filtru. Větší hodnota znamená širší nepropustnou oblast s větším tlumením. Pro praktické aplikace je doporučeno nastavit hodnotu `p_a2` $= 0.5$ pro maximální robustnost tvarovacího filtru vůči chybě v modelu systému/signálu.



Doplňkový parametr `p_a3` je nutné nastavit pro symetrické filtry (volba `p_alpha` $=$ 0). V tomto případě je pro praktické použití vhodné volit *shodné hodnoty* `p_a2` $=$ `p_a3` na intervalu $< 0, 0.75 >$. Menší hodnoty vedou na rychlejší filtry s úzkým nepropustným pásmem, větší pak na robustní tvarovače s širokým pásmem útlumu a delší odezvou (viz obrázek).

Vstup

`u`	Vstupní signál filtru	`Double (F64)`

Výstupy

`y`	Filtrovaný výstupní signál	`Double (F64)`
`E`	Příznak chyby	`Bool`
	`off` ... bez chyby `on` .... nastala chyba	

Parametry

`omega`	Vlastní frekvence	⊙`1.0`	`Double (F64)`
`xi`	Součinitel relativního tlumení		`Double (F64)`
`ipar`	Specifikace `1` ..... základní typy tvarovačů signálu `2` ..... kompletní parametrizace	⊙`1`	`Long (I32)`
`istype`	Typ `1` ..... ZV `2` ..... ZVD `3` ..... ZVDD `4` ..... MISZV `5` ..... UEI1 `6` ..... UEI2 `7` ..... UEI5 `8` ..... UTHEI1 `9` ..... UTHEI2 `10` UTHEI5	⊙`2`	`Long (I32)`
`p_alpha`	Asymetrie tvarovače	⊙`0.2`	`Double (F64)`
`p_a2`	Necitlivost	⊙`0.5`	`Double (F64)`
`p_a3`	Přídavný parametr (pouze pro $p\_alpha = 0$)	⊙`0.5`	`Double (F64)`
`nmax`	Délka vyrovnávací paměti (počet vzorků). Používá se pro interní alokaci paměti.	↓`10` ↑`10000000` ⊙`1000`	`Long (I32)`

Kapitola 6

GEN – Generátory signálů

Obsah

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 t1,y1; t2,y2; t3,y3; t4,y4. Počáteční hodnota y je definována parametrem y0. Start generování funkce (časový okamžik 0) je určen náběžnou hranou vstupu RUN. V intervalu $\langle t_i, t_{i+1}\rangle, i = 0, \ldots, 3, t_0 = 0$ 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 = off (výstup je resetován na y0 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 $t_1 < \ldots < t_i$. 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

y0	Počáteční hodnota výstupu		Double (F64)
t1	Čas uzlového bodu 1	⊙1.0	Double (F64)
y1	Hodnota uzlového bodu 1		Double (F64)
t2	Čas uzlového bodu 2	⊙1.0	Double (F64)
y2	Hodnota uzlového bodu 2	⊙1.0	Double (F64)
t3	Čas uzlového bodu 3	⊙2.0	Double (F64)
y3	Hodnota uzlového bodu 3	⊙1.0	Double (F64)
t4	Čas uzlového bodu 4	⊙2.0	Double (F64)

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

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 t1, t2,...,t8 se mění hodnota výstupu Y na hodnotu opačnou (off→on, on→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. 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/nepravda 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 `off` ... zakázáno `on` .... povoleno		`Bool`

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 t1, t2,...,t8 se mění hodnota výstupu Y na hodnotu opačnou (off→on, on→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 off ... vypnuto/nepravda 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 off ... zakázáno on .... povoleno		Bool

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 RPFT = off, je nahození BSTATE během generování výstupního pulzu 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

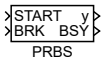
Parametry

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

PRBS – Pseudonáhodná binární posloupnost

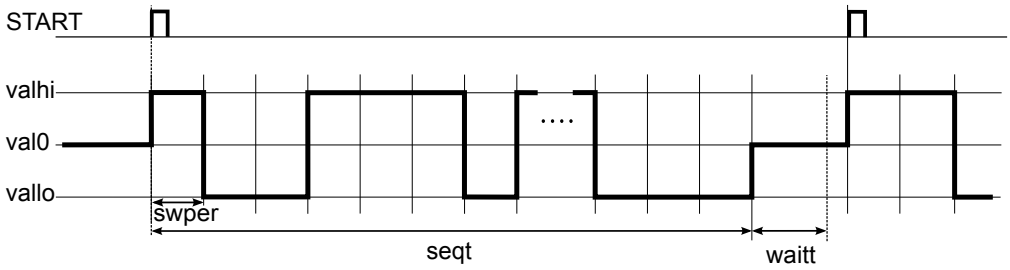
Symbol bloku

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

`START`	Spouštěcí signál (náběžná hrana)	`Bool`
`BRK`	Signál pro přerušení	`Bool`

Výstupy

`y`	Vygenerovaná pseudonáhodná binární posloupnost	`Double (F64)`
`BSY`	Příznak probíhající operace	`Bool`

Parametry

`val0`	Počáteční a koncová hodnota		`Double (F64)`
`valhi`	Horní hladina výstupu `y`	⊙`1.0`	`Double (F64)`
`vallo`	Dolní hladina výstupu `y`	⊙`-1.0`	`Double (F64)`

`swper`	Perioda náhodného přepínání výstupu y mezi hladinami [s]	⊙1.0	`Double (F64)`
`swprob`	Pravděpodobnost přepnutí	↓0.0 ↑1.0 ⊙0.2	`Double (F64)`
`seqt`	Délka posloupnosti [s]	⊙10.0	`Double (F64)`
`waitt`	Doba ustáleni [s]	⊙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 parametr 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

RUN	Povolení běhu algoritmu, spuštění generátoru	Bool
SYN	Synchronizační signál	Bool

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

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

`iphunit`	Jednotky pro fázový posun `1` ..... stupně `2` ..... radiány	⊙`1` `Long (I32)`

Kapitola 7

REG – Bloky pro regulaci

Obsah

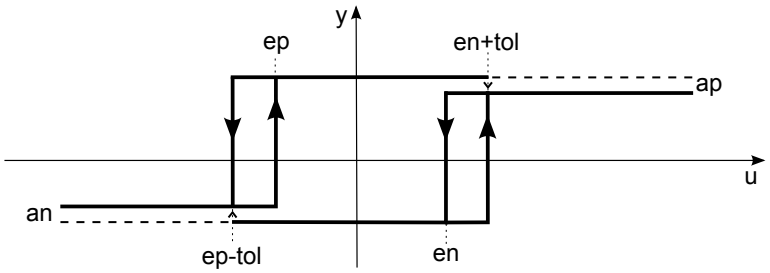
ARLY – Relé s předstihem

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

FLCU – Fuzzy regulátor

Symbol bloku

Licence: ADVANCED

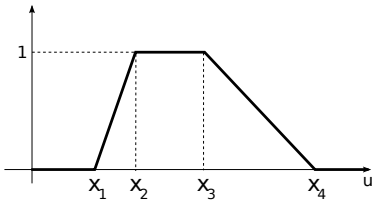
Popis funkce

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:

$$\text{Jestliže (u je } U_i\text{) AND (v je } V_j\text{), potom (y je } Y_k\text{)},$$

kde $U_i, i = 1, \ldots, \mathtt{nu}$ jsou jazykové výrazy příslušné ke vstupu u; $V_j, j = 1, \ldots, \mathtt{nv}$ jsou jazykové výrazy příslušné ke vstupu v a $Y_k, k = 1, \ldots, \mathtt{ny}$ 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 $x_1, \ldots, x_4$ vesměs různá. Matice funkcí příslušnosti vstupů u a v se potom skládají z řádků $[x_1, x_2, x_3, x_4]$. Matice mfu a mfv jsou tedy po řadě typu $(\mathtt{nu} \times 4)$ a $(\mathtt{nv} \times 4)$.

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

$$\mathtt{y}_k,\ a_k,\ b_k,$$

kde $\mathtt{y}_k$ je hodnota výstupu přiřazená jazykovému výrazu $Y_k, k = 1, \ldots, \mathtt{ny}$ 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 $\mathtt{y}_k + a_k\mathtt{u} + b_k\mathtt{v}$. Matice funkcí příslušnosti výstupu sty je typu $(\mathtt{ny} \times 3)$ a skládá se po řadě z řádků $[\mathtt{y}_k, a_k, b_k], k = 1, \ldots, \mathtt{ny}$.

Soubor pravidel se skládá též jako matice a její řádky jsou $[i_l, j_l, k_l, w_l], l = 1, \ldots, \mathtt{nr}$, 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 $w_l \in \{0, 1, \ldots, 100\}$. 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 reservovaných 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)`

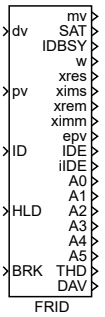
Výstupy

`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`	Stejnosměrná 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 ..... Ruční volba frekvencí 2 ..... Lineárně nmw frekvencí v intervalu <wb,wf> 3 ..... Logaritmicky nmw frekvencí v intervalu <wb,wf> 4 ..... Automatická detekce důležitých frekvencí (N/A)	⊙1	Long (I32)
nwm	Počet frekvencí v automatickém režimu		Long (I32)
wm	Pole frekvencí pro ruční režim [array of rad/s]	⊙[2.0 4.0 6.0 8.0]	Double (F64)

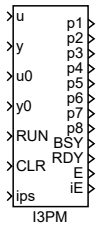
Výstupy

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 $mu0$
  - p2 ... parametr $mu1$
  - p3 ... parametr $mu2$
  - p4 ... parametr $my0$
  - p5 ... parametr $my1$
  - p6 ... parametr $my2$
- 2 ..... momenty procesu
  - p1 ... parametr $mp0$
  - p2 ... parametr $mp1$
  - p3 ... parametr $mp2$
- 3 ..... charakteristická čísla procesu
  - p1 ... parametr κ
  - p2 ... parametr μ
  - p3 ... parametr σ^2
  - p4 ... parametr σ

Výstupy

`p`i	Identifikované parametry v závislosti na `ips`, $i = 1, ..., 8$	`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` ..... `mu0` $= 0$	
	`3` ..... `mp0` $= 0$	
	`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 diskrétní simulátor přenosu derivačního článku

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

kde td je derivační konstanta a nd je parametr určující vliv parazitního filtru prvního řádu. Doporučená hodnota nd je $2 \leq \mathtt{nd} \leq 10$. 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 off ... nulový počáteční stav on .... ustálený počáteční stav		Bool

LLC – Integračně-derivační kompenzátor

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LLC realizuje diskrétní simulátor přenosu integračně-derivačního článku

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

kde tau je časová konstanta jmenovatele a její a-násobek (a * tau) je časová konstanta čitatele. Je-li ISSF = on, potom je stav integračního článku 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

tau	Časová konstanta	⊙1.0	Double (F64)
a	Koeficient pro výpočet časové konstanty čitatele		Double (F64)
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav		Bool

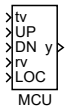
Výstup

y	Analogový výstupní signál	Double (F64)

MCU – Jednotka pro ruční zadávání

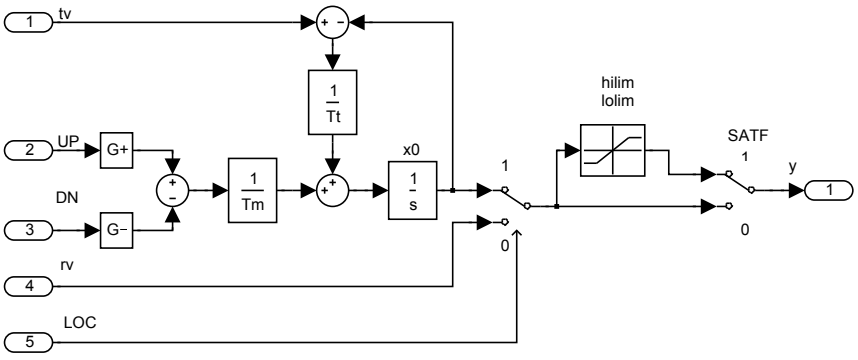
Symbol bloku

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 `y0` na žádanou hodnotu je určena integrační konstantou `tm` a dobou stlačení ovládacích tlačítek. Po uplynutí každých `ta` sekund je strmost vždy násobena faktorem `q`, až do vypršení doby `tf`. 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)`

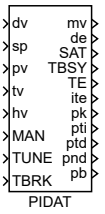
Parametry

`tt`	Časová konstanta vysledování 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 `off` ... signál není omezen aktivní `on` .... saturační meze jsou		`Bool`
`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 `off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	`Bool`
`TBSY`	Příznak probíhajícího ladicího experimentu	`Bool`
`TE`	Příznak chyby během ladění `off` ... Ladění proběhlo bez chyby `on` .... Během ladění se vyskytla chyba	`Bool`
`ite`	Kód chyby (během probíhajícího ladícího experimentu očekávaný čas v sekundách do jeho konce) `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	`Long (I32)`
`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 `1` ..... D `4` ..... P `7` ..... PID `2` ..... I `5` ..... PD `3` ..... ID `6` ..... PI	`Long (I32)`

`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
		`Bool`

`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)`

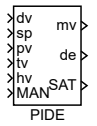
`b`	Váhový faktor pro proporcionální složku	↓`0.0` ⊙`1.0`	`Double (F64)`
`c`	Váhový faktor pro derivační složku	↓`0.0`	`Double (F64)`
`tt`	Časová konstanta vysledování.	↓`0.0` ⊙`1.0`	`Double (F64)`
`hilim`	Horní mez akčního zásahu regulátoru	⊙`1.0`	`Double (F64)`
`lolim`	Dolní mez akčního zásahu regulátoru	⊙`-1.0`	`Double (F64)`
`iainf`	Druh apriorní informace `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	⊙`1`	`Long (I32)`
`k0`	Statické zesílení procesu (musí být zadáno v případě `iainf` = 3, 4, 5)	⊙`1.0`	`Double (F64)`
`n1`	Maximální počet půlperiod pro nalezení bodu frekvenční charakteristiky	⊙`20`	`Long (I32)`
`mm`	Maximální počet půlperiod pro průměrování	⊙`4`	`Long (I32)`
`amp`	Amplituda reléového regulátoru	⊙`0.1`	`Double (F64)`
`uhys`	Hystereze reléového regulátoru		`Double (F64)`
`ntime`	Čas vymezený pro odhad amplitudy šumu na počátku experimentu [s]	⊙`5.0`	`Double (F64)`
`rerrap`	Ukončovací hodnota relativní chyby amplitudy kmitů	⊙`0.1`	`Double (F64)`
`aerrph`	Ukončovací hodnota absolutní chyby fáze odhadovaného bodu	⊙`10.0`	`Double (F64)`
`maxdev`	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 ε, 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)`

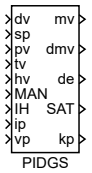
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`Bool`

`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 vysledování. Pro regulátory bez integrační složky nemá žádný význam.	↓`0.0` ⊙`1.0`	`Double (F64)`
`bp`	Hodnota regulační odchylky, která způsobí, že výstup regulárotu `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 thsha. 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 ⊙`1.0`	`Double (F64)`
`lolim`	Dolní mez akčního zásahu regulátoru ⊙`-1.0`	`Double (F64)`
`dz`	Pásmo necitlivosti	`Double (F64)`
`icotype`	Typ výstupu regulátoru ⊙`1` `1` ..... analogový výstup `2` ..... šířkově modulovaný výstup (PWM) `3` ..... krokový regulátor s polohovou zpětnou vazbou (SCU) `4` ..... krokový regulátor bez polohové zpětné vazby (SCUV)	`Long (I32)`
`nmax`	Alokovaný počet sad parametrů ↓`4` ↑`10000` ⊙`10`	`Long (I32)`
`GSCF`	Přepínání parametrů podle vstupu `vp` `off` ... přepínání indexem sady parametrů `on` .... přepínání analogovým signálem	`Bool`
`hys`	Hystereze pro přepínání podle vstupu `vp`	`Double (F64)`
`irtypea`	Vektor typů regulátoru ⊙`[6 6 6 6 6 6]` `1` ..... D `4` ..... P `7` ..... PID `2` ..... I `5` ..... PD `3` ..... ID `6` ..... PI	`Byte (U8)`

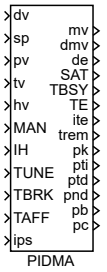
`RACTA`	Vektor příznaků obráceného působení výstupu regulátoru ⊙`[0 0 0 0 0 0]`
	`0` ..... vyšší `mv` → vyšší `pv`
	`1` ..... vyšší `mv` → nižší `pv`
		`Bool`

`ka`	Vektor zesílení regulátoru K ⊙`[1.0 1.0 1.0 1.0 1.0 1.0]`	`Double (F64)`
`tia`	Vektor integračních časových konstant T_i ⊙`[4.0 4.0 4.0 4.0 4.0 4.0]`	`Double (F64)`
`tda`	Vektor derivačních časových konstant T_d ⊙`[1.0 1.0 1.0 1.0 1.0 1.0]`	`Double (F64)`
`nda`	Vektor parametrů filtru derivační složky N ⊙`[10.0 10.0 10.0 10.0 10.0 10.0]`	`Double (F64)`
`ba`	Váhové faktory pro proporcionální složku ⊙`[1.0 1.0 1.0 1.0 1.0 1.0]`	`Double (F64)`
`ca`	Váhové faktory pro derivační složku ⊙`[0.0 0.0 0.0 0.0 0.0 0.0]`	`Double (F64)`
`tta`	Vektor časových konstant vysledování ⊙`[1.0 1.0 1.0 1.0 1.0 1.0]`	`Double (F64)`
`thrsha`	Vektor mezí přepínací veličiny ⊙`[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 (`MAN` = `off`) realizuje blok `PIDMA` řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s}\left[W(s) - Y(s)\right] + \frac{T_d s}{\frac{T_d}{N}s + 1}\left[cW(s) - Y(s)\right] \right\} + Z(s)$$

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` = `on`. Pro správnou funkci regulátoru je nutné propojit výstup regulátoru `mv` se vstupem `tv` a správně nastavit časovou konstantu vysledovávání `tt`

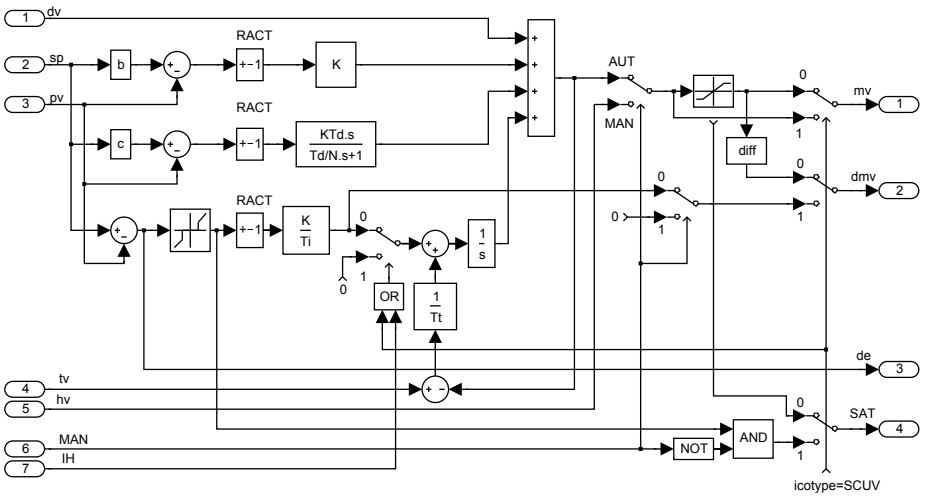
Doporučená výchozí hodnota pro PID regulátor je `tt` $\approx \sqrt{T_i T_d}$, pro PI regulátor pak `tt` $\approx T_i/2$. Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu `mv` (tzv. antiwindup). Úpravou parametru `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 `PIDMA` 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 PIDMA a SCUV se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku c rovný nule.

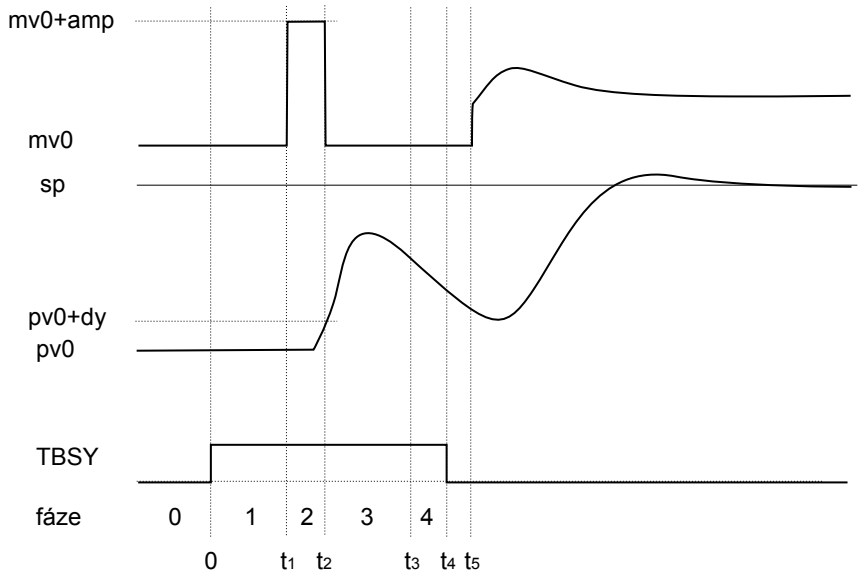
V manuálním režimu (MAN = on) je vstup hv kopírován na výstup mv. Celková regulační funkce bloku PIDMA je zřejmá z následujícího obrázku.



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 (zadává se vždy jako kladné číslo). Pokud je nastaven příznak DGC, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajícího 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 odeznívání 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

`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`
`TUNE`	Zahájení ladicího experimentu nebo vynucení přechodu do další fáze 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 pk, pti, ptd, pnd, pb a pc 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	Long (I32)

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
TBSY	Příznak probíhajícího ladicího experimentu	Bool
TE	Příznak chyby během ladění off ... Ladění proběhlo bez chyby on .... Během ladění se vyskytla chyba	Bool
ite	Kód chyby *Kódy chyb ladění (po experimentu):* 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 přerušení experimentu uživatelem 12 .... nesprávný směr řídicí veličiny (změňte parametr RACT) 100 ... ruční ukončení ladění (varování)	Long (I32)

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`).

`trem`	Odhad času do ukončení experimentu [s]	`Double (F64)`
`pk`	Navržené zesílení regulátoru K (`ips` = `0`)	`Double (F64)`
`pti`	Navržená integrační časová konstanta regulátoru T_i (`ips` = `0`)	`Double (F64)`
`ptd`	Navržená derivační časová konstanta regulátoru T_d (`ips` = `0`)	`Double (F64)`
`pnd`	Navržený parametr filtru derivační složky N (`ips` = `0`)	`Double (F64)`
`pb`	Navržený váhový faktor pro proporcionální složku (`ips` = `0`)	`Double (F64)`
`pc`	Navržený váhový faktor pro derivační složku (`ips` = `0`)	`Double (F64)`

Parametry

`irtype`	Typ regulátoru ⊙`6` `1` ..... D `3` ..... ID `5` ..... PD `7` ..... PID `2` ..... I `4` ..... P `6` ..... PI	`Long (I32)`

`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
		`Bool`

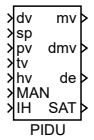
`k`	Zesílení regulátoru K. Hodnota 0 (dle definice) vypne regulátor, záporné hodnoty nejsou dovoleny (k tomu slouží parametr `RACT`). ↓`0.0` ⊙`1.0`	`Double (F64)`
`ti`	Integrační časová konstanta T_i. Hodnota 0 znamená vypnutí integrační složky regulátoru (stejný efekt jako vypnutí parametrem `irtype`). ↓`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`	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 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` `1` ..... analogový výstup `2` ..... šířkově modulovaný výstup (`PWM`) `3` ..... krokový regulátor s polohovou zpětnou vazbou (`SCU`) `4` ..... krokový regulátor bez polohové zpětné vazby (`SCUV`)	`Long (I32)`
`ittype`	Požadovaný typ regulátoru pro návrh ⊙`6` `6` ..... PI regulátor `7` ..... PID regulátor	`Long (I32)`
`iainf`	Druh apriorní informace ⊙`1` `1` ..... proces bez integrátoru `2` ..... proces s integračním chováním	`Long (I32)`
`DGC`	Kompenzace gradientu trendu ⊙`on` `off` ... zakázáno `on` .... povoleno	`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 ⊙`2` `1` ..... požadována pomalá uzavřená smyčka `2` ..... požadována středně rychlá uzavřená smyčka `3` ..... požadována rychlá uzavřená smyčka	`Long (I32)`
`ipid`	Forma PID regulátoru ⊙`1` `1` ..... paralelní realizace `2` ..... sériová realizace	`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

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s}\left[W(s) - Y(s)\right] + \frac{T_d s}{\frac{T_d}{N}s + 1}\left[cW(s) - Y(s)\right] \right\} + Z(s)$$

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 vysledovávání tt.

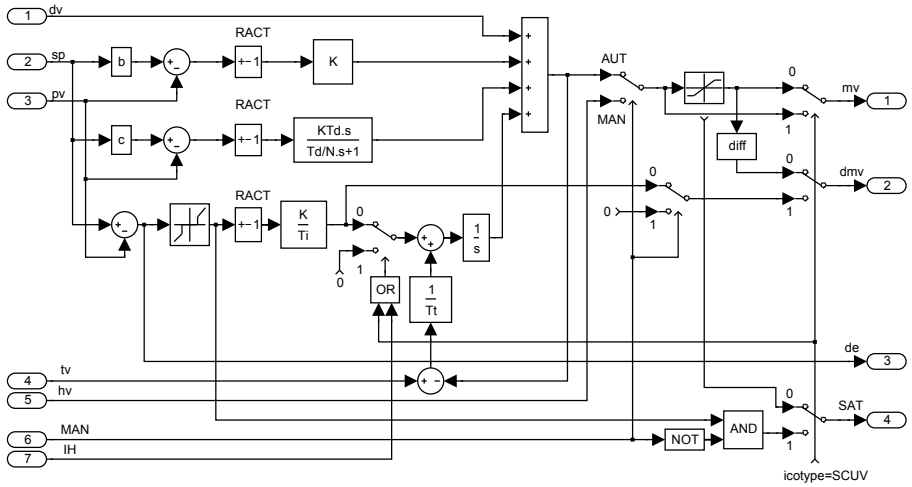
Doporučená výchozí hodnota pro PID regulátor je $\texttt{tt} \approx \sqrt{T_i T_d}$, pro PI regulátor pak $\texttt{tt} \approx T_i/2$. Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu mv (tzv. antiwindup). Úpravou 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 nenaráží 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 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`

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

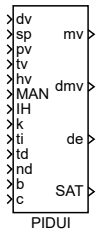
Parametry

irtype	Typ regulátoru ⊙6 1 ..... D 4 ..... P 7 ..... PID 2 ..... I 5 ..... PD 3 ..... ID 6 ..... PI	Long (I32)
RACT	Převrácené působení výstupu regulátoru off ... vyšší mv → vyšší pv on .... vyšší mv → nižší pv	Bool
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	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 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 1 ..... analogový výstup 2 ..... šířkově modulovaný výstup (PWM) 3 ..... krokový regulátor s polohovou zpětnou vazbou (SCU) 4 ..... krokový regulátor bez polohové zpětné vazby (SCUV)	Long (I32)

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`	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`
`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 (diference)	`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`

`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` ..... 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)`

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

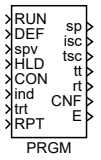
Parametry

dtime	Minimální trvání výstupního pulzu [s]	⊙1.0	Double (F64)
btime	Minimální prodleva mezi sousedními výstupními pulzy [s]	⊙1.0	Double (F64)

PRGM – Programátor

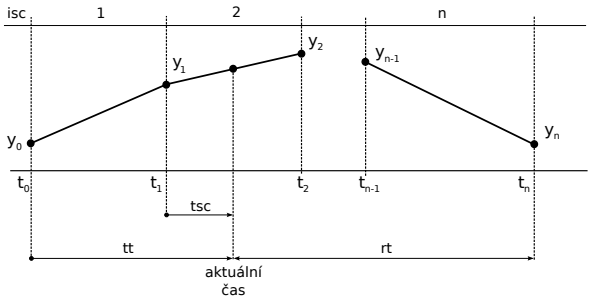
Symbol bloku

Licence: STANDARD



Popis funkce

Blok PRGM je určen pro generování časových funkcí (programů) složených z n lineárních částí definovaných (n + 1) rozměrnými vektory tm = $[t_0, ..., t_n]$ času a požadovaných hodnot y = $[y_0, ..., y_n]$ (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 sp na hodnotu spv	Bool
spv	Inicializační hodnota	Double (F64)
HLD	Zmrazení výstupu a výstupních časů	Bool
CON	Pokračování od uzlového bodu ind	Bool

`ind`	Index uzlového bodu pro pokračování	`Long (I32)`
`trt`	Čas pro dosažení požadovaného uzlu `ind`	`Double (F64)`
`RPT`	Příznak opakování generování časové funkce	`Bool`

Výstupy

`sp`	Požadovaná hodnota (hodnota časové funkce v daném čase)	`Double (F64)`
`isc`	Aktuální sektor funkce	`Long (I32)`
`tsc`	Čas od začátku sektoru	`Double (F64)`
`tt`	Čas od startu generování časové funkce	`Double (F64)`
`rt`	Čas do konce programu	`Double (F64)`
`CNF`	Příznak sledování nakonfigurované křivky	`Bool`
`E`	Chyba, časy uzlů nejsou seřazeny vzestupně	`Bool`

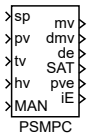
Parametry

`nmax`	Maximalní (alokovaná) délka pole	↓4 ↑10000000 ⊙10	`Long (I32)`
`tmunits`	Jednotky pro zadávání časů `1` ..... sekundy `2` ..... minuty `3` ..... hodiny	⊙1	`Long (I32)`
`tm`	(n + 1)-rozměrný vektor vzestupně uspořádaných časů	⊙[0 1 2]	`Double (F64)`
`y`	(n + 1)-rozměrný vektor hodnot časové funkce	⊙[0 1 0]	`Double (F64)`

PSMPC – Prediktivní „pulse-step“ regulátor

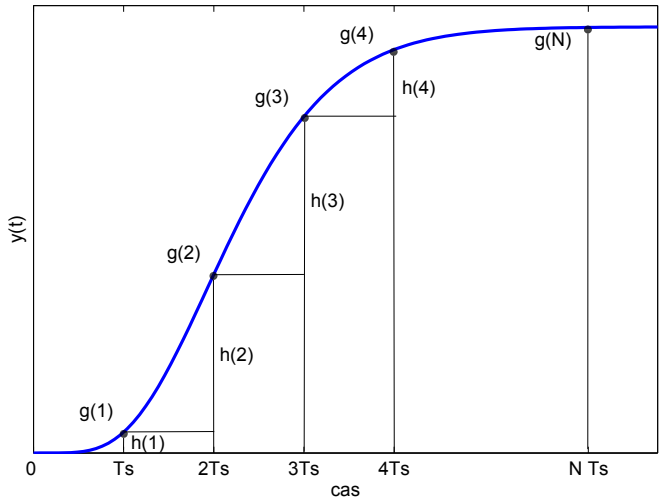
Symbol bloku

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). Zvlášť 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 $g(j), j = 1, ..., N$. 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

κ, μ a σ^2, které je možno určit z jednoduchého pulzního experimentu. Řízený systém pak aproximujeme buď přenosem prvého řádu s dopravním zpožděním

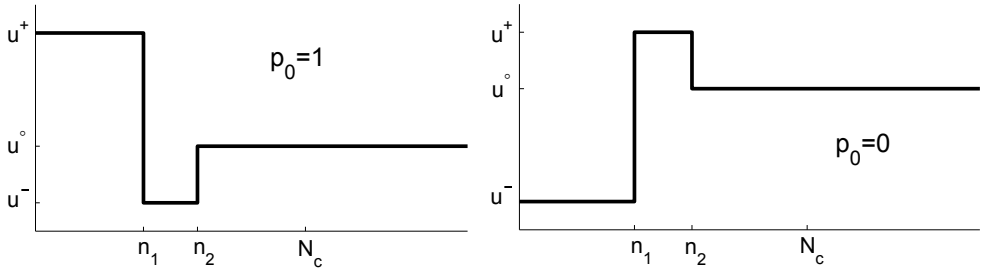
$$F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = \tau + D, \quad \sigma^2 = \tau^2 \tag{7.1}$$

nebo přenosem druhého řádu s dopravním zpožděním

$$F_{SOPDT}(s) = \frac{K}{(\tau s + 1)^2} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = 2\tau + D, \quad \sigma^2 = 2\tau^2 \tag{7.2}$$

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 $n_1, n_2 \in \{0, \ldots, N_C\}$ a $u^\infty \in \langle u^-, u^+ \rangle$, kde $N_C \in \{0, 1, \ldots\}$ 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^∞) spočívá v minimalizaci kritéria

$$I = \sum_{i=N_1}^{N_2} \hat{e}(k+i|k)^2 + \lambda \sum_{i=0}^{N_C} \Delta\hat{u}(k+i\,|k)^2 \to \min, \tag{7.3}$$

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 diference řídicího signálu na intervalu $i \in \{0, N_C\}$ a λ je koeficient penalizace změn akční veličiny. Pro nalezení optima úlohy (7.3) je použita kombinace metody nejmenších čtverců a hrubé síly. Hodnota u^∞ 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 prvý 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 λ. 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 μ, σ^2. Regulátor potom může být efektivně laděn „ručně“ pouze seřizováním charakteristických čísel procesu κ, μ, σ^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

`sp`	Požadovaná hodnota (setpoint)	`Double (F64)`
`pv`	Řízená veličina	`Double (F64)`
`tv`	Veličina pro vysledování (použitý řídicí signál)	`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)`
`dmv`	Rychlostní výstup regulátoru (diference)	`Double (F64)`
`de`	Regulační odchylka	`Double (F64)`
`SAT`	Saturace `off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	`Bool`
`pve`	Predikovaná hodnota regulované veličiny na základě zadaného modelu	`Double (F64)`
`iE`	Kód chyby `0` ..... bez chyby `1` ..... nesprávný FOPDT model `2` ..... nesprávný SOPDT model `3` ..... chyba v zadání přechodové charakteristiky	`Long (I32)`

Parametry

`nc`	Délka horizontu řízení (N_C)	⊙`5`	`Long (I32)`
`np1`	Začátek koincidenčního intervalu (N_1)	⊙`1`	`Long (I32)`
`np2`	Konec koincidenčního intervalu (N_2)	⊙`10`	`Long (I32)`
`lambda`	Koeficient penalizace změn řízení (λ)	⊙`0.05`	`Double (F64)`
`umax`	Horní mez akčního zásahu regulátoru (u^+)	⊙`1.0`	`Double (F64)`
`umin`	Dolní mez akčního zásahu regulátoru (u^-)	⊙`-1.0`	`Double (F64)`
`imtype`	Typ modelu řízené soustavy `1` ..... model prvního řádu `2` ..... model druhého řádu `3` ..... přechodová charakteristika	⊙`3`	`Long (I32)`
`kappa`	Statické zesílení (κ)	⊙`1.0`	`Double (F64)`
`mu`	Míra zpoždění soustavy (μ)	⊙`20.0`	`Double (F64)`

`sigma`	Míra délky odezvy soustavy ($\sqrt{\sigma^2}$)	⊙`10.0`	`Double (F64)`
`nmax`	Reservovaná velikost pole sr	↓`10` ↑`10000` ⊙`32`	`Long (I32)`
`sr`	Diskrétní přechodová charakteristika ($[g(1), \ldots, g(N)]$)		`Double (F64)`

⊙`[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:

$$L = \texttt{pertm} * |\texttt{u}|\,,$$

kde pertm je perioda modulace. Je-li $\texttt{u} > 0$ ($\texttt{u} < 0$), pulz je generován na výstupu UP (DN). Z praktických důvodů je však délka generovaného pulzu dále upravována podle zadaných parametrů bloku. Faktor asymetrie asyfac definuje poměr mezi délkou negativního pulzu DN a délkou pozitivního pulzu UP. Modifikované délky se počítají podle vztahů:

$$\text{jestliže} \quad \texttt{u} > 0 \quad \text{potom} \quad L(\texttt{UP}) := \begin{cases} L & \text{pro} \quad \texttt{asyfac} \leq 1.0 \\ L/\texttt{asyfac} & \text{pro} \quad \texttt{asyfac} > 1.0 \end{cases}$$

$$\text{jestliže} \quad \texttt{u} < 0 \quad \text{potom} \quad L(\texttt{DN}) := \begin{cases} L * \texttt{asyfac} & \text{pro} \quad \texttt{asyfac} \leq 1.0 \\ L & \text{pro} \quad \texttt{asyfac} > 1.0 \end{cases}$$

které pro libovolnou hodnotu $\texttt{asyfac} > 0$ zajišťují, že maximální délka generovaných pulzů je rovna pertm. Dále, jestliže vypočtená délka pulzu je menší než dtime, potom je výsledná délka nastavena na nulu. Jestliže se vypočtená délka pulzu liší od pertm méně než btime, potom je výsledná délka nastavena na 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 offtime. Jestliže $\texttt{SYNCH} = \texttt{on}$, potom změna vstupu u způsobí okamžitý přepočet 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

`pertm`	Perioda šířkové modulace [s]	⊙`10.0`	`Double (F64)`
`dtime`	Minimální trvání výstupního pulzu [s]	⊙`0.1`	`Double (F64)`
`btime`	Minimální prodleva mezi pulzy [s]	⊙`0.1`	`Double (F64)`
`offtime`	Minimální prodleva mezi pulzy opačné polarity [s]	⊙`1.0`	`Double (F64)`
`asyfac`	Faktor asymetrie	⊙`1.0`	`Double (F64)`
`SYNCH`	Synchronizační příznak pro začátek periody `off` ... synchronizace vypnuta `on` .... synchronizace zapnuta		`Bool`

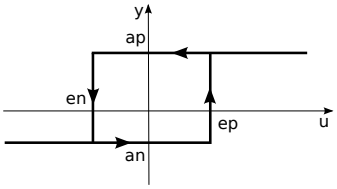
RLY – Relé s hysterezí

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“)	⊙1.0	Double (F64)
en	Hodnota u < en způsobí y = an („Vypnuto“)	⊙-1.0	Double (F64)
ap	Výstup ve stavu „Zapnuto“	⊙1.0	Double (F64)
an	Výstup ve stavu „Vypnuto“	⊙-1.0	Double (F64)
y0	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 ≤ u a u ≤ 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

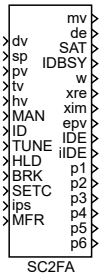
Parametry

tp	Časová konstanta rychlosti změn aktivních hodnot mezí v kladném směru	⊙1.0	Double (F64)
tn	Časová konstanta rychlosti změn aktivních hodnot mezí v záporném směru	⊙1.0	Double (F64)
hilim0	Horní omezení výstupu (platné pro HLD = on)	⊙1.0	Double (F64)
lolim0	Dolní omezení výstupu (platné pro HLD = on)	⊙-1.0	Double (F64)

`HLD` Pevné saturační meze ⊙`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

$$F(s) = \frac{b_1 s + b_0}{s^2 + 2\xi\Omega s + \Omega^2}, \tag{7.4}$$

kde $\Omega > 0$ je přirozená (netlumená) frekvence, ξ, $0 < \xi < 1$, 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 ω postupně probíhající interval $\langle$wb, wf$\rangle$. Aktuální frekvence ω je přitom kopírována na výstup w. Rychlost změny (rozmítání) frekvence je dána parametrem cp, který udává relativní zmenšení počáteční periody $T_b = \frac{2\pi}{\mathtt{wb}}$ budící sinusovky za čas T_b, tedy

$$c_p = \frac{\mathtt{wb}}{\omega(T_b)} = \frac{\mathtt{wb}}{\mathtt{wb}e^{\gamma T_b}} = e^{-\gamma T_b}. \tag{7.5}$$

Hodnota parametru cp se obvykle pohybuje v intervalu cp $\in \langle 0{,}95; 1)$. 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 $\omega =$ 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,

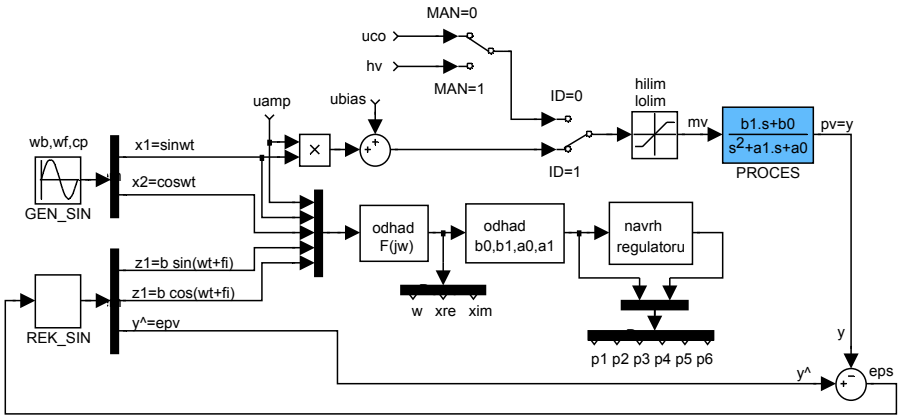
potom se v procesu identifikace dvakrát zastaví rozmítání frekvence na dobu `stime` a to v okamžicích, kdy jsou poprvé dosaženy body s fázovým zpožděním `ph1` a `ph2`. Přednastavené hodnoty parametrů `ph1` a `ph2` jsou po řadě $-60°$ a $-120°$ a mohou být změněny na libovolné hodnoty v intervalu $(-360°, 0°)$, 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.

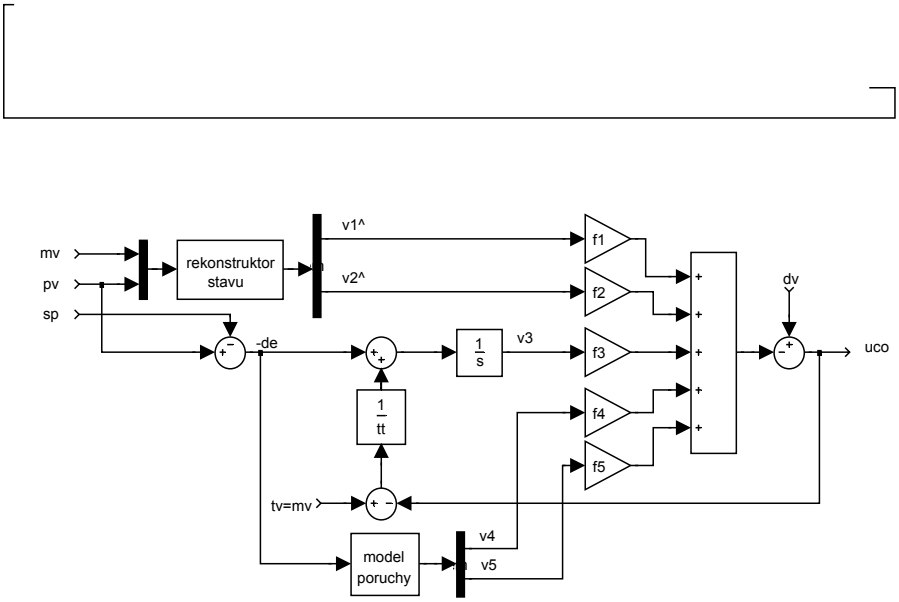
Během vlastní „identifikace a návrhu“ je výstup `IDBSY` nastaven na 1. Po skončení je shozen na 0. Při bezchybném návrhu regulátoru je výstup `IDE` = `off` a výstup `iIDE` signalizuje jednotlivé fáze identifikačního experimentu. Přibližování k prvnímu bodu je `iIDE` $= -1$, zastavení v prvním bodě `iIDE` $= 1$, 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 rekonstruktoru stavu a koeficienty `f1`, `f2`, . . . , `f5` stavové zpětné vazby.



Model řízeného systému je brán jako systém 2. řádu s přenosem ve tvaru (7.4). Jednoduchými úpravami lze dojít k přenosům

$$F(s) = \frac{(b_1 s + b_0)}{s^2 + a_1 s + a_0} \tag{7.6}$$

a

$$F(s) = \frac{K_0 \Omega^2 (\tau s + 1)}{s^2 + 2\xi\Omega s + \Omega^2}. \tag{7.7}$$

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

`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`
`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 τ `p3` ... parametr Ω v rad/s `p4` ... parametr ξ `p5` ... parametr Ω 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

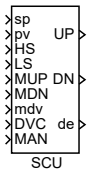
ubias	Stejnosměrná složka budicího harmonického signálu		Double (F64)
uamp	Amplituda budicího harmonického signálu	⊙1.0	Double (F64)
wb	Počátek frekvenčního intervalu [rad/s]	⊙1.0	Double (F64)
wf	Konec frekvenčního intervalu [rad/s]	⊙10.0	Double (F64)
isweep	Způsob rozmítání frekvence 1 ..... logaritmické 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)	⊙2.0	Double (F64)
xi	Koeficient tlumení rekonstruktoru (ve fázi identifikace)	⊙0.707	Double (F64)
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 ustáleni [s]	⊙10.0	Double (F64)
ralpha	Relativní poloha pólů rekonstruktoru	⊙4.0	Double (F64)
rxi	Koeficient tlumení rekonstruktoru	⊙0.707	Double (F64)
acl1	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 off ... stavový model neobsahuje integrátor on .... integrátor je zahrnut ve stavovém modelu	⊙on	Bool
apcl	Relativní poloha reálného pólu	⊙1.0	Double (F64)
DISF	Příznak rozšíření o model poruchy off ... stavový model neobsahuje model poruchy on .... model poruchy je zahrnut ve stavovém modelu		Bool
dom	Přirozená frekvence modelu poruchy	⊙1.0	Double (F64)
dxi	Koeficient tlumení modelu poruchy		Double (F64)
acl2	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



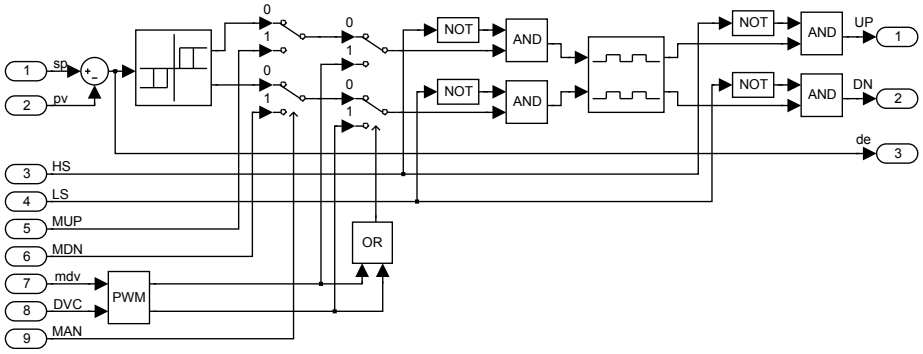
Popis funkce

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.

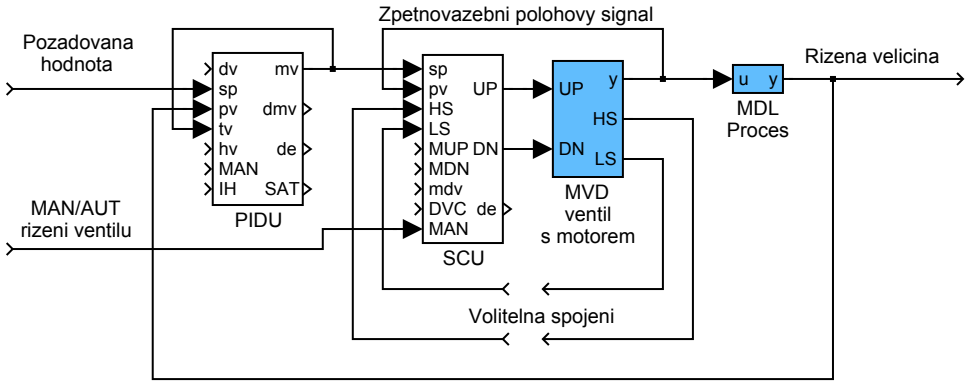
Blok SCU nejprve zpracovává regulační odchylku sp − pv na třístavový výstup symetrickým třístavovým algoritmem s parametry (práhy) thron a throff (viz blok TSE, uvažujte parametry ep = thron, epoff = throff, en = -thron a enoff = -throff). Parametr RACT určuje, pro kterou polaritu odchylky jsou generovány 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, eventuelně lze pomocí vstupu mdv nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (off→on) na vstupu DVC.

Celková funkce bloku SCU je dostatečné 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)`

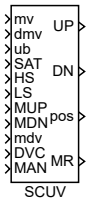
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`Bool`

`trun`	Časová konstanta motoru	↓`0.0` ⊙`10.0`	`Double (F64)`

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 (CWOI = off), potom jsou všechny tři vstupy mv, dmv a SAT bloku SCUV zpracovávány speciálním integračním algoritmem a symetrickým třístavovým algoritmem s parametry (práhy) thron a throff (viz blok TSE, uvažujte parametry ep = thron, epoff = throff, en = -thron a enoff = -throff). Vzniklé pulsy (více, méně) jsou dále upravovány 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. Parametr RACT určuje směr otáčení motoru. Poznamenejme, že nadřazený regulátor PIDU musí mít nastavení 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 (SAT = on), potom je výstup vnitřního integrátoru bloku SCUV nulován. Takto je dosaženo klidu servoventilu při dostatečně malé regulační odchylce nadřazeného regulátoru (|de| < dz – viz popis bloku PIDU).

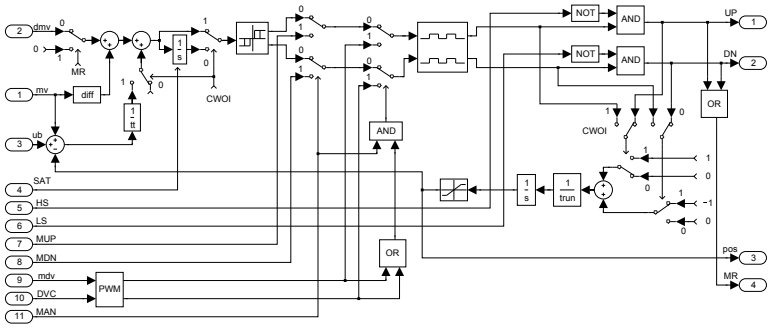
Poloha servoventilu pos je odhadována dalším vnitřním integrátorem s časovou konstantou trun. Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

Jestiže je nadřazený regulátor typu P nebo PD (CWOI = 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 thron, throff a tt musí být pečlivě nastaveny pro potlačení střídání pulsů více a méně v ustáleném stavu.

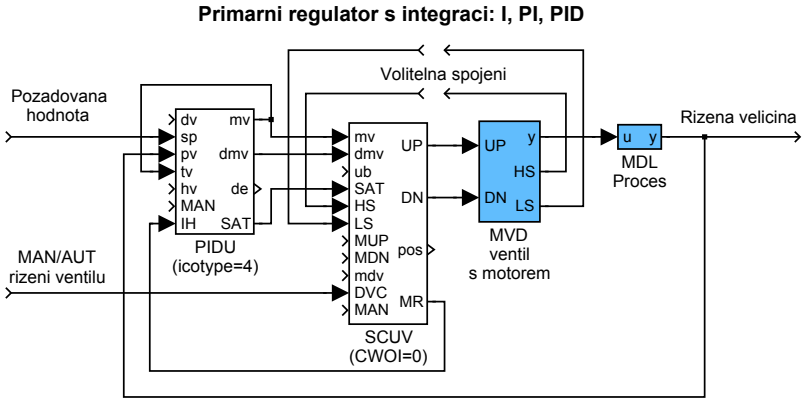
K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu MAN = on, pak lze s motorem pohybovat tam a zpět pomocí signálů MUP a MDN, eventuelně lze pomocí vstupu mdv nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (off→on) na vstupu

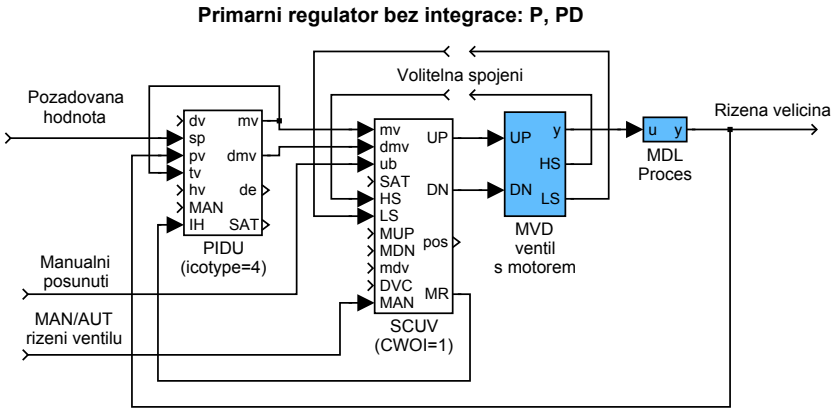
DVC.

Celková funkce bloku `SCUV` je zřejmá z následujícího diagramu:



Úplné 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 `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`
`pos`	Simulovaná poloha motoru	`Double (F64)`
`MR`	Požadavek na běh motoru `off` ... motor neběží (`UP` = `off` a `DN` = `off`) `on` .... motor se má pohybovat (`UP` = `on` nebo `DN` = `on`)	`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)`

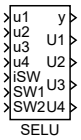
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`Bool`

`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 `off` ... primární regulátor s integrátorem (I, PI, PID) `on` .... 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 u1, u2, u3, u4 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	u1	off	on	on	on
1	off	on	u2	on	off	on	on
2	on	off	u3	on	on	off	on
3	on	on	u4	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

u1..u4	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

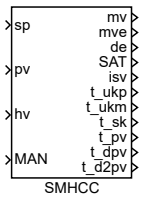
Parametr

BINF	Výběr pomocí binárních vstupů off ... zakázáno (výběr přes iSW) on .... povoleno (výběr přes SW1 a SW2)	Bool

SMHCC – Regulátor pro procesy s topením a chlazením

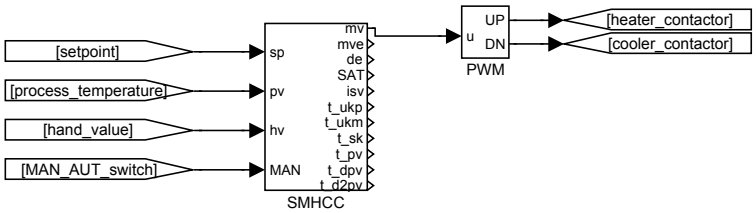
Symbol bloku

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 plastikářský 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 $T_C = i_{pwmc} T_S$ 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_{pwmc} 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 plastikářském průmyslu jsou doporučeny následující hodnoty:

$$T_S = 0.1,\ i_{pwmc} = 100,\ T_C = 10s,\ T_R = 0.01s.$$

Zákon řízení bloku regulátoru SMHCC v automatickém režimu (MAN=off) je založen na diskrétní 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

$$s_k \stackrel{\triangle}{=} \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

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 ξ a Ω 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

$$s \stackrel{\triangle}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

stále přibližně nulová. Jinak řečeno regulační odchylka e je popsána diferenciální rovnicí druhého řádu

$$s \stackrel{\triangle}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e = 0.$$

Z toho plyne, že vývoj e může být ovlivněn volbou parametrů ξ a Ω. Poznamenejme, že pro stabilní chování musí být splněno $\xi > 0$ a $\Omega > 0$. Typická optimální hodnota ξ leží v intervalu $[0.1, 8]$. Optimální hodnota Ω je silně závislá na řízeném procesu, pomalejší procesy mají menší hodnotu a rychlejší větší. Doporučená hodnota Ω pro začátek ladění parametrů je $\pi/(5T_C)$.

Řídicí veličena `mv` je obvykle v intervalu $[-1, 1]$. Kladná hodnota odpovídá topení, záporná chlazení, např. $\texttt{mv} = 1$ 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 $\texttt{hilim\_p} = 1$ and $\texttt{hilim\_m} < 1$. Pokud chceme omezení aplikovat pouze v intervalu po změně požadované hodnoty `sp`, volíme `u0_p` a `u0_m` tak, že platí $\texttt{u0\_p} \leq \texttt{hilim\_p}$ a $\texttt{u0\_m} \leq \texttt{hilim\_m}$.

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

$$if \quad (\texttt{sk} < 0.0) \quad then \quad \texttt{mv} = \texttt{t\_ukp} \quad else \quad \texttt{mv} = -\texttt{t\_ukm}\,.$$

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 $\texttt{beta} = 0.1$.

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 0 ..... automatický režim 1 ..... manuální režim	Bool

Výstupy

mv	řídicí veličina (manipulated variable)	Double (F64)
de	regulační odchylka (deviation error) de = sp − pv	Double (F64)
SAT	příznak saturace 0 ..... regulátor pracuje v lineární oblasti 1 ..... výstup regulátoru je saturován, mv ≥ hilim_p nebo mv ≤ −hilim_m	Bool
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)

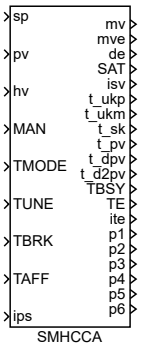
Parametry

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 ≤ 1)		Double (F64)
hilim_m	horní saturační mez amplitudy chlazení (0 < hilim_m ≤ 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	⊙10.0	Double (F64)
tauf	Časová konstanta filtru ekvivalentní akční veličiny	⊙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)`

`taup`	Časová konstanta adaptace amplitudy topení [s]	⊙700.0	Double (F64)
`taum`	Časová konstanta adaptace amplitudy chlazení [s]	⊙400.0	Double (F64)
`beta`	Šířka pásma derivačního filtru	⊙0.01	Double (F64)
`hilim_p`	Horní saturační mez amplitudy topení	↓0.0 ↑1.0 ⊙1.0	Double (F64)
`hilim_m`	Horní saturační mez amplitudy chlazení	↓0.0 ↑1.0 ⊙1.0	Double (F64)
`u0_p`	Počáteční hodnota amplitudy topení	⊙1.0	Double (F64)
`u0_m`	Počáteční hodnota amplitudy chlazení	⊙1.0	Double (F64)
`sp_dif`	Práh pro detekci změny setpointu	⊙10.0	Double (F64)
`tauf`	Časová konstanta filtru ekvivalentní akční veličiny	⊙400.0	Double (F64)
`itm`	Metoda ladění regulátoru	⊙1	Long (I32)
	`1` ..... omezeno na symetrické procesy		
	`2` ..... asymetrické procesy (zatím není implementováno)		
`ut_p`	Amplituda topení pro ladicí experiment	↓0.0 ↑1.0 ⊙1.0	Double (F64)
`ut_m`	Amplituda chlazení pro ladicí experiment	↓0.0 ↑1.0 ⊙1.0	Double (F64)

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	Double (F64)
`mve`	Ekvivalentní akční veličina	Double (F64)
`de`	Regulační odchylka	Double (F64)
`SAT`	Saturace	Bool
	`off` ... lineární zákon řízení	
	`on` .... výstup regulátoru je saturován	
`isv`	Počet kroků přepínací proměnné	Long (I32)
`t_ukp`	Aktuální amplituda topení	Double (F64)
`t_ukm`	Aktuální amplituda chlazení	Double (F64)
`t_sk`	Přepínací proměnná	Double (F64)
`t_pv`	Filtrovaná řízená veličina	Double (F64)
`t_dpv`	Filtrovaná první derivace řízené veličiny	Double (F64)
`t_d2pv`	Filtrovaná druhá derivace řízené veličiny	Double (F64)
`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)
	`0` ..... bez chyby	
	`1` ..... příliš zašuměné pv, zkontroluj teplotní vstup 2	
	`2` ..... nesprávný parametr `ut_p`	
	`3` ..... setpoint je příliš nízký	
	`4` ..... vzorkovací perioda je příliš nízká nebo je druhá derivace příliš zašuměná	
	`5` ..... předčasné ukončení ladicího experimentu	
`p1..p6`	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 off	Double (F64)
uo	Tento vstup je kopírován na výstup, jestliže alespoň jeden vstup OR1, OR2, OR3 nebo OR4 je on	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)

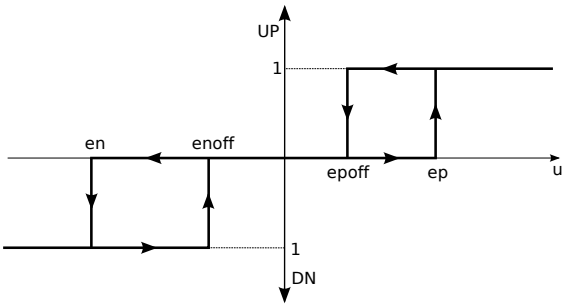
TSE – Třístavový prvek

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	⊙1.0	Double (F64)
en	Hodnota u < en způsobí nastavení výstupů UP = off a DN = off	⊙-1.0	Double (F64)
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

Symbol bloku

Licence: STANDARD

Popis funkce

Blok AND dělá logický součin dvou vstupních signálů U1 a U2.

Pokud potřebujete pracovat s více vstupními signály, použijte blok ANDOCT.

Vstupy

U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

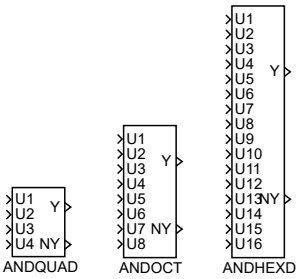
Výstupy

Y	Výstup bloku, logický součin (U1 $\wedge$ U2)	Bool
NY	Negace výstupního signálu Y (NY $= \neg$Y)	Bool

ANDQUAD, ANDOCT, ANDHEXD – Logický součin osmi 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 `nl` se před provedením logického součinu negují.

Pokud je tedy parametr `nl` prázdný, tak se provádí logický součin $\texttt{Y} = \texttt{U1} \wedge \texttt{U2} \wedge \texttt{U3} \wedge \texttt{U4} \wedge \texttt{U5} \wedge \texttt{U6} \wedge \texttt{U7} \wedge \texttt{U8}$. Pokud bude například `nl=1,3..5`, pak $\texttt{Y} = \neg\texttt{U1} \wedge \texttt{U2} \wedge \neg\texttt{U3} \wedge \neg\texttt{U4} \wedge \neg\texttt{U5} \wedge \texttt{U6} \wedge \ldots\texttt{U16}$.

Pokud pracujete s méně než 8 signály, je potřeba ošetřit nepřipojené vstupy bloku pomocí parametru `nl`. Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku AND_.

Vstupy

`U1.U16`	Logické vstupy bloku	`Bool`

Výstupy

`Y`	Výsledek logické oprace	`Bool`
`NY`	Negace výstupního signálu `Y`	`Bool`

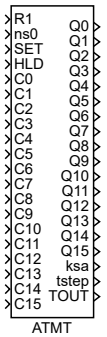
Parametr

`nl`	Seznam negovaných signálů. 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`.	`Long (I32)`

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, $i = 0, 1, \ldots, 15$, je indikován pomocí binárních výstupů `Q0`,`Q1`, ..., `Q15`. Pokud je automat ve stavu i, je nastaven příslušný výstup `Q`i = `on`. Aktuální stav automatu je též indikován celočíselným výstupem `ksa` $\in \{0, 1, \ldots, 15\}$.

Podmínky přechodů C_k, $k = 0, 1, \ldots, 15$ jsou aktivovány pomocí binárních vstupů bloku `C0`, `C1`, ..., `C15`. Pokud je `C`k = `on`, je podmínka Ck splněna, naopak pro `C`k = `off` splněna není.

Funkce automatu se zadává pomocí tabulky stavů a přechodů:

$$\begin{array}{lll} S1 & C1 & NS1 \\ S2 & C2 & NS2 \\ & & \ldots \\ Sn & Cn & NSn \end{array}$$

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. prvý 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` = `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 setrvá 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 morestps 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 zkonstruovat 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.

Vstupy

R1	Resetovací signál, je-li R1 = on, je automat převeden do počátečního stavu S0 (vstup R1 má prioritu před vstupem SET)	Bool
ns0	Do tohoto stavu přejde automat při náběžné hraně na vstupu SET	Long (I32)
SET	Náběžná hrana na vstupu SET způsobí přechod z aktuálního stavu do stavu ns0	Bool
HLD	Blokovací vstup, HLD = on zablokuje činnost automatu, stav zůstává, výstup tstep se neinkrementuje	Bool
C0...C15	Podmínky přechodu, Ci = on značí, že i-tá podmínka je splněna	Bool

Výstupy

Q0...Q15	Výstupní signály určující stav automatu, aktivní je ten stav i, pro který platí Qi = on	Bool
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

Parametry

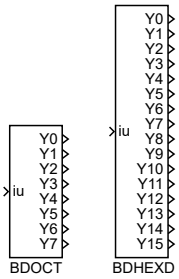
morestps	Povolit více přechodů mezi stavy automatu v jednom cyklu off ... zakázáno on .... povoleno	Bool
sfcname	Jméno souboru, kam si konfigurátor bloku ukládá data (pokud se nevyplní, zvolí se automaticky podle jména bloku)	String

`STT`	Tabulka přechodů mezi stavy ⊙`[0 0 1; 1 1 2; 2 2 3; 3 3 0]`	`Byte (U8)`
`touts`	Vektor časových limitů $TO0 \ldots TO15$ pro jednotlivé stavy $S0 \ldots S15$) ⊙`[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]`	`Double (F64)`

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 `Y0` je vždy nejnižší bit tohoto čísla.

Vstup

`iu`	Vstupní signál k dekompozici	`Long (I32)`

Výstupy

`Y0...Y15`	Jednotlivé bity vstupního signálu	`Bool`

Parametr

`shift`	Počet bitů, o který se posune vstupní signál `iu` před rozebráním na jednotlivé bitové výstupy ↓0 ↑31	`Long (I32)`
`vtype`	Typ hodnoty vstupu, může nabývat hodnot: ⊙4 `2` ..... Byte (rozsah 0 ... 255) `3` ..... Short (rozsah -32768 ... 32767) `4` ..... Long (rozsah -2147483648 ... 2147483647) `5` ..... Word (rozsah 0 ... 65536) `6` ..... DWord (rozsah 0 ... 4294967295) `10` .... Large (rozsah -9223372036854775808...9223372036854775807)	`Long (I32)`

BITOP – Bitová operace dvou celočíselných signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok BITOP provádí operaci i1 ∘ 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).

Vstupy

i1	První celočíselný vstup bloku	Long (I32)
i2	Druhý celočíselný vstup bloku	Long (I32)

Výstup

n	Výsledek bitové operace určené parametrem iop	Long (I32)

Parametr

iop	Bitová operace 1 ..... bitová negace (Bit NOT) 2 ..... logický součet po jednotlivých bitech (Bit OR) 3 ..... logický součin po jednotlivých bitech (Bit AND) 4 ..... logický exkluzivní součet po jednotlivých bitech (Bit XOR) 5 ..... posun signálu i1 doleva o i2 bitů (Shift Left) 6 ..... posun signálu i1 doprava o i2 bitů (Shift Right) 7 ..... dvojkový doplněk signálu i1 na 8 bitech (2's Complement - Byte) 8 ..... dvojkový doplněk signálu i1 na 16 bitech (2's Complement - Word) 9 ..... dvojkový doplněk signálu i1 na 32 bitech (2's Complement - Long)	⊙1	Long (I32)

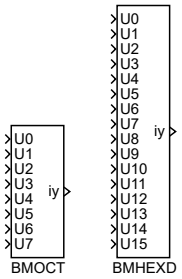
`vtype` Typ hodnoty vstupů a výstupu, může nabývat hodnot: ⊙`4` `Long (I32)`

- `2` ..... Byte (rozsah 0 ... 255)
- `3` ..... Short (rozsah -32768 ... 32767)
- `4` ..... Long (rozsah -2147483648 ... 2147483647)
- `5` ..... Word (rozsah 0 ... 65536)
- `6` ..... DWord (rozsah 0 ... 4294967295)
- `10` .... Large (rozsah -9223372036854775808...9223372036854775807)

BMOCT, BMHEXD – Bitový multiplexer

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, v nejnižším bitu je vždy signál `U0`.

Vstupy

`U0...U15`	Jednotlivé bity výstupního signálu	`Bool`

Výstup

`iy`	Výsledný výstupní signál	`Long (I32)`

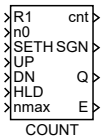
Parametr

`shift`	Počet bitů, o který se posune složená celočíselná hodnota z jednotlivých vstupů těsně před předáním na výstup `iy` ↓0 ↑31	`Long (I32)`
`vtype`	Typ hodnoty výstupu, může nabývat hodnot: ⊙4 `2` ..... Byte (rozsah 0 ... 255) `3` ..... Short (rozsah -32768 ... 32767) `4` ..... Long (rozsah -2147483648 ... 2147483647) `5` ..... Word (rozsah 0 ... 65536) `6` ..... DWord (rozsah 0 ... 4294967295) `10` .... Large (rozsah -9223372036854775808...9223372036854775807)	`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 cnt $\geq$ nmax způsobí nastavení výstupu Q na hodnotu on.

Vstupy

R1	Reset bloku (R1 = on)	Bool
n0	Hodnota pro nastavení čítače pomocí vstupu SETH	Long (I32)
SETH	Nastavení hodnoty čítače na n0 (SETH = on)	Bool
UP	Vstup pro přičítání	Bool
DN	Vstup pro odečítání	Bool
HLD	Zmrazení čítače off ... čítač běží on .... čítač je zablokován	Bool
nmax	Cílová hodnota čítače	Long (I32)

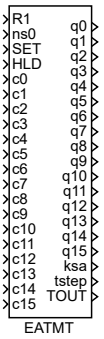
Výstupy

cnt	Celkový počet načtených pulzů	Long (I32)
SGN	Znaménko výstupu cnt off ... pro cnt $<$ 0 on .... pro cnt $\geq$ 0	Bool
Q	Indikátor dosažení cílové hodnoty off ... pro cnt $<$ nmax on .... pro cnt $\geq$ nmax	Bool
E	Indikátor současného výskytu náběžných hran na vstupech UP a DN	Bool

EATMT – Extended finite-state automaton

Symbol bloku

Licence: ADVANCED



Popis funkce

The EATMT block implements a finite automat with at most 256 states and 256 transition rules, thus it extends the possibilities of the ATMT block.

The current state of the automat i, $i = 0, 1, \ldots, 255$ is indicated by individual bits of the integer outputs q0, q1, ..., q15. Only a single bit with index $i \,\mathrm{MOD}\, 16$ of the q($i \,\mathrm{DIV}\, 16$) output is set to 1. The remaining bits of that output and the other outputs are zero. The bits are numbered from zero, least significant bit first. Note that the DIV and MOD operators denote integer division and remainder after integer division respectively. The current state is also indicated by the ksa $\in \{0, 1, \ldots, 255\}$ output.

The transition conditions C_k, $k = 0, 1, \ldots, 255$) are activated by individual bits of the inputs c0, c1, ..., c15. The k-th transition condition is fulfilled when the ($k \,\mathrm{MOD}\, 16$)-th bit of the input c($k \,\mathrm{DIV}\, 16$) is equal to 1. The transition cannot happen otherwise.

The BMHEXD or BMOCT bitwise multiplexers can be used for composition of the input signals c0, c1, ..., c15 from individual Boolean signals. Similarly the output signals q0, q1, ..., q15 can be decomposed using the BDHEXD or BDOCT bitwise demultiplexers.

The automat function is defined by the following table of transitions:

$$\begin{array}{ccc} S1 & C1 & FS1 \\ S2 & C2 & FS2 \\ & & \ldots \\ Sn & Cn & FSn \end{array}$$

Each row of this table represents one transition rule. For example the first row

$$S1 \quad C1 \quad FS1$$

has the meaning

If ($S1$ is the current state AND transition condition $C1$ is fulfilled)
then proceed to the following state $FS1$.

The above described meaning of the table row holds for $C1 < 1000$. Negation of the $(C1 - 1000)$-th transition condition is assumed for $C1 \geq 1000$.

The above mentioned table can be easily constructed from the automat state diagram or SFC description (Sequential Function Charts, formerly Grafcet).

The `R1 = on` input resets the automat to the initial state $S0$. The `SET` input allows manual transition from the current state to the `ns0` state when rising edge occurs. The `R1` input overpowers the `SET` input. The `HLD = on` input freezes the automat activity, the automat stays in the current state regardless of the `c`i input signals and the `tstep` timer is not incremented. The `TOUT` output indicates that the machine remains in the given state longer than expected. The time limits TOi for individual states are defined by the `touts` array. There is no time limit for the given state when TOi is set to zero. The `TOUT` output is set to `off` whenever the automat changes its state.

It is possible to allow more state transitions in one cycle by the `morestps` parameter. However, this option must be thoroughly considered and tested, namely when the `TOUT` output is used in transition conditions. In such a case it is strongly recommended to incorporate the `ksa` output in the transition conditions as well.

The development tools of REXYGEN include also the SFCEditor program. You can create SFC schemes graphically using this tool. Run this editor from REXYGEN Studio by clicking the *Configure* button in the parameter dialog of the `EATMT` block.

Vstupy

`R1`	Reset signal, `R1 = on` brings the automat to the initial state `S0`; the `R1` input overpowers the `SET` input	`Bool`
`ns0`	This state is reached when rising edge occurs at the the `SET` input	`Long (I32)`
`SET`	The rising edge of this signal forces the transition to the `ns0` state	`Bool`
`HLD`	The `HLD = on` freezes the automat, no transitions occur regardless of the input signals, `tstep` is not increasing	`Bool`
`c0...c15`	Transition conditions, each input signal contains 16 transition conditions, see details above	

Výstupy

`q0...q15`	Output signals indicating the current state of the automat, see details above	`Long (I32)`
`ksa`	Integer code of the active state	`Long (I32)`
`tstep`	Time elapsed since the current state was reached; the timer is set to `0` whenever a state transition occurs	`Double (F64)`
`TOUT`	Flag indicating that the time limit for the current state was exceeded	`Bool`

Parametry

`morestps`	Allow multiple transitions in one cycle of the automat `off` ... Disabled `on` .... Enabled	`Bool`
`sfcname`	Jméno souboru, kam si konfigurátor bloku ukládá data (pokud se nevyplní, zvolí se automaticky podle jména bloku)	`String`
`STT`	State transition table (matrix) ⊙`[0 0 1; 1 1 2; 2 2 3; 3 3 0]`	`Short (I16)`
`touts`	Vector of timeouts `TO0`, `TO1`, ..., `TO255` for the states $S0$, $S1$, ..., $S255$ ⊙`[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]`	`Double (F64)`

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).

Vstup

`U`	Logický vstupní signál	`Bool`

Výstup

`Y`	Indikace výskytu zvolené hrany	`Bool`

Parametr

`iedge`	Typ detekovaných hran `1` ..... náběžná hrana `2` ..... sestupná hrana `3` ..... obě hrany	⊙`1`	`Long (I32)`

EQ – Shodnost dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok porovnává dva vstupní signály. Výstup `Y = 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.

Vstupy

`u1`	Vstup bloku	`Any`
`u2`	Vstup bloku	`Any`

Výstup

`Y`	Výstup bloku	`Bool`
`Y`	Negace výstupu bloku	`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.

Vstup

i	Celočíselný signál pro zpracování ↓-9.22337E+18 ↑9.22337E+18	Large (I64)

Parametry

shift	Bitový posun (záporné číslo=doleva, kladné číslo=doprava) ↓-63 ↑63	Long (I32)
mask	Bitová maska (aplikovaná po bitovém posunu) ↓0 ↑4294970000 ⊙4294967295	Large (I64)
vtype	Typ hodnoty výstupu, může nabývat hodnot: ⊙4 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	Long (I32)

Výstup

n	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.

Vstupy

i1	První celočíselný vstup bloku	Long (I32)
i2	Druhý celočíselný vstup bloku	Long (I32)
SW	Přepínací signál off ... je zvolen vstupní signál i1 on .... je zvolen vstupní signál i2	Bool

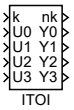
Výstup

n	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 $(\texttt{U3 U2 U1 U0})_2$ z množiny $\{0, 1, 2, \ldots, 15\}$ výstupní číslo nk a jeho binární reprezentaci $(\texttt{Y3 Y2 Y1 Y0})_2$ z téže množiny. Příslušné zobrazení je popsáno tabulkou

k	0	1	2	...	15
nk	n0	n1	n2	...	n15

kde n0, ..., n15 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 $(\texttt{U3 U2 U1 U0})_2$.

Vstupy

k	Celočíselný vstupní signál	Long (I32)
U0	Binární číslice vstupu s vahou 1	Bool
U1	Binární číslice vstupu s vahou 2	Bool
U2	Binární číslice vstupu s vahou 4	Bool
U3	Binární číslice vstupu s vahou 8	Bool

Výstupy

nk	Celočíselný výstupní signál	Long (I32)
Y0	Binární číslice výstupu s vahou 1	Bool
Y1	Binární číslice výstupu s vahou 2	Bool
Y2	Binární číslice výstupu s vahou 4	Bool
Y3	Binární číslice výstupu s vahou 8	Bool

Parametry

BINF	Transformace hodnoty z binárních vstupů off ... zakázáno (transformace vstupu k) on .... povoleno (vstupem je hodnota $(\texttt{U3 U2 U1 U0})_2$)	⊙on	Bool
fktab	Cílové hodnoty převodní tabulky	⊙[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]	Byte (U8)

NOT – Logická negace

Symbol bloku

Licence: STANDARD

Popis funkce

Blok NOT dělá logickou negaci vstupního signálu.

Vstup

U	Logický vstupní signál	Bool

Výstup

Y	Logický výstupní signál (Y = ¬U)	Bool

OR – Logický součet dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok OR dělá logický součet dvou vstupních signálů U1 a U2.

Pokud potřebujete pracovat s více vstupními signály, použijte blok OROCT.

Vstupy

U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

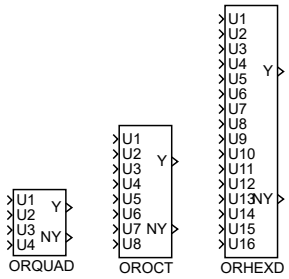
Výstupy

Y	Logický výstupní signál (U1 ∨ U2)	Bool
NY	Negace výstupního signálu Y (NY = ¬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 `nl` se před provedením logického součinu negují.

Pokud je tedy parametr `nl` prázdný, tak se provádí logický součet $\texttt{Y} = \texttt{U1} \vee \texttt{U2} \vee \texttt{U3} \vee \texttt{U4} \vee \texttt{U5} \vee \ldots \vee \texttt{U16}$. Pokud bude například `nl=1,3..5`, pak $\texttt{Y} = \neg\texttt{U1} \vee \texttt{U2} \vee \neg\texttt{U3} \vee \neg\texttt{U4} \vee \neg\texttt{U5} \vee \texttt{U6} \vee \ldots \vee \texttt{U16}$.

Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku OR_.

Vstupy

`U1..U16`	Logické vstupy bloku	`Bool`

Parametr

`nl`	Seznam negovaných signálů. 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`.	`Long (I32)`

Výstupy

`Y`	Výsledek logické oprace	`Bool`
`NY`	Negace výstupního signálu `Y`	`Bool`

RS – Klopný obvod

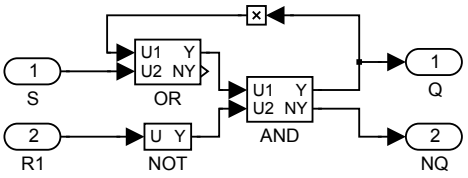
Symbol bloku

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.



Vstupy

S	Nahození klopného obvodu, nastaví výstup Q na on	Bool
R1	Přednostní shození klopného obvodu, nastaví výstup Q na off	Bool

Výstupy

Q	Stav klopného obvodu	Bool
NQ	Negace výstupního signálu Q	Bool

SR – Klopný obvod

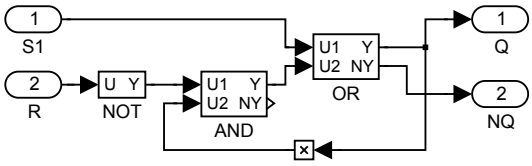
Symbol bloku

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.



Vstupy

`S1`	Přednostní nahození klopného obvodu, nastaví výstup `Q` na `on`, má přednost před vstupem `R`	`Bool`
`R`	Shození klopného obvodu, nastaví výstup `Q` na `off`	`Bool`

Výstupy

`Q`	Stav klopného obvodu	`Bool`
`NQ`	Negace výstupního signálu `Q`	`Bool`

TIMER – Vícefunkční časovač

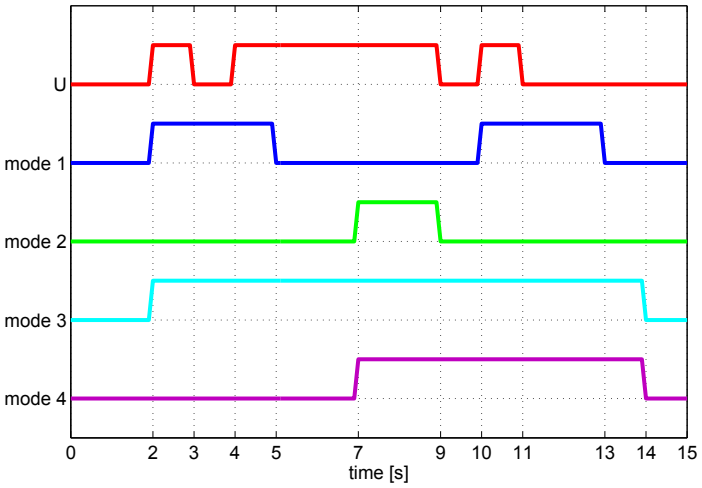
Symbol bloku

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.

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.

Vstupy

U	Signál spouštějící časovač	Bool
HLD	Pozastavení časovače	Bool
R1	Reset bloku (R1 = on)	Bool

Výstupy

Q	Výstupní signál časovače	Bool
et	Doba uplynulá od startu časovače [s]	Double (F64)
rt	Zbývající doba [s]	Double (F64)

Parametry

`mode`	Režim činnosti časovače `1` ..... generovaný pulz – 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` ..... zpožděné zapnutí – signál ze vstupu `U` je kopírován na výstup `Q` tak, že začátek impulzu na výstupu je opožděn o `pt` sekund proti začátku pulzu na vstupu; pulzy kratší než `pt` sekund se na výstupu neobjeví `3` ..... zpožděné vypnutí – signál ze vstupu `U` je kopírován na výstup `Q` tak, že konec impulzu 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` ..... zpožděná změna – výstupní signál `Q` se přepne na hodnotu vstupu `U` až tehdy, když je vstup po dobu `pt` sekund neměnný	⊙`1`	`Long (I32)`
`pt`	Doba časování [s]	⊙`1.0`	`Double (F64)`

Kapitola 9

TIME – Bloky pro práci s časem

Obsah

Datum a čas je v systému REXYGEN reprezentován časovou značkou, což je 64-bitové číslo. Momentálně jsou to nanosekundy od 1.ledna 2000 0:00:00.0 UTC, přičemž přestupné sekundy (leap second) se neuvažují. Doporučuje se o časové značce nic nepředpokládat, a pro práci s ní používat pouze bloky z této podskupiny, které vědí, jak ji správně interpretovat.

Většina bloků pro převod různých formátů datumu a času má parametr `tz` (timezone, tj. časové pásmo). Pokud je tento parametr prázdný nebo má hodnotu `localtime`, použije se časové pásmo nastavené v operačním systému a to včetně přepínání letního a zimního času (pokud to je v operačním systému nastaveno). Dále je možné nastavit pevné posunutí od UTC (formát hhmm, např. +0200 nebo -0800). Další časová pásma se nastavují podle toho, jak to vyžaduje operační systém (počítače, kde běží exekutiva).

Ve Windows je formát `<třípísmenná zkratka><ofset v hodinách>[<třípísmenná zkratka letního času>]`, tj. například PST8PDT . Přičemž přepínání mezi zimním a letním časem je podle pravidel v USA (i když dokumentace je nejednoznačná - podle některých údajů mají Windows časová pásma podle IANA tabulek).

V linuxu je každé časové pásmo (včetně definice přepínání na letní a zimní čas i s respektováním jiných pravidel v různých letech) definováno souborem v adresáři `/usr/share/zoneinfo` nebo jeho podaresářích. Do parametru `tz` se pak píše dvojtečka a jméno tohoto souboru, tj. například `:CET`, `:UTC`, `:Europe/Paris` (dvojtečka na začátku není nutná, ale může se to pochopit jako zadání 2. způsobem a pak to zdánlivě nefunguje). Další možnost je zadat přímo definiční text (tj. obsah výše uvedeného souboru), např. `NZST-12:00:00NZDT-13:00:00,M10.1.0,M3.3.0` (podrobný popis např. `man timezone`).

Známé problémy:

- Blok `DT2STR` a `STR2DT` pro formát používá funkci `strftime()`, která neumí pracovat s milisekundami (ani nemůže, protože to není ve struktuře `struct tm`, kterou používá jako parametr). Pokud je to potřeba, je nutné nechat parametr `format` prázdný. Textová reprezentace se pak formátuje jinou funkcí ve formátu ISO8601, která s milisekundami pracovat umí.
- Blok `DT2STR` a `STR2DT` nemá parametr `locale`. Pokud jsou ve stringu dny v týdnu nebo měsíce jako text, tak se použije aktuální `locale` z operačního systému (resp. locale nastavené v environment proměnných RexCore).
- Blok `STR2DT` na Windows neumožňuje zadat formátovací string
- Blok `STR2DT` na Windows neumožňuje zadat formátovací string (parameter `format` musí zůstat prázdný a načítá se datum a čas v ISO8601 formátu)
- Pokud se u bloku `STR2DT` zadá formátovací string (parameter `format`) nelze načíst časové pásmo ze vstupního stringu
- Pokud se do parametru `tz` zadá časové pásmo, které operační systém nezná, blok nehlásí žádnou chybu, ale použité časové pásmo je nedefinované (na Windows mi to dělalo UTC).
- !!!!!!! mktime() i timelocal() na linuxu nerespektuje nastavení TZ v environment

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

year	Rok	Long (I32)
month	Měsíc	Long (I32)
day	Den	Long (I32)
dow	Den v týdnu, první den je neděle (1)	Long (I32)

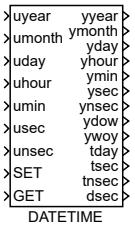
Parametr

tz	Časové pásmo 1 ..... lokální čas 2 ..... UTC	⊙1	Long (I32)

DATETIME – Čtení, nastavování a konverze času

Symbol bloku

Licence: STANDARD



Popis funkce

Blok DATETIME 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

uyear	Vstup pro nastavení roku	Long (I32)
umonth	Vstup pro nastavení měsíce	Long (I32)
uday	Vstup pro nastavení dne	Long (I32)
uhour	Vstup pro nastavení hodin	Long (I32)
umin	Vstup pro nastavení minut	Long (I32)

usec	Vstup pro nastavení sekund		Long (I32)
unsec	Vstup pro nastavení nanosekund	↓-9.22E+18 ↑9.22E+18	Large (I64)
SET	Nastavení času pomocí náběžné hrany		Bool
GET	Přečtení času pomocí náběžné hrany		Bool

Výstupy

yyear	Rok	Long (I32)
ymonth	Měsíc	Long (I32)
yday	Den	Long (I32)
yhour	Hodiny	Long (I32)
ymin	Minuty	Long (I32)
ysec	Sekundy	Long (I32)
ynsec	Nanosekundy	Long (I32)
ydow	Den v týdnu	Long (I32)
ywoy	Týden v roce	Long (I32)
tday	Počet dní od začátku epochy	Long (I32)
tsec	Počet sekund od začátku epochy	Long (I32)
tnsec	Počet nanosekund od začátku epochy	Large (I64)
dsec	Počet sekund od půlnoci	Long (I32)

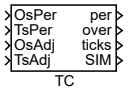
Parametry

isetmode	Zdroj podle kterého nastavit čas 1 ..... čas OS 2 ..... vstupy bloku 3 ..... vstup unsec 4 ..... vstup usec 5 ..... vstup unsec relativně	⊙1	Long (I32)
igetmode	Zdroj ze kterého přečíst čas pro nastavení či konverzi 1 ..... čas OS 2 ..... vstupy bloku 3 ..... vstup unsec 4 ..... vstup usec 5 ..... vstup uday 6 ..... čas systému REXYGEN	⊙6	Long (I32)
settol	Tolerance pro nastavení času systému REXYGEN [s]	⊙1.0	Double (F64)
setper	Perioda nastavování času [s] (0=bez opakování)		Double (F64)
getper	Perioda čtení času [s] (0=bez opakování)	⊙0.001	Double (F64)
FDOW	První den v týdnu je neděle off ... týden začíná pondělím on .... týden začíná nedělí		Bool
tz	Časové pásmo 1 ..... lokální čas 2 ..... UTC	⊙1	Long (I32)

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 parameterem tick bloku EXEC) a to jak skutečnou délku tiku, tak i logickou délku (kolik se kund se přičte do časovevé značky při každém tiku). 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 příliž narušeno časování, jsou větší posuny realizovány tak, že je dočasně snížena nebo zvýšena perioda tiku, dokud nedojde k požadovanému posunu. Jak se změní perioda ovlivňuje parametr OsMax.

Příklad: předpokládejme periodu tiku 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

OsPer	Skutečná perioda tiku [s]	Double (F64)
"TsPer	Perioda tiku pro časovou značku[s]	Double (F64)
OsAdj	Posun skutečné polohy tiku [s]	Double (F64)

`TsAdj`	Posun časové značky tiku [s]	`Double (F64)`

Parametry

`OsMax`	Maximální relativní změna tiku pro vstup OsAdj ↓`0.0` ↑`1.0` ⊙`0.1`	`Double (F64)`
`TsMax`	Maximální relativní změna tiku pro vstup TsAdj ↓`0.0` ↑`1.0` ⊙`0.1`	`Double (F64)`

Výstupy

`per`	Skutečná délka posledního tiku [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)

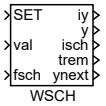
Parametr

tz	Časové pásmo 1 ..... lokální čas 2 ..... UTC	⊙1	Long (I32)

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 `y` a `iy` pomocí náběžné hrany	`Bool`
`val`	Hodnota pro dočasné nastavení výstupů `y` a `iy`	`Double (F64)`
`fsch`	Vynucený denní program `0` ..... provoz dle týdenního programu `1` ..... pondělí `2` ..... úterý .... `7` ..... neděle `8 a více` další denní programy dle tabulky `wst`	`Long (I32)`

Výstupy

`iy`	Celočíselná výstupní hodnota	`Long (I32)`
`y`	Výstupní hodnota	`Double (F64)`
`isch`	Identifikace denního programu	`Long (I32)`
`trem`	Čas zbývající v aktuálním intervalu [s]	`Double (F64)`
`ynext`	Výstupní hodnota v dalším intervalu	`Double (F64)`

Parametry

`tz`	Časové pásmo ⊙`1` `1` ..... lokální čas `2` ..... UTC	`Long (I32)`
`nmax`	Velikost alokovaných polí ↓`10` ↑`1000000` ⊙`100`	`Long (I32)`
`wst`	Tabulka týdenního programu (seznam trojic *den-hodina-hodnota*) ⊙`[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`	`Double (F64)`
`specdays`	Seznam speciálních dní (seznam dvojic *datum-denní program*) ⊙`[20150406 1; 20151224 1; 20151225 1; 20151226 1; 20160101 1; 20160328 1; 20170`	`Long (I32)`

Kapitola 10

ARC – Archivace dat

Obsah

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 řídícího 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 zachovávání 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.

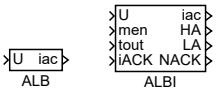
Daná hardwarová platforma nemusí podporovat všechny druhy archivů. Příznaky podporovaných druhů archivů jsou součástí verze řídicího systému cílového zařízení a lze je zjistit v diagnostickém panelu v REXYGEN Studio po kliknutí na jméno cílového zařízení (IP adresu) ve stromu exekutivy. Nachází se na kartě `Target` v levé spodní části.

10.2 Generování alarmů u a událostí

ALB, ALBI – Alarmy pro logickou hodnotu

Symboly 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 lvl. Pokud je $1 \leq$ lvl ≤ 127, jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah $128 \leq$ lvl ≤ 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`

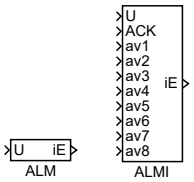
Parametry

`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 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 $\texttt{id} = 0$. ⊙`1`	`Word (U16)`
`lvl`	Úroveň (závažnost) alarmu, určující, zda jde o skutečný alarm či jen o událost. ↓`1` ⊙`1`	`Byte (U8)`
`Desc`	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REXYGEN. ⊙`Alarm Description`	`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=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=on.

Poznámka: Systém zobrazuje stav potvrzení, umožňuje potvrdit i trvající 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 vizualziaci a na návrhu celého systému.

Vstupy

U	Stav alarmu. Alarm je aktivní, pokud U = on	Bool

Výstupy

iE	Kód chyby	Error

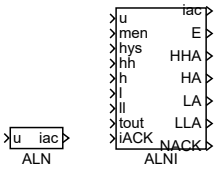
Parametry

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) ↓-1 ↑65535 ⊙-1	Long (I32)
ACK	Potvrzení alarmu nastavením ACK = on. Blok po zpracování potvrzení sám nastaví ACK = off.	Bool
av1..av8	Hodnota spojená s alarmem. Podrobný popis, jak se hodnota projeví v alarmu (jeho popisu) je v bloku ALARMS. ↓1.79769e+308 ⊙-1.79769e+308	Double (F64)

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 `lvl`. Pokud je $1 \leq$ `lvl` ≤ 127, jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah $128 \leq$ `lvl` ≤ 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

`u`	Analogový vstupní signál, podle jehož hodnoty se generují alarmy		`Double (F64)`
`hys`	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)`
`hh`	Mez pro druhý horní alarm. Musí být větší než mez `h`.		`Double (F64)`
`h`	Mez pro horní alarm. Musí být větší než mez `l`.		`Double (F64)`
`l`	Mez pro dolní alarm. Musí být větší než mez `ll`.		`Double (F64)`
`ll`	Mez pro druhý dolní alarm		`Double (F64)`
`tout`	Doba zpoždění aktivace alarmu [s]	↓`0.0`	`Double (F64)`

iACK	Potvrzení alarmů 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.	Byte (U8)

V případě, že stačí daným blokem generovat jen jednoúrovňové alarmy, stačí nastavit lvl2=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

iac	Kód aktuálního stavu alarmového bloku 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.	Long (I32)
E	Příznak chyby uspořádání alarmových mezí off ... bez chyby on .... nastala chyba	Bool
HHA	Indikátor druhého horního alarmu	Bool
HA	Indikátor (prvního) horního alarmu	Bool
LA	Indikátor (prvního) dolního alarmu	Bool
LLA	Indikátor druhého dolního alarmu	Bool
NACK	Indikátor nepotvrzení alarmu	Bool

Parametry

acls	Třída alarmu (typ proměnné, která bude do archivu ukládána) ⊙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)	Byte (U8)

`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. ⊙`1`	`Word (U16)`
`lvl1`	Úroveň (závažnost) prvních horních a dolních alarmů (`HA` a `LA`), určující, zda jde o skutečný alarm či jen o událost ↓`1` ⊙`1`	`Byte (U8)`
`lvl2`	Úroveň (závažnost) druhých horních a dolních alarmů (`HHA` a `LLA`) ↓`1` ⊙`10`	`Byte (U8)`
`Desc`	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REXYGEN. ⊙`Alarm Description`	`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 tiku tasku 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 zapíše 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 polžky, generuje je téměř jistě nějaký blok `ARS` (nebo analogická funkce ve skriptu bloku REXLANG).

Vstupy

`u`	Signál pro uložení do archivu	`Any`
`RUN`	Povolení běhu algoritmu	`Bool`

Parametry

`type`	Typ všech použitých bufferů `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	⊙`12`	`Byte (U8)`
`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

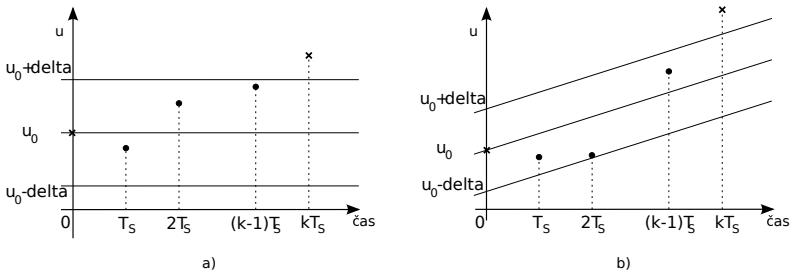
Blok ACD (Archive Compression using Delta criterion) je určen pro ukládání komprimovaných analogových signálů do archivu pomocí archivních událostí.

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 ⟨tmin,tmax⟩ sekund (doby jsou zaokrouhleny na nejbližší násobek periody vzorkování). Pokud se hodnota signálu „hodně“ mění, ukládá se signál jednou za čas tmin, pokud se hodnota signálu mění „málo“ nebo je konstantní, ukládá se signál jednou za čas tmax. Po spuštění bloku se vždy uloží první hodnota vstupu u, označme ji u0. Přesná pravidla ukládání dalších vzorků jsou určena vstupem delta a parametrem TR.

Je-li TR=off, testuje se podmínka $|u - u0| > delta$. Pokud je splněna a od minulého uložení uplynul alespoň čas tmin uloží se tato hodnota u do archivu a nastaví se u0=u. Je-li podmínka splněna dříve než za čas tmin od posledního uložení nastaví se chybový výstup E na 1 a počká se s uložením na první vzorek po uplynutí času tmin, v tomto okamžiku se nastavuje E=0. Pak se celý postup opakuje od začátku.

Je-li TR=on, pracuje blok tak, že ukládá první vzorek, který se odchyluje o více než toleranci 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 TR=off, b) pro TR=on. Ukládané vzorky jsou označeny symbolem ×.



Vstupy

u	Komprimovaně ukládaný signál		Double (F64)
delta	Práh pro ukládání signálu do archivu	↓0.0 ↑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`

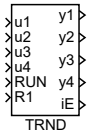
Parametry

`acls`	Třída alarmu, určující typ proměnné, která bude do archivu ukládána ⊙8 `1` ..... Bool `5` ..... Word (U16) .... `2` ..... Byte (U8) `6` ..... DWord (U32) ... Large (I64) `3` ..... Short (I16) `7` ..... Float (F32) `4` ..... Long (I32) `8` ..... Double (F64)	`Byte (U8)`
`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 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. ⊙on `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 `ptype1` až `ptype4`. 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 $s \cdot \mathtt{n} \cdot \mathtt{l}$, kde s je velikost proměnné daného typu v bytech. Přednastavený typ `Double` zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů $\mathtt{n} = 4$, délka každého trendu $\mathtt{l} = 1000$, 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 18.

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 nezapisují do archivu a jsou dostupná jen v diagnostických nástrojích.

Vstupy

`u1..u4`	Analogové vstupy bloku určené pro zpracování a ukládání do trendu	`Double (F64)`
`RUN`	Povolení běhu algoritmu. Data se zpracovávají a ukládají jen pokud je `RUN` = `on`.	`Bool`
`R1`	Signál pro vymazání obsahu trendového bloku. Data jsou mazána při každém spuštění bloku, je-li `R1` = `on`. Vstup má přednost před vstupem `RUN`.	`Bool`

Výstupy

`y1..y4`	Analogové výstupy bloku nastavované jednou za `pfac` spuštění bloku na poslední hodnoty uložené do trendových bufferů	`Double (F64)`
`iE`	Kód chyby ukládání do trendových bufferů, viz popis v textu výše	`Long (I32)`

Parametry

`n`	Počet signálů (bufferů) v trendu ↓1 ↑4 ⊙4	`Long (I32)`
`l`	Počet vzorků vyhrazený v paměti pro každý buffer trendu ↓0 ↑268435000 ⊙1000	`Long (I32)`
`btype`	Typ všech použitých bufferů trendu ⊙8 `1` ..... Bool `4` ..... Long `7` ..... Float `2` ..... Byte `5` ..... Word `8` ..... Double `3` ..... Short `6` ..... DWord `10` .... Large	`Long (I32)`
`ptype`i	Způsob zpracování signálu `u`i, $i = 1 \ldots 4$. Zvolený způsob se aplikuje na posledních `pfac` vzorků a výsledek se uloží do i-tého trendového bufferu ⊙1 `1` ..... ukládání bez zpracování `2` ..... minimum z `pfac` vzorků `3` ..... maximum z `pfac` vzorků `4` ..... součet `pfac` vzorků `5` ..... aritmetický průměr z `pfac` vzorků `6` ..... směrodatná odchylka `pfac` vzorků `7` ..... rozptyl `pfac` vzorků	`Long (I32)`

`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`
`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 `RTC`. Ostatní volby jsou určeny pouze pro ladicí nebo speciální účely. ⊙1 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 – pří spuštění bloku 4 ..... `PFC` – hrubý čas 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 l 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 $s \cdot \mathtt{n} \cdot \mathtt{l}$, kde s je velikost proměnné daného typu v bytech. Přednastavený typ Double zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů $\mathtt{n} = 4$, délka každého trendu $\mathtt{l} = 1000$, 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 18.

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

uVec	Vektorový signál určený k uložení	Reference
HLD	Pozastavení ukládání dat do cyklických bufferů, při HLD = on se neukládají žádná data	Bool

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

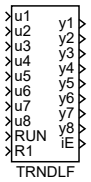
n	Počet signálů (bufferů) v trendu ↓1 ↑64 ⊙8	Long (I32)
l	Počet vzorků pro každý buffer trendu ↓2 ↑268435000 ⊙1000	Long (I32)
btype	Typ všech použitých bufferů trendu ⊙8 1 ..... Bool 4 ..... Long 7 ..... Float 2 ..... Byte 5 ..... Word 8 ..... Double 3 ..... Short 6 ..... DWord 10 .... Large	Long (I32)
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 · 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 · pfac · 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

`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 `RTC`. Ostatní volby jsou určeny pouze pro ladicí nebo speciální účely. ⊙1	`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 (pořadí signálu).	`String`

TRNDLF – * Záznam trendů v reálném čase (lock-free)

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

`u1..u8`	Analogové vstupy bloku určené pro zpracování a ukládání do trendu	`Double (F64)`
`RUN`	Povolení běhu algoritmu	`Bool`
`R1`	Signál pro vymazání obsahu trendového bloku. Data jsou mazána při každém spuštění bloku, je-li `R1` = `on`. Vstup má přednost před vstupem `RUN`.	`Bool`

Parametry

`n`	Počet signálů (bufferů) v trendu	↓1 ↑8 ⊙8	`Long (I32)`
`l`	Počet vzorků pro každý buffer trendu	↓0 ↑268435000 ⊙1024	`Long (I32)`
`btype`	Typ všech použitých bufferů `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)	⊙8	`Long (I32)`
`Title`	Název trendu	⊙`Trend Title`	`String`
`timesrc`	Zdroj časových značek	⊙1	`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`

Výstupy

`y1`	První analogový výstup bloku	`Double (F64)`
`y2`	Druhý analogový výstup bloku	`Double (F64)`
`y3`	Třetí analogový výstup bloku	`Double (F64)`
`y4`	Čtvrtý analogový výstup bloku	`Double (F64)`
`y5`	Pátý analogový výstup bloku	`Double (F64)`
`y6`	Šestý analogový výstup bloku	`Double (F64)`
`y7`	Sedmý analogový výstup bloku	`Double (F64)`
`y8`	Osmý analogový výstup bloku	`Double (F64)`
`iE`	Kód chyby ukládání (po bitech)	`Long (I32)`

TRNDVLF – * Záznam trendů v reálném čase (pro vektory, lock-free)

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

`uVec`	Vektorový signál určený k uložení	`Reference`
`HLD`	Pozastavení	`Bool`
`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`

Parametry

`n`	Počet signálů (bufferů) v trendu	↓1 ↑64 ⊙8	`Long (I32)`
`l`	Počet vzorků pro každý buffer trendu	↓2 ↑268435000 ⊙1024	`Long (I32)`
`btype`	Typ všech použitých bufferů `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)	⊙8	`Long (I32)`
`Title`	Název trendu	⊙`Trend Title`	`String`
`timesrc`	Zdroj časových značek	⊙1	`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 (pořadí signálu).		`String`

Výstup

iE	Kód chyby i ..... obecná chyba systému REXYGEN	Error

10.4 Správa archivů

AFLUSH – Vynucené zapsání archivu

Symbol bloku

Licence: STANDARD

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

FLUSH	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 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)

Kapitola 11

STRING – Bloky pro práci s řetězci

Obsah

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

`scv`	Textová hodnota		`String`
`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65520	`Long (I32)`

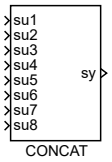
Výstup

`sy`	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

len	Minimální délka výstupního řetězce	↓0 ↑30	Long (I32)
mode	Formát výstupního textu	⊙1	Long (I32)
	1 ..... zarovanat vpravo, vyplnit mezerami		
	2 ..... zarovnat vpravo, vyplnit nulami		
	3 ..... zarovnat vlevo, vyplnit mezerami		
radix	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

`su`	Vstupní textová hodnota		`String`

Parametr

`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65520	`Long (I32)`

Výstup

`len`	Délka vstupního textu		`Long (I32)`

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

`jtxt`	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)`

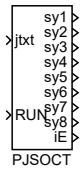
Výstupy

`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

`jtxt`	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)`

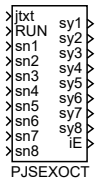
Výstupy

`sy1..8`	Výstupní textová hodnota	`String`
`iE`	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

`jtxt`	Text v JSON formátu		`String`
`RUN`	Povolení běhu algoritmu		`Bool`
`sn1..8`	Jméno objektu v textu formátu JSON		`String`

Parametry

`name1..8`	Jméno objektu v textu formátu JSON		`String`
`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65520	`Long (I32)`

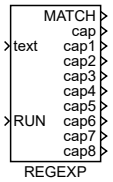
Výstupy

`sy1..8`	Výstupní textová hodnota		`String`
`iE`	Kód chyby		`Error`

REGEXP – * Regular expresion 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

`text`	Text k rozpoznání		`String`
`RUN`	Povolení běhu algoritmu		`Bool`

Parametry

`expr`	Regulární výraz k rozpoznání		`String`
`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65534	`Long (I32)`
`bufmax`	Velikost intrní pracovní paměti (0 = automaticky)	↓0 ↑10000000	`Long (I32)`

Výstupy

`MATCH`	Příznak rozpoznání	`Bool`
`cap`	Rozpoznaný text odpovídající celému regulárnímu výrazu	`String`
`cap1`	Rozpoznaný text odpovídající 1. závorce v regulárním výrazu	`String`
`cap2`	Rozpoznaný text odpovídající 2. závorce v regulárním výrazu	`String`
`cap3`	Rozpoznaný text odpovídající 3. závorce v regulárním výrazu	`String`
`cap4`	Rozpoznaný text odpovídající 4. závorce v regulárním výrazu	`String`
`cap5`	Rozpoznaný text odpovídající 5. závorce v regulárním výrazu	`String`
`cap6`	Rozpoznaný text odpovídající 6. závorce v regulárním výrazu	`String`
`cap7`	Rozpoznaný text odpovídající 7. závorce v regulárním výrazu	`String`
`cap8`	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

u	Analogový vstupní signál	Double (F64)

Výstup

sy	Výstupní textová hodnota	String

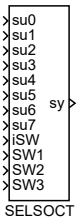
Parametry

prec	Přesnost (počet cifer)	↓0 ↑20	Long (I32)
mode	Formát výstupního textu	⊙1	Long (I32)
	1 ..... nejvhodnější – automaticky se vybere normální nebo exponenciální formát; prec je maximální počet číslic (nevýznamné nuly se nezobrazují)		
	2 ..... normální – formát s pevným počtem desetiných míst; prec je počet míst za desetinou tečkou (nevýznamné nuly se zobrazují)		
	3 ..... exponenciální – vědecký formát; prec 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`

Kapitola 12

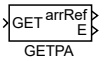
PARAM – Bloky pro manipulaci s parametry

Obsah

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.mereni.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 číst 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 déletrvající 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 data se čtou jen při náběžné hraně na tomto vstupu.	Bool

Výstupy

arrRef	Odkaz na pole (vektor nebo matice)	Reference
E	Příznak chyby	Bool

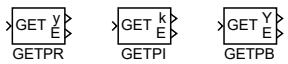
Parametry

sc	Jméno vzdáleného parametru	String
GETF	Načtení parametru pouze po vyžádání off ... režim průběžného čtení parametru on .... režim jednorázového přečtení parametru po náběžné hraně na vstupu GET	Bool
nmax	Maximální velikost vektoru (pole) ↓10 ⊙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. ⊙8 2 ..... Byte 5 ..... Word 8 ..... Double 3 ..... Short 6 ..... DWord 10 .... Large 4 ..... Long 7 ..... Float	Long (I32)

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 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 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 déletrvající 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

`GET`	Vstup pro jednorázové přečtení parametru	`Bool`

Výstupy

`y`	Hodnota parametru, výstup bloku `GETPR`	`Double (F64)`
`k`	Hodnota parametru, výstup bloku `GETPI`	`Long (I32)`
`Y`	Hodnota parametru, výstup bloku `GETPB`	`Bool`
`E`	Příznak chyby `off` ... bez chyby `on` .... nastala chyba	`Bool`

Parametry

`sc`	Jméno vzdáleného parametru podle výše uvedených pravidel	`String`
`GETF`	Zapnutí manuálního čtení vzdáleného parametru `off` ... režim průběžného čtení parametru `on` .... režim jednorázového přečtení parametru po náběžné hraně na vstupu `GET`	`Bool`

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 off ... režim průběžného čtení parametru on .... režim jednorázového přečtení parametru	Bool
nmax	Rezervovaná paměť pro řetězec	Long (I32)

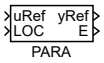
Výstupy

sy	Hodnota parametru	String
E	Příznak chyby off ... bez chyby on .... nastala chyba	Bool

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 umožnuje změnu jednoho svého parametru změnou vstupu. Parametr `apar` se může měnit podle vstupu `uRef`.

Logický vstup `LOC` (LOCal) 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

`uRef`	Odkaz na pole (vektor nebo matice)	`Reference`
`LOC`	Aktivace lokálního režimu `off` ... parametr je ovládán vstupním signálem `on` .... lokální režim aktivován	`Bool`

Výstup

`yRef`	Odkaz na pole (vektor nebo matice)	`Reference`

Parametry

`SETS`	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.	`Bool`
`nmax`	Maximální (reservovaná) velikost pole `apar` ↓`10` ⊙`100`	`Long (I32)`
`etype`	Typ položek pole `apar` ⊙`8` `2` ..... Byte `5` ..... Word `8` ..... Double `3` ..... Short `6` ..... DWord `10` .... Large `4` ..... Long `7` ..... Float	`Long (I32)`
`par`	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

ip	Hodnota parametru	Long (I32)
LOC	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on .... lokální režim aktivován	

Parametry

ipar	Interní hodnota parametru ⊙1	Long (I32)
pupstr	Definice seznamu konstant ⊙1: option A|2: option B|3: option C	String
NUM	Číslo ve výstupním textu	Bool
LIST	Režim bitového pole	Bool
SATF	Saturace (zda se na výstup nastaví nedefinované hodnoty)	Bool

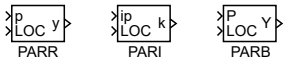
Výstupy

iy	Celočíselný výstupní signál	Long (I32)
sy	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

p	Hodnota parametru (blok PARR)	Double (F64)
ip	Hodnota parametru (blok PARI)	Long (I32)
P	Hodnota parametru (blok PARB)	Bool
LOC	Aktivace lokálního režimu off ... parametr je ovládán vstupním signálem on .... lokální režim aktivován	Bool

Výstup

y	Analogový výstupní signál bloku PARR	Double (F64)
k	Celočíselný výstupní signál bloku PARI	Long (I32)
Y	Logický výstupní signál bloku PARB	Bool

Parametry

par	Interní hodnota parametru bloku PARR	⊙1.0	Double (F64)

`ipar`	Interní hodnota parametru bloku `PARI`	⊙`1`	`Long (I32)`
`PAR`	Interní hodnota parametru bloku `PARB`	⊙`on`	`Bool`
`SATF`	Omezení výstupu pro bloky `PARR` a `PARI` `off` ... signál není omezen `on` .... saturační meze jsou aktivní		`Bool`
`hilim`	Horní saturační mez (bloky `PARR` a `PARI`)	⊙`1.0`	`Double (F64)`
`lolim`	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)

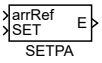
Výstup

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:Outport).

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: ".CNDR: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: "%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.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 SETPSslouží 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 parametrů 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.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 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

p	Požadovaná hodnota parametru, vstup bloku SETPR	Double (F64)
ip	Požadovaná hodnota parametru, vstup bloku SETPI	Double (F64)
P	Požadovaná hodnota parametru, vstup bloku SETPB	Double (F64)
SET	Vstup pro jednorázový zápis parametru	Bool

Výstupy

y	Hodnota parametru, výstup bloku SETPR	Double (F64)
k	Hodnota parametru, výstup bloku SETPI	Long (I32)
Y	Hodnota parametru, výstup bloku SETPB	Bool
E	Příznak chyby off ... bez chyby on .... nastala chyba	Bool

Parametry

sc	Jméno vzdáleného parametru podle výše uvedených pravidel	String
SETF	Zapnutí manuálního zápisu vzdáleného parametru off ... režim průběžného nastavování parametru on .... režim jednorázového nastavení parametru po náběžné hraně na vstupu SET	Bool

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)

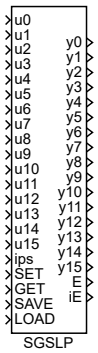
Výstupy

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 `sc`*i* prázdný (nezadaný), není žádný parametr specifikován, jinak parametry `sc`*i* 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, $i = 0 \ldots 15$ volit různý způsob zadání `sc`i, 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 specifikaci <blok> v parametrech sc$i$. V případě, že je broot = '.', je výsledek stejný, jako by parametr obsahoval cestu k subsystému, do nějž 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 sc$i$ 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 modifovanou 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 sc$i$, $i = 0, \ldots, m$, kde $m < 16$ je maximální číslo, pro něž je parametr sc$m$ 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 sc$i$ obsahuje jedinou dvojici <blok>:<param> (viz 1. syntaxe výše). Tato skutečnost umožňuje přecházet mezi oběma syntaxemi parametrů sc$i$, aniž by musel být soubor upravován.

Při ukládání malého počtu hodnot můžete rovněž využít blok SILO.

Vstupy

ui	i-tý analogový vstupní signál, $i = 0, \ldots, 15$	Double (F64)
ips	Číslo sady parametrů (číslováno od 0)	Long (I32)
SET	Přečtení vstupů ui a nastavení parametrů sady ips na jejich hodnoty	Bool
GET	Přečtení parametrů sady ips a nastavení výstupů yi na jejich hodnoty	Bool
SAVE	Uložení parametrů sady ips do souboru na disk cílového zařízení	Bool
LOAD	Načtení parametrů sady ips ze souboru na disku cílového zařízení	Bool

Výstupy

yi	i-tý analogový výstupní signál, $i = 0, \ldots, 15$	Double (F64)
E	Příznak chyby off ... bez chyby on .... nastala chyba, viz výstup iE	Bool

`iE`	Chybový nebo varovný výstup poslední operace `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 `ips` 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 `ips` je mimo přípustný rozsah	`Long (I32)`

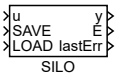
Parametry

`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`, nečte se při inicializaci žádná sada ↓`-1` ↑`15`	`Long (I32)`
`iprec`	Počet platných číslic pro zápis hodnoty typu `double` 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 `sc`i, viz popis v textu výše ⊙.	`String`
`sc`i	Řetězce specifikující připojení vstupů `u`i a výstupů `y`i, $i = 0, ..., 15$ 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

u	Vstupní signál	Double (F64)
SAVE	Uložení vstupní hodnoty do souboru	Bool
LOAD	Načtení hodnoty výstupu ze souboru	Bool

Parametry

fname	Jméno souboru pro ukládání/načítání parametrů	String
CSF	Příznak pro průběžné ukládání	Bool
CLF	Příznak pro průběžné načítání	Bool
yerr	Náhradní hodnota pro případ chyby	Double (F64)

Výstupy

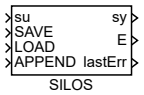
y	Výstupní signál	Double (F64)
E	Příznak chyby	Bool

`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

su	Vstupní řetězec	⊙0	String
SAVE	Uložení vstupního řetězce do souboru		Bool
LOAD	Načtení řetězce ze souboru		Bool
APPEND	Načtení řetězce ze souboru		Bool

Výstupy

sy	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`
`LLO`	Načítání jen poslední řádky		`Bool`
`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65520	`Long (I32)`

Kapitola 13

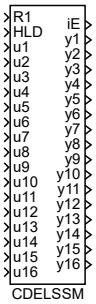
MODEL – Simulace dynamických systémů

Obsah

CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním

Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok CDELSSM (Continuous State Space Model with time DELay) simuluje chování lineárního spojitého systému s dopravním zpožděním *del* ve stavové reprezentaci

$$\begin{aligned} \frac{dx(t)}{dt} &= A_c x(t) + B_c u(t - del),\ x(0) = x0 \\ y(t) &= C_c x(t) + D_c u(t), \end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ 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 $x0$ 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

$$\begin{aligned} x((k+1)T) &= A_d x(kT) + B_{d1} u((k-d)T) + B_{d2} u((k-d+1)T),\ x(0) = x0 \\ y(kT) &= C_c x(kT) + D_c u(kT), \end{aligned}$$

kde $k \in \{1, 2, \ldots\}$ 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 diskrétních matic A_d, B_{d1} a B_{d2} je založen na metodě popsané v [4], využívající Padéových aproximaci maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak 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 `x0`. 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

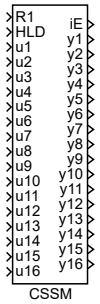
`UD`	Příznak použití matice `Dc`. Pokud je `UD`=`off`, matice `Dc` se při simulaci nepoužívá (chová se jako by byla nulová).	`Bool`
`del`	Dopravní zpoždění modelu [s]. ↓`0.0`	`Double (F64)`
`is`	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému. ↓`0` ↑`4` ⊙`2`	`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)`

`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 `Dc` se nekontrolují.	`Double (F64)`
`x0`	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

$$\begin{aligned} \frac{dx(t)}{dt} &= A_c x(t) + B_c u(t),\ x(0) = x0 \\ y(t) &= C_c x(t) + D_c u(t), \end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ 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 $x0$ 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

$$\begin{aligned} x((k+1)T) &= A_d x(kT) + B_d u(kT),\ x(0) = x0 \\ y(kT) &= C_c x(kT) + D_c u(kT), \end{aligned}$$

kde $k \in \{1, 2, \ldots\}$ 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

$$\begin{aligned} A_d &= e^{A_c T} \\ B_d &= \int_0^T e^{A_c \tau} B_c d\tau \end{aligned}$$

Výpočet diskrétních matic A_d a B_d je založen na metodě popsané v [4], využívající Padéových aproximaci maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak 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 `x0`. 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)`

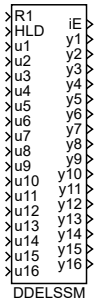
Parametry

`UD`	Příznak použití matice `Dc`. Pokud je `UD`=`off`, matice `Dc` se při simulaci nepoužívá (chová se jako by byla nulová).	`Bool`
`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 `Dc` 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

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k-d),\ x(0) = x0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ 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 $x0$ 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 x0. 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 `Bd`.	`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)`

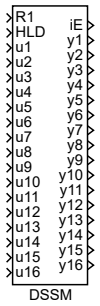
Parametry

`UD`	Příznak použití matice `Dd`. Pokud je `UD=off`, matice `Dd` se při simulaci nepoužívá (chová se jako by byla nulová).	`Bool`
`del`	Dopravní zpoždění modelu [s]. ↓`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)`

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

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k),\ x(0) = x0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ 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 $x0$ 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 x0. 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 Bd.	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)`

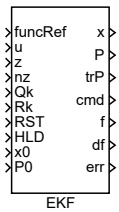
Parametry

`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 $\mathrm{d}x/\mathrm{d}t = f(x,u) + w(t), y = h(x,u) + v(t)$ 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 $x0$ a počáteční kovariancí $P0$. 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) ↓5 ↑10000 ⊙20		`Long (I32)`
`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)

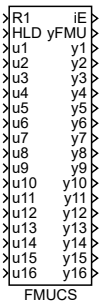
Výstupy

`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)`

FMUCS – Import modelu FMU CS (pro Co-Simulation)

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `FMUCS` umožnuje simulovat dané FMU ve formátu Co-Simulation (tj. včetně FMU, obsahujícího kromě modelu i simulační solver).

V parametru `FMUPath` je uložena úplná cesta k souboru modelu s příponou `.FMU` na počítači, kde je tento blok konfigurován. Tento blok předpokládá, že načtený model má maximálně 16 vstupů (symbolicky označených `u1`..`u16`), 16 výstupů (`y1`..`y16`) a 16 parametrů (`p1`..`p16`), které jsou po načtení modelu namapovány na jim pořadím odpovídajícím vstupům, výstupům a parametrům modelu FMU, které lze zjistit pomocí informačního bloku `FMUINFO`. Protože FMU může mít velký počet parametrů, lze jich pro mapování zvolit jen několik pomocí seznamu `SelPars`, jehož význam je popsán v bloku `FMUINFO`. Všechny vstupy `u1`..`u16`, výstupy `y1`..`y16` i parametry `p1`..`p16` jsou typu `Double (F64)`, který odpovídá typu `fmu2Real` ze specifikace [5]. Pokud má některý z těchto signálů v dané FMU typ `fmu2Integer` nebo `fmu2Boolean`, je v tomto bloku na tento typ automaticky zkonvertován. Signály typu `fmu2String` nejsou v tomto bloku podporovány.

První dva vstupy řídí běh simulace. Logický vstup `R1` umožňuje resetovat (inicializovat) simulaci do počátečních podmínek. Náběžnou hranou (`0` → `1`) tohoto vstupu se simulace zastaví, sestupnou hranou (`1` → `0`) se simulace spustí z počátečních podmínek od simulačního času 0. Hodnota logického vstup `HLD` = `1` umožňuje pozastavit simulaci, je-li `HLD` = `0` simulace běží (tj. i v případě, že vstup `HLD` není připojen).

Výstup `iE` indikuje výskyt chyby simulace, pokud jeho hodnota je nenulová. Vyskytne-li se při simulaci chyba, simulace se ukončí až do nového startu simulačního systému. Druhý výstup `yFMU` obsahuje odkaz na načtenou FMU. Tento odkaz může být připojen na vstup bloku `FMUINFO`, který umožní získat počáteční informace o FMU. Odpojením tohoto odkazu od bloku `FMUINFO` se žádné informace nevyhodnocují, což může start simulace zrychlit.

Zbylé parametry bloku `FMUCS` slouží pro řízení běhu simulace. Pokud je logický vstup `TSTOPDEF` = `1`, bude experiment ukončen v čase `tstop`, po němž se hodnoty výstupů

nemění (jako by byla nastaven vstup `HLD = 1`) až do následujícího resetu vstupem `R1`. Parametr `eps` určuje přesnost simulace.

Parametr `loglevel` určuje závažnost výpisů, které budou ukládány při běhu simulačního systému do systémového logu. Je třeba dát pozor na to, že velké množství výpisů během probíhající simulace tuto simulaci značně zpomaluje, a proto je vhodné množství výpisů ukládaných do systémového logu vizuálně zkontrolovat!

FMI 2.0 zavádí, kromě obyčejných parametrů, ještě tzv. *laditelné parametry* (tunable parameters). Obyčejné parametry lze nastavit ve fázi inicializace modelu (před zahájením vlastní simulace), kdežto laditelné parametry lze měnit i mezi jednotlivými simulačními kroky. Tento požadavek specifikace FMI 2.0 však některé simulační systémy nerespektují, např. OpenModelica. Proto byl v tomto bloku zaveden logický parametr `TUNEALLP`. Pokud je jeho hodnota nulová, nastavují se před každým simulačním krokem pouze laditelné parametry. Je-li `TUNEALLP = 1`, považují se všechny vybrané parametry (pomocí parametru `SelPars`) za laditelné, a proto se blok před každým simulačním krokem pokouší nastavit všechny.

Vstupy

`R1`	Reset bloku	`Bool`
`HLD`	Podržení aktuálního stavu modelu	`Bool`
`u1..u16`	Vstupní signály daného FMU	`Double (F64)`

Výstupy

`iE`	Kód chyby	`Error`
`yFMU`	Výstupní odkaz na instanci FMU	`Reference`
`y1..y16`	Výstupní signály daného FMU	`Double (F64)`

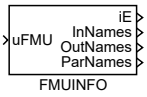
Parametry

`tstop`	Koncový čas simulace ↓`1e-06` ⊙`1.0`	`Double (F64)`
`eps`	Přesnost aproximace ↓`0.0` ↑`1.0` ⊙`1e-06`	`Double (F64)`
`loglevel`	Úroveň protokolování knihovny FMI do systémového logu ↓`0` ↑`7` ⊙`2` `0` ..... Nic `1` ..... Fatální chyba `2` ..... Chyba `3` ..... Varování `4` ..... Info `5` ..... Podrobný `6` ..... Ladění `7` ..... Všechno	`Long (I32)`
`SelPars`	Seznam vybraných parametrů FMU	`String`
`TUNEALLP`	Považuj všechny vybrané parametry za laditelné parametry	`Bool`
`p1..p16`	Vybrané parametry modelu FMU podle nastavení parametru `SelPars`	`Double (F64)`

FMUINFO – Informace o importovaném modelu FMU

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `FMUINFO` je pomocným blokem, který poskytuje informace o daném modelu ve formátu FMU, aniž by musel být uživatelem analyzován vnitřek souboru s příponou `.FMU`.

Na vstup `uFMU` bloku je připojen odkaz na instanci simulačního modelu FMU, která je výstupem výkonného simulačního bloku (např. výstup `yFMU` bloku `FMUCS`). Na výstupech `InNames`, `OutNames` a `ParNames` jsou řetězce obsahující seznamy názvů vstupů, výstupů a parametrů dané FMU v pořadí, v jakém je poskytují funkce použité knihovny FMI. Jednotlivé názvy jsou odděleny řetězcem uloženým v parametru `Separ` tohoto bloku. Protože typické modely obsahují velké množství parametrů (kromě parametrů modelu v nejvyšší úrovni hierarchie též parametry všech modelů v nižších úrovních hierarchie), umožňuje parametr `SelPars` zvolit požadované parametry, které se objeví ve výstupu `ParNames` (detaily viz tabulku parametrů níže).

Vstup

`uFMU`	Vstupní odkaz na instanci FMU	`Reference`

Výstupy

`iE`	Kód chyby	`Error`
`InNames`	Seznam jmen vstupů FMU oddělených parametrem `Separ`	`String`
`OutNames`	Seznam jmen výstupů FMU oddělených parametrem `Separ`	`String`
`ParNames`	Seznam jmen vybraných parametrů FMU oddělených parametrem `Separ`. Do seznamu jsou zařazeny pouze takové parametry, které odpovídají nastavení parametru `SelPars`, viz níže	`String`

Parametry

`SelPars`	Seznam vybraných parametrů FMU, které budou zařazeny do výstupu `ParNames`. Pokud je tento řetězec prázdný (neobsahuje žádný znak), budou do seznamu zařazeny všechny parametry z nejvyšší úrovně hierarchie modelu v daném FMU. V případě, že je tento řetězec neprázdný, obsahuje seznam parametrů, které se mají objevit na výstupu `ParNames`, oddělených znakem `,` (čárka)	`String`
`Separ`	Oddělovač jmen v řetězcových výstupech (nejlépe složený z jediného znaku), kterým budou oddělena jména vstupů, výstupů a parametrů ve výstupních řetězcích `InNames`, `OutNames` a `ParNames` ⊙,	`String`

FOPDT – Model systému 1. řádu s dopravním zpožděním

Symbol bloku

Licence: STANDARD

Popis funkce

Blok FOPDT realizuje diskrétní simulátor lineárního systému prvního řádu s přídavným dopravním zpožděním, který je popsán následující přenosovou funkcí:

$$P(s) = \frac{\mathtt{k0}}{(\mathtt{tau} \cdot s + 1)} \cdot e^{-\mathtt{del} \cdot s}$$

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)

MDL – Model procesu

Symbol bloku

Licence: STANDARD

Popis funkce

Blok MDL 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.

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

u	Analogový vstupní signál	Double (F64)
k0	Statické zesílení	Double (F64)
del	Dopravní zpoždění [s]	Double (F64)
tau1	První časová konstanta	Double (F64)
tau2	Druhá časová konstanta	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

Parametry

nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (používá se pro interní alokaci paměti) ↓10 ↑10000000 ⊙1000	Long (I32)

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/tv, kde tv je parametr bloku. Při UP = on (DN = on) otevírání probíhá až do úplného otevření y = hilim (úplného zavření y = hilim) ventilu. Krajní poloha otevření (zavření) je signalizována koncovým spínačem HS (LS). Počáteční poloha ventilu po spuštění je y = y0. 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

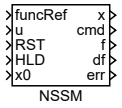
Parametry

y0	Počáteční poloha ventilu		Double (F64)
tv	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 $\mathrm{d}x/\mathrm{d}t = f(x,u), y = h(x,u)$ pro spojitý čas nebo $x(k+1) = f(x(k),u(k)), y(k) = h(x(k),u(k))$ pro diskrétní 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 $\mathrm{d}f(x,u)/\mathrm{d}x$ 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 spouštění 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 $\mathrm{d}f(x,u)/\mathrm{d}x$ 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 aktivacu 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 $f = h(x,u)$, $\mathrm{d}f = \mathrm{d}h(x,u)/\mathrm{d}x$.

Vstupy

`funcRef`	Odkaz na spolupracující blok REXLANG	`Reference`
`u`	Vektor vstupů modelu	`Reference`
`RST`	Reset bloku	`Bool`
`HLD`	Pozastavení	`Bool`
`x0`	Vektor počáteční hodnoty stavu modelu	`Reference`

Parametry

`nmax`	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓5 ↑10000 ⊙20	`Long (I32)`
`solver`	Metoda numerické integrace ⊙2 `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)	`Long (I32)`

Výstupy

`x`	Vektor stavu modelu	`Reference`
`y`	Výstupní vektor modelu	`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)`

SOPDT – Model systému 2. řádu s dopravním zpožděním

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SOPDT realizuje diskrétní simulátor lineárního systému druhého řádu s přídavným dopravním zpožděním, který je alternativně popsán, v závislosti na parametru itf, následujícími přenosovými funkcemi:

$$\begin{aligned}
\texttt{itf} = 1: \quad P(s) &= \frac{\texttt{pb1} \cdot s + \texttt{pb0}}{s^2 + \texttt{pa1} \cdot s + \texttt{pa0}} \cdot e^{-\texttt{del} \cdot s} \\
\texttt{itf} = 2: \quad P(s) &= \frac{\texttt{k0}\,(\texttt{tau} \cdot s + 1)}{(\texttt{tau1} \cdot s + 1)\,(\texttt{tau2} \cdot s + 1)} \cdot e^{-\texttt{del} \cdot s} \\
\texttt{itf} = 3: \quad P(s) &= \frac{\texttt{k0} \cdot \texttt{om}^2 \cdot (\texttt{tau}/\texttt{om} \cdot s + 1)}{(s^2 + 2 \cdot \texttt{xi} \cdot \texttt{om} \cdot s + \texttt{om}^2)} \cdot e^{-\texttt{del} \cdot s} \\
\texttt{itf} = 4: \quad P(s) &= \frac{\texttt{k0}\,(\texttt{tau} \cdot s + 1)}{(\texttt{tau1} \cdot s + 1)\,s} \cdot e^{-\texttt{del} \cdot s}
\end{aligned}$$

Diskrétní 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

itf	Tvar přenosové funkce 1 ..... obecný tvar 2 ..... reálné póly ve jmenovateli 3 ..... komplexní póly (kmitavý systém) 4 ..... systém s integrátorem	⊙1	Long (I32)
k0	Statické zesílení	⊙1.0	Double (F64)
tau	Časová konstanta v čitateli		Double (F64)
tau1	První časová konstanta ve jmenovateli	⊙1.0	Double (F64)

`tau2`	Druhá časová konstanta ve jmenovateli	⊙`1.0`	`Double (F64)`
`om`	Vlastní frekvence	⊙`1.0`	`Double (F64)`
`xi`	Relativní koeficient tlumení	⊙`1.0`	`Double (F64)`
`pb0`	Koeficient čitatele: s^0	⊙`1.0`	`Double (F64)`
`pb1`	Koeficient čitatele: s^1	⊙`1.0`	`Double (F64)`
`pa0`	Koeficient jmenovatele: s^0	⊙`1.0`	`Double (F64)`
`pa1`	Koeficient jmenovatele: s^1	⊙`1.0`	`Double (F64)`
`del`	Dopravní zpoždění [s]		`Double (F64)`
`nmax`	Délka vyrovnávací paměti pro dopravní zpoždění `del` (používá se pro interní alokaci paměti)	↓`10` ↑`10000000` ⊙`1000`	`Long (I32)`

Kapitola 14

MATRIX – Bloky pro maticové a vektorové operace

Obsah

Poznámky k imlementaci: 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 REXYGENvyžaduje 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

`filename`	Datový soubor s hodnotami oddělenými čárkou		`String`
`TRN`	Transponuj načtenou matici		`Bool`
`nmax`	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓2 ↑10000000 ⊙100		`Long (I32)`
`etype`	Typ prvků `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)	⊙8	`Long (I32)`
`acn`	Počáteční hodnota pole	⊙`[0 1 2 3]`	`Double (F64)`

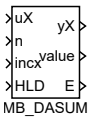
Výstup

`vec`	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

uX	Vstupní odkaz na vektor x	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
HLD	Pozastavení	Bool

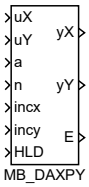
Výstupy

yX	Výstupní odkaz na vektor x	Reference
value	Návratová hodnota funkce	Double (F64)
E	Příznak chyby	Bool

MB_DAXPY – * Provádí y := a*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

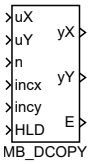
Výstupy

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

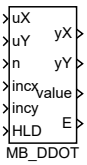
Výstupy

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

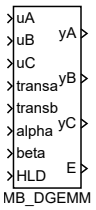
Výstupy

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*op(A)*op(B) + beta*C, where op(X) = X or 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

uA	Vstupní odkaz na matici A		Reference
uB	Vstupní odkaz na matici B		Reference
uC	Vstupní odkaz na matici C		Reference
transa	Transpozice matice A	↓0 ↑3	Long (I32)
transb	Transpozice matice B	↓0 ↑3	Long (I32)
alpha	Skalární koeficient alpha		Double (F64)
beta	Skalární koeficient beta		Double (F64)
HLD	Pozastavení		Bool

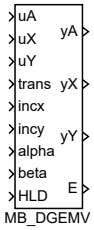
Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
yC	Výstupní odkaz na matici C	Reference
E	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

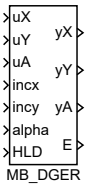
Výstupy

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

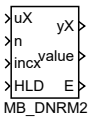
Výstupy

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



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

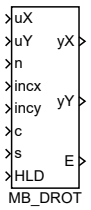
Výstupy

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

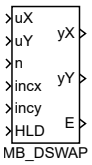
Výstupy

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`

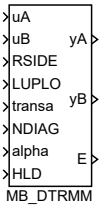
Výstupy

`yX`	Výstupní odkaz na vektor x	`Reference`
`yY`	Výstupní odkaz na vektor y	`Reference`
`E`	Příznak chyby	`Bool`

MB_DTRMM – * Provádí B := alpha*op(A)*B or B := alpha*B*op(A), where op(X) = X or 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

uA	Vstupní odkaz na matici A		Reference
uB	Vstupní odkaz na matici B		Reference
RSIDE	Operace je aplikována z pravé strany		Bool
LUPLO	Matice A je dolní trojúhelníková matice		Bool
transa	Transpozice matice A	↓0 ↑3	Long (I32)
NDIAG	Nepředpokládá se, že matice A má na diagonále jedničky		Bool
alpha	Skalární koeficient alpha		Double (F64)
HLD	Pozastavení		Bool

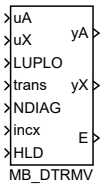
Výstupy

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

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

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

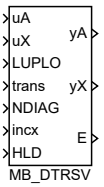
Výstupy

yA	Výstupní odkaz na matici A	Reference
yX	Výstupní odkaz na vektor x	Reference
E	Příznak chyby	Bool

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

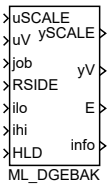
Výstupy

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`

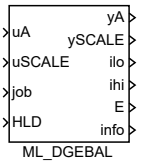
Výstupy

`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 výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou 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

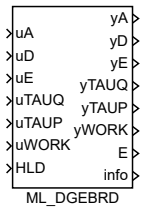
Výstupy

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 výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou 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 nepřípustnou 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

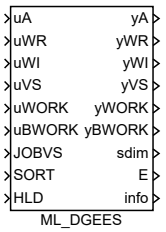
Výstupy

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 výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou 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`

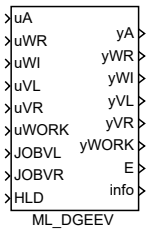
Výstupy

`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 nepřípustnou 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`

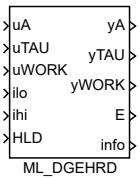
Výstupy

`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 výstleku 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`

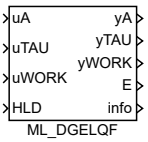
Výstupy

`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 výstleku 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 x 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

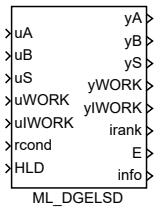
Výstupy

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 nepřípustnou 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`

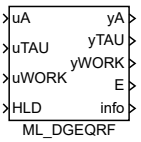
Výstupy

`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 výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou hodnotu	`Long (I32)`

ML_DGEQRF – * Výpočet QR factorizace reálné matice A s rozměry M x 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

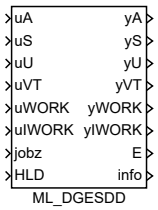
Výstupy

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 nepřípustnou hodnotu	Long (I32)

ML_DGESDD – * Výpočet singulární dekompozice (SVD) reálné matice A s rozměry M x 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 nepřípustnou 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 `0` ..... Celá `1` ..... Celá `2` ..... Horní `3` ..... Dolní	`Long (I32)`
`HLD`	Pozastavení	`Bool`

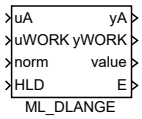
Výstupy

`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`

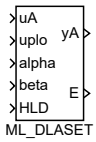
Výstupy

`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

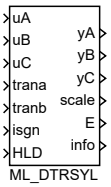
Výstupy

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 výstleku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou 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)

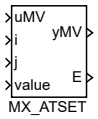
Výstupy

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)

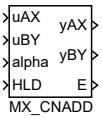
Výstupy

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`

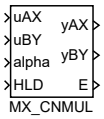
Výstupy

`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`

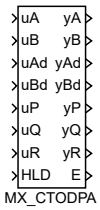
Výstupy

`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)

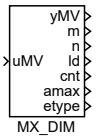
Výstupy

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`

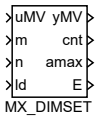
Výstupy

`yMV`	Výstupní reference na matici nebo vektor	`Reference`
`m`	Počet řádků matice	`Long (I32)`
`n`	Počet sloupců matice	`Long (I32)`
`ld`	Vedoucí dimenze (>= počtu řádků)	`Long (I32)`
`cnt`	Počet použitých prvků (počtu řádků * počet sloupců)	`Long (I32)`
`amax`	Počet reservovaných (alokovaných) prvků (>= 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



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 (>= počtu řádků)	Long (I32)

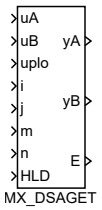
Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
n	Počet sloupců matice	Long (I32)
cnt	Počet alokovaných prvků (>= 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 0 ..... Celá 1 ..... Celá 2 ..... Horní 3 ..... Dolní	Long (I32)
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)

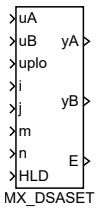
Výstupy

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 0 ..... Celá 1 ..... Celá 2 ..... Horní 3 ..... Dolní	Long (I32)
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`

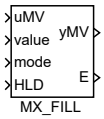
Výstupy

`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)

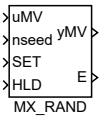
Výstup

yMat	Výstupní reference na matici	Reference

MX_RAND – * Náhodně vygerenovaná 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)`

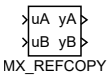
Výstupy

`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

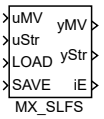
Výstupy

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 pripojený 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 provézt (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

`uMV`	Vstupní reference na matici nebo vektor	`Reference`
`uStr`	Text pro konverzi na matici/vektor nebo jméno souboru	`String`
`LOAD`	Povolení zápisu hodnoty do matice/vektoru	`Bool`
`SAVE`	Povolení zápisu hodnoty do souboru/textu	`Bool`

Parametry

`mode`	Režim spouštění konverze ⊙2 `1` ..... aktivované úrovní, do/z souboru – data jsou konvertována ze souboru do matice při `LOAD=on` a z matice do souboru při `SAVE=on`; 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 `LOAD` a z matice do souboru při náběžné hraně `SAVE`; 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 `uStr` do matice při `LOAD=on` a z matice na výstup `yStr` při `SAVE=on`; pokud jsou aktivní oba signály, je to chyba a žádná akce se neprovede `4` ..... aktivované hranou, do/z textu – data jsou konvertována ze vstupu `uStr` do matice při náběžné hraně na vstupu `LOAD` a z matice na výstup `yStr` při náběžné hraně `SAVE`; 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 `uStr` do matice v každém kroku algoritmu `6` ..... trvale matice do textu – data jsou konvertována z matice na výstup `yStr` 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	`Long (I32)`
`format`	Formát souboru/textu ⊙1 `1` ..... CSV `2` ..... CSV(středník) `3` ..... JSON `4` ..... MATLAB	`Long (I32)`
`prec`	Počet platných cifer pro každou hodnotu ↓0 ↑20 ⊙6	`Long (I32)`
`TRN`	Příznak transpozice matice	`Bool`
`nmax`	Rezervovaná paměť pro řetězec ↓0	`Long (I32)`

Výstupy

`yMV`	Výstupní reference na matici nebo vektor	`Reference`
`yStr`	Textová podoba matice/vektoru	`String`
`iE`	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)		

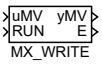
Výstup

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	⊙A	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 0 ..... Žádný 1 ..... Žádný 2 ..... Podrobný 3 ..... Informace 4 ..... Varování 5 ..... Chyba	⊙3	Long (I32)

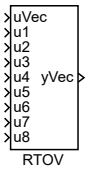
Výstupy

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

RTOV – Vektorový multiplexer

Symbol bloku

Licence: STANDARD



Popis funkce

Blok slouží pro vkladaní 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 REXYGENvždy maximální (alokovanou) velikost a aktuální velikost. Hodnoty, které by se zapsaly na indexy větší než maximální velikost nebo menší než 0 se nezapíší. 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 hodny 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 přivézt na vstup `uVec`.

Vstupy

`uVec`	Vektorový signál	`Reference`
`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)`

Parametry

`nmax`	Alokovaná velikost vektoru	↓0 ⊙8	`Long (I32)`
`offset`	Index prvního vstupu ve vektoru	↓0	`Long (I32)`
`n`	Počet použitých vstupů	↓0 ↑8 ⊙8	`Long (I32)`
`etype`	Typ prvků `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)	⊙8	`Long (I32)`

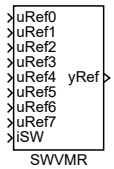
Výstup

`yVec`	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)

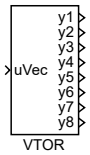
Výstup

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

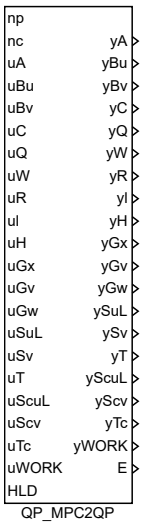
OPTIM – Bloky pro optimalizaci

Obsah

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
ul	Vstupní odkaz na celočíselný vektor indexů l		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

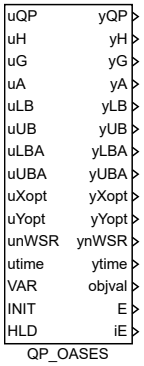
Výstupy

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
yl	Výstupní odkaz na celočíselný vektor indexů l	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 init() místo hotstart()	`Bool`
`HLD`	Pozastavení	`Bool`

Parametry

`hessianType`	Typ Hessovy matice	`Long (I32)`
`printLevel`	Úroveň výpisů	`Long (I32)`
`enableRamping`	Povolit tzv. rampování	`Bool`
`enableFarBounds`	Povolit použití tzv. dalekých mezí	`Bool`
`enableFlippingBounds`	Povolit používání tzv. převrácených mezí (flipping bounds)	`Bool`
`enableRegularisation`	Povolit regularizaci semidefinitní Hessovy matice	`Bool`
`enableFullLITests`	Povolit použití tvrdších testů lineární nezávislosti	`Bool`
`enableNZCTests`	Povolit testy nenulového zakřivení	`Bool`
`enableDriftCorrection`	Korekce frekvence nebo driftu (0 = off)	`Long (I32)`
`enableCholeskyRefact`	Frekvence plné refaktorizace projektovaného Hessiánu (0 = off)	`Long (I32)`
`enableEqualities`	Rovnosti se vždy považují za aktivní omezení	`Bool`
`terminationTolerance`	Ukončovací tolerance	`Double (F64)`
`boundTolerance`	Pokud se horní a spodní meze liší méně než o tuto toleranci, jsou považovány za shodné, tj. za podmínku s rovností	`Double (F64)`
`boundRelaxation`	Počáteční relaxace mezí pro startovací homotopii a počáteční hodnota dalekých mezí	`Double (F64)`
`epsNum`	Tolerance čitatele pro poměrové testy	`Double (F64)`
`epsDen`	Tolerance jmenovatele pro poměrové testy	`Double (F64)`
`maxPrimalJump`	Maximálně povolený skok v primálních proměnných pro testy lineární nezávislosti	`Double (F64)`
`maxDualJump`	Maximálně povolený skok v duálních proměnných pro testy lineární nezávislosti	`Double (F64)`
`initialRamping`	Počáteční hodnota pro strategii rampování	`Double (F64)`
`finalRamping`	Koncová hodnota pro strategii rampování	`Double (F64)`
`initialFarBounds`	Počáteční velikost dalekých mezí	`Double (F64)`
`growFarBounds`	Faktor zvětšování dalekých mezí	`Double (F64)`
`initialStatusBounds`	Počáteční stav mezí v první iteraci	`Long (I32)`
`epsFlipping`	Tolerance kvadrátu prvku na Choleského diagonále, která aktivuje tzv. převrácené meze (flipping bounds)	`Double (F64)`
`numRegularisationSteps`	Maximální počet po sobě jdoucích regularizačních kroků	`Long (I32)`
`epsRegularisation`	Měřítkovací faktor identické matice použitý pro regularizaci Hessiánu	`Double (F64)`
`numRefinementSteps`	Maximální počet kroků pro iteračních upřesnění	`Long (I32)`
`epsIterRef`	Tolerance předčasného ukončení pro iterativní upřesnění	`Double (F64)`
`epsLITests`	Tolerance pro testy lineární nezávislosti	`Double (F64)`

`epsNZCTests`	Tolerance pro testy nenulového zakřivení	`Double (F64)`

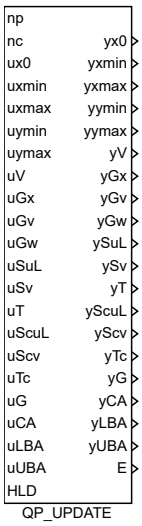
Výstupy

`yQP`	Výstupní odkaz na úlohu kvadratického programování	`Reference`
`yH`	Výstupní odkaz na Hessovu matici H	`Reference`
`yG`	Výstupní odkaz na gradientní vektor G	`Reference`
`yA`	Výstupní odkaz na matici omezení A	`Reference`
`yLB`	Výstupní odkaz na vektor dolních mezí LB	`Reference`
`yUB`	Výstupní odkaz na vektor horních mezí UB	`Reference`
`yLBA`	Výstupní odkaz na vektor dolních mezí LBA omezujících podmínek	`Reference`
`yUBA`	Výstupní odkaz na vektor horních mezí UBA omezujících podmínek	`Reference`
`yXopt`	Výstupní odkaz na vektor prvotního optimálního řešení	`Reference`
`yYopt`	Výstupní odkaz na vektor duálního optimálního řešení	`Reference`
`ynWSR`	Skutečný počet výpočtů pracovní množiny během inicializace	`Long (I32)`
`ytime`	Spotřebovaný čas CPU v sekundách pro celou inicializaci	`Double (F64)`
`objval`	Optimální hodnota účelové funkce	`Double (F64)`
`E`	Příznak chyby	`Bool`
`iE`	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	Vsýtupní 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

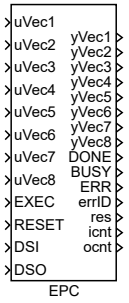
SPEC – Speciální bloky

Obsah

EPC – Blok pro spouštění externích programů

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 možné 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 řetězit, 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

`uVec1..uVec8`	Vstupní vektorové signály	`Reference`
`EXEC`	Náběžná hrana spouští externí program	`Bool`
`RESET`	Reset bloku (smaže vstupní i výstupní soubory a zastaví externí program)	`Bool`
`DSI`	Pozastavení vzorkování na vstupech	`Bool`
`DSO`	Pozastavení vzorkování na výstupech	`Bool`

Výstupy

`yVec1..yVec8`	Výstupní vektorové signály	`Reference`
`DONE`	Příznak skončení externího programu	`Bool`
`BUSY`	Příznak běhu externího programu	`Bool`
`ERR`	Příznak chyby	`Bool`
`errID`	Kód chyby `i` ..... obecná chyba systému REXYGEN	`Error`
`res`	Návratový kód externího programu	`Long (I32)`
`icnt`	Aktuální číslo vzorku na vstupech	`Long (I32)`
`ocnt`	Aktuální číslo vzorku na výstupech	`Long (I32)`

Parametry

`cmd`	Externí program	`String`
`ifns`	Vstupní soubory (oddělené středníkem) ⊙`epc_uVec1;epc_uVec2`	`String`
`ofns`	Výstupní soubory (oddělené středníkem) ⊙`epc_yVec1;epc_yVec2`	`String`
`sl`	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)`
`ifm`	Maximální počet vzorků ve vstupním souboru ⊙`10000`	`Long (I32)`
`format`	Formát vstupních a výstupních souborů ⊙`1` `1` ..... textový (pouze hodnoty oddělené mezerou) `2` ..... CSV (desetinná tečka a čárky) `3` ..... CSV (desetinná čárka a středníky)	`Long (I32)`
`nmax`	Maximální délka výstupních vektorů ↓`2` ↑`1000000` ⊙`100`	`Long (I32)`

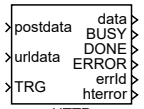
Poznámky

- 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 podtržítko. 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 podtržítko.
- 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

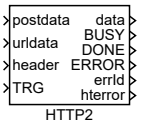
`data`	Data z odpovědi	`String`
`BUSY`	Odesílání HTTP požadavku	`Bool`

`DONE`	HTTP požadavek byl zpracován	`Bool`
`ERROR`	Příznak chyby	`Bool`
`errId`	Kód chyby	`Error`
`hterror`	HTTP odpověď	`Long (I32)`

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)`

`datamax`	Alokovaná paměť pro odpověď na příkaz HTTP ↓128 ↑10000000 ⊙64000	`Long (I32)`

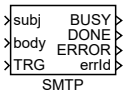
Výstupy

`data`	Data z odpovědi	`String`
`BUSY`	Odesílání HTTP požadavku	`Bool`
`DONE`	HTTP požadavek byl zpracován	`Bool`
`ERROR`	Příznak chyby	`Bool`
`errId`	Kód chyby	`Error`
`hterror`	HTTP odpověď	`Long (I32)`

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` ..... None `2` ..... StartTLS `3` ..... TLS	⊙1	`Long (I32)`
`user`	Uživatelské jméno		`String`
`password`	Heslo		`String`
`domain`	domain		`String`
`auth`	Metoda autentifikace `1` ..... Login `2` ..... Plain	⊙1	`Long (I32)`
`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 ruzné 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 výpocty 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/m3). 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 [m3/kg]
- rho Hustota [kg/m3]
- s Měrná entropie
- u Měrná vnitřní energie [kJ/kg]
- Cp Měrná tepelná kapacita pri konstantním tlaku [kJ/kg/K]
- Cv Měrná tepelná kapacita pri 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` ..... MPa `2` ..... bar `3` ..... kPa	⊙1	`Long (I32)`
`tunit`	Jednotka teploty `1` ..... K `2` ..... ∘C	⊙1	`Long (I32)`

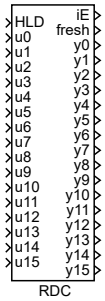
Výstupy

`y`	Hodnota výstupní veličiny	`Double (F64)`
`E`	Příznak chyba (výstupní hodnota je neplatná)	`Bool`

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 přenese `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

`HLD`	Vstup pozastavující činnost bloku. Pokud je `HLD = on`, blok nevysílá ani nepřijímá žádné pakety.	`Bool`
`u0..u15`	Hodnoty, které se předávají/zapisují na hodnoty `y0` až `y15` spolupracujícího bloku RDC	`Double (F64)`

Výstupy

`iE`	Zobrazuje kód poslední chyby. Použitá čísla jsou v následující tabulce: `0` ..... Bez chyby *Trvalé chyby, vznikají v inicializační části bloku, systém je nedokáže sám opravit (< 0)* `-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 `lport`) `-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 `RDC` 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	`Long (I32)`

	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)`

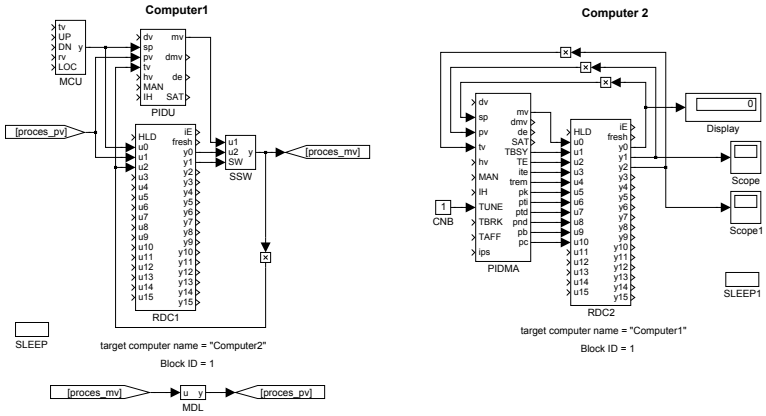
Parametry

`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 `pk`, `pti`, `ptd` 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 `pk`, `pti`, `ptd` 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 `u0` bloku `RDC1`, s tím, co původně vedlo na výstup `y0` bloku `RDC2`, atd. pro `u1`, `y1`,....



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

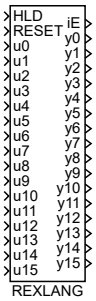
`<pin>` je název signálu nabývající jedné z hodnot:

- `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



Popis funkce

V některých případech se může stát, že je do řídícího algoritmu nutné implementovat funkci, kterou nelze efektivně vytvořit z dostupné množiny bloků. Pro takový účel byl vyvinut blok `REXLANG`, který implementuje algoritmus definovaný skriptovacím jazykem. Je použit skriptovací jazyk velice podobný jazyku C (nebo Java).

Skriptovací jazyk

Jak již bylo řečeno, skriptovací jazyk vychází z jazyka C a je mu velmi podobný, nicméně existují určité rozdíly a omezení:

- Jsou podpořeny datové typy `double`, `long` a `string` (lze použít i `int`, `short`, `bool`, které se interně zpracovávají jako `long`, a typ `float`, který se interně zpracovává jako `double`). Není implementován `typedef`.
- Nejsou implementovány pointery a struktury. V kontextu bloku `REXLANG` lze však definovat pole a používat indexy (operátor `[ ]`). Vstupy, výstupy a parametry bloku nemohou být pole.
- Není zaveden operátor '`,`'.
- Z preprocesoru jsou podpořeny příkazy `#include`, `#define`, `#ifdef` .. [`#else` ..] `#endif`, `#ifndef` .. [`#else` ..] `#endif` (tzn. není podpořeno `#pragma` a zejména `#if` .. [`#else` ..] `#endif`).
- Nejsou implementovány standardní knihovny ANSI C, je však definována většina funkcí z `math.h` a potom některé další (viz dále).
- Jsou definována klíčová slova `input`, `output` a `parameter` pro napojení na vstupy, výstupy a parametry bloku. Dále jsou definovány systémové funkce pro řízení běhu a diagnostiku (viz dále).

- Kromě funkce `main()`, která se volá periodicky při běhu řídícího systému mohou být implementovány funkce `init()` (volá se při startu), `exit()` (volá se při ukončení řídícího algoritmu) a `parchange()` (volá se v systému REXYGEN při změně jakéhokoliv z parametrů).
- Ve funkcích a procedurách bez parametrů musí být v deklaraci explicitně uvedeno `void`.
- Nelze přetěžovat identifikátory, tj. nelze používat klíčová slova a názvy vestavěných funkcí jako identifikátor, nelze pojmenovat stejně globální a lokální proměnnou.
- Nelze inicializovat pole, ať už globální nebo lokální.
- Uživatelské návratové hodnoty funkcí `main()`, `init()` a `exit()` jsou zapsány na výstup `iE`. Hodnoty `< -99` zastaví běh algoritmu bloku `REXLANG` (pro další běh je nutná reinicializace vstupem `RESET`). Uživatelské návratové hodnoty:

  `iE >= 0` ... Bez chyby

  `0 > iE >= -99` ... Warning, provádění algoritmu bloku beze změny

  `iE < -99` ... Chybový stav, provádění algoritmu bloku zastaveno

Syntaxe skriptovacího jazyka

Syntaxe skriptovacího jazyka vychází z jazyka C, přičemž nejsou podpořeny pointery a jiné typy než `long` a `double`. Navíc jsou definována klíčová slova `input`, `output` a `parameter`, která slouží pro odkazování na vstupy, výstupy a parametry bloku. Syntaxe je následující:

- `<typ> input(<číslo vstupu>) <jméno proměnné>;`
- `<typ> output(<číslo výstupu>) <jméno proměnné>;`
- `<typ> parameter(<číslo parametru>) <jméno proměnné>;`

Proměnné typu `input` a `parameter` lze pouze číst a do proměnných typu `output` lze pouze přiřazovat. Například:

```
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

Ve skriptovacím jazyce je možné používat následující funkce:

- **Matematické** (viz ANSI C, soubor `math.h`):
  `atan`, `sin`, `cos`, `exp`, `log`, `sqrt`, `tan`, `asin`, `acos`, `fabs`, `fmod`, `sinh`, `cosh`, `tanh`, `pow`, `atan2`, `ceil`, `floor` a `abs` Význam funkcí by měl být zřejmý, jen je potřeba zmínit, že funkce `abs` pracuje s celými čísly. Pro výpočet absolutní hodnoty desetinného čísla slouží funkce `fabs`.

- **Vektorové** (v ANSI C nejsou)

  `double max([n,]val1,...,valn)`
  : Vrací hodnotu největšího prvku. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double max(n,vec)`
  : Vrací hodnotu největšího prvku z vektoru `vec`.

  `double min([n,]val1,...,valn)`
  : Vrací hodnotu nejmenšího prvku. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double min(n,vec)`
  : Vrací hodnotu nejmenšího prvku z vektoru `vec`.

  `double poly([n,]x,an,...,a1,a0)`
  : Vypočte hodnotu polynomu $y = an * x^n + \ldots + a1 * x + a0$. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double poly(n,x,vec)`
  : Vypočte hodnotu polynomu $y = vec[n] * x^n + \ldots + vec[1] + x + vec[0]$.

  `double scal(n,vec1,vec2)`
  : Vypočte skalární součin $y = vec1[0] * vec2[0] + \ldots + vec1[n-1] * vec2[n-1]$.

  `double scal(n,vec1,vec2,skip1,skip2)`
  : Vypočte skalární součin $y = vec1[0]*vec2[0]+vec1[skip1]*vec2[skip2]+\ldots+vec1[(n-1)*skip1]*vec2[(n-1)*skip2]$. Toto je výhodné, pokud vektory představují matice a potřebujeme vynásobit sloupce (resp. řádky, pokud je matice uložena po sloupcích).

  `double conv(n,vec1,vec2)`
  : Vypočte konvolutorní součin $y = vec1[0]*vec2[n-1]+vec1[1]*vec2[n-2]+\ldots+vec1[n-1]*vec2[0]$.

  `double sum(n,vec)`
  : Sečte prvky vektoru, tj. $y = vec[0] + vec[1] + \ldots + vec[n-1]$.

  `double sum([n,]val1,...,valn)`
  : Sečte prvky, tj. $y = val1 + val2 + \ldots + valn$. Funkce má nepovinný první parametr, který určuje počet prvků.

`[]array([n,],an-1,...,a1,a0)`
: Vrací pole/vektor, které obsahuje hodnoty parametrů. Funkce/operátor má nepovinný první parametr, který určuje počet prvků. Návratový typ se volí automaticky podle typu parametrů (musí být všechny stejného typu).

`[]subarray(idx,vec)`
: Vrací pole/vektor, které představuje pole `vec` od indexu `idx`. Návratový typ se volí automaticky podle typu vstupního pole.

`copyarray(count,vecSource,idxSource,vecTarget,idxTarget)`
: Kopíruje `count` hodnot z pole `vecSource` od indexu `idxSource` do pole `vecTarget` 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).

- **Funkce pro práci s textem** (v ANSI C jsou analogické funkce v souboru `string.h`)

`string strsub(str, index, len)`
: Vrací textový podřetězec.

`long strlen(str)`
: Vrací délku stringu (počet znaků).

`long strfind(str,substr) nebo long strfind(str,substr,offset)`
: Vrací polohu začátku (kolikátý znak počítáno od 0) substringu (parametr `substr`) ve stringu `str`. Prohledávání začíná od znaku s indexem `offset` (pokud není zadán, tak od začátku). Parametr `substr` může být i znak.

`long strrfind(str,substr)`
: Stejně jako předchozí funkce, ale prohledává od konce.

`strreplace(str,pattern,substr)`
: Najde všechny výskyty stringu `pattern` ve stringu `str` a nahradí je stringem `substr`. Nový string je uložen do `str`.

`strupr(str)`
: Převede text na velká písmena.

`strlwr(str)`
: Převede text na malá písmena.

`strtrim(str)`
: Odstraní bílé znaky (mezery) na začátku a konci textu.

`long str2long(str [, default])`
: Převede text na celé číslo. Uvažuje jen tolik znaků od začátku, dokud text odpovídá číslu, zbylé znaky jsou ignorovány. Druhý (nepovinný) parametr určuje vrácené číslo pokud převod selže. Pokud parametr není uveden, při chybě funkce vrací 0.

`double str2double(str [, default])`
: Převede text na desetiné číslo. Uvažuje jen tolik znaků od začátku, dokud text odpovídá číslu, zbylé znaky jsou ignorovány. Druhý (nepovinný) parametr určuje vrácené číslo pokud převod selže. Pokud parametr není uveden, při chybě funkce vrací 0.

`string long2str(num [, radix])`
: Převede celé číslo `num` na text. Nepovinný parametr `radix` udává číselnou soustavu, ve které je převod proveden (typicky 10 nebo 16). Pokud není uveden, použije se `radix` = 10. Výstupní řetězec neobsahuje žádnou identifikaci použité číselné soustavy (např. předponu 0x u šestnáctkové soustavy).

`string double2str(num)`
: Převede desetinné číslo `num` na text.

`strcpy(dest,src)`
: Kopie řetězce. Funkce je zavedena z důvodu kompatibility s ANSI C. Lze použít konstrukci `dest=src` se stejným výsledkem.

`strcat(dest,src)`
: Spojení řetězců. Funkce je zavedena z důvodu komplatibility s ANSI C. Lze použít konstrukci `dest=dest+src` se stejným výsledkem.

`strcmp(str1,str2)`
: Porovnání stringů. Funkce je zavedena z důvodu komplatibility s ANSI C. Lze použít konstrukci `str1==str2` se stejným výsledkem.

`float2buf(buf,x[,endian])`
: Převádí číslo `x` na 4 prvky pole `buf`. Každý prvek představuje jeden byte čísla uloženého v single precision formátu dle IEEE 754 (známé jako float). Funkce se hodí pro přípravu dat při přenosu dat mezi zařízeními. Nepovinný třetí parametr má význam: 0 (default) = endian podle procesoru, 1 = nejnižší byte první, 2 = nejvyšší byte první.

`double2buf(buf,x[,endian])`
: Podobná funkce jako `float2buf`, jen ukládá 8 byte, tzn. double precision formát.

`double buf2float(buf[,endian])`
: Opačná funkce k funkci `float2buf`.

`double buf2double(buf[,endian])`
: Opačná funkce k funkci `double2buf`.

`long RegExp(str,regexp,capture[])`
: Porovnání stringu `str` s regulárním výrazem `regexp`. Pokud string vyhovuje, do pole `capture` se nastaví stringy odpovídající jednotlivým uzávorkovaným sekcím regulárního výrazu. `capture[0]` je vždy celý regulární výraz (první výskyt). Funkce vrací počet nastavených prvků v poli `capture` nebo záporné číslo v případě chyby. V regulárním výrazu jsou podporovány následující konstrukce:

  `(?i)` ... Must be at the beginning of the regular expression. Makes the matching case-insensitive.

  `^` ... Match beginning of a string

  `$` ... Match end of a string

  `()` ... Grouping and substring capturing

  `\s` ... Match whitespace

`\S` ... Match non-whitespace

`\d` ... Match decimal digit

`\n` ... Match new line character

`\r` ... Match line feed character

`\f` ... Match vertical tab character

`\v` ... Match horizontal tab character

`\t` ... Match horizontal tab character

`\b` ... Match backspace character

`+` ... Match one or more times (greedy)

`+?` ... Match one or more times (non-greedy)

`*` ... Match zero or more times (greedy)

`*?` ... Match zero or more times (non-greedy)

`?` ... Match zero or once (non-greedy)

`x|y` ... Match x or y (alternation operator)

`\meta` ... Match one of the meta characters: `^$().[]*+?|\`

`\xHH` ... Match byte with hex value 0xHH, e.g. `\x4a`.

`[...]` ...] Match any character from the set. Ranges like `[a-z` are supported.

`[^...]` ... Match any character but the ones from the set.

Příklad:

```
RegExp("48,string1,string2","^(\\d+),([^,]+),",capture);
```

Výsledek: `capture=["48,string1","48","string1"]`

`long ParseJson(json,cnt,names[],values[])`

Funkce předpokládá, že parametr `json` obsahuje text v JSON formátu. V poli `names` jsou názvy požadovaných objektů (k subpoložkám se přistupuje přes tečku, index pole se píše do `[]` - např. `"cars[1].model"`), do pole `values` funkce nastaví hodnoty těchto objektů. Parametr `cnt` udává počet požadovaných objektů (délku pole `names` i `values`). Funkce vrací počet skutečně nastavených hodnot (záporná čísla znamenají chybu).

Poznámka: Textová proměnná se deklaruje stejně jako v ANSI C, tj. `char <název proměnné>[<maximální počet znaků>];`. Pro předání stringu do funkce se používá konstrukce `char <název proměnné>[]` nebo `string <název proměnné>`.

- **Systémové** (v ANSI C nejsou)

`Archive(arc, type, id, lvl_cnt, value)`

Uloží hodnotu do archivního subsystému. `arc` je bitová maska archivů, do ktrých se má hodnota zapsat (např. pro zápis do archivů 3,5 nastavte `arc`=20 -> (BIN)10100 = (DEC)20). Archivy jsou číslovány od 1 a maximum je 15 (archiv s číslem 0 je interní systémový log). `type`

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` je unikátní identifikátor události v archivu. `lvl_cnt` je úroveň (závažnost) alarmu v případě zápisu alarmů nebo počet prvků v případě zápisu grupy. `value` je hodnota pro uložení do archivu nebo reference na pole v případě zápisu grupy.

`Trace(id,val)`

Výpis čísla `id` a hodnoty `val`. Funkce je určena pro odladění bloku. Číslo `id` je uživatelem definovaná konstanta v rozsahu 0 až 9999 pro snadnou identifikaci výpisu. Hodnota `val` může být libovolného datového typu včetně textových řetězců (string). Zprávy se vypisují do systémového logu systému REXYGEN.

Pro zobrazení výpisů v systémovém logu je potřeba je aktivovat. Jděte do menu
Target→Diagnostic messages a zaškrtněte položku *Information* v poli *Function block messages*.

Zároveň musí být povolen výpis zpráv z konkrétního bloku - zaškrtnutím checkboxu *Enable logging* na kartě *Runtime* v parametrech bloku. Ve výchozím stavu po vložení bloku z knihovny je toto povoleno. Teprve poté se zprávy objeví 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 spuštění (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 $\langle 0, 1)$. 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)`
: Přečte hodnotu ze souboru. Předpokládá binární soubor, kde jsou za sebou uloženy hodnoty typu `double` nebo textový soubor, kde na každé řádce je jedno číslo. Pořadí hodnoty v tomto souboru, kterou chceme přečíst udává parametr `idx` (počítá se od 0). Soubor je identifikován parametrem `fileid`. V současnosti jsou podporovány následující hodnoty:

`0` ...soubor na disku s názvem v parametru `p0`

`1` ...soubor na disku s názvem stejným jako název bloku rozšířený o příponu `.dat`.

`2` ...soubor na disku s názvem v parametru `srcname`, ale s příponou .dat

`3` ...soubor na disku s názvem `rexlang.dat` v aktuálním adresáři

`4-7` ...stejné jako `0-3`, ale soubor je textový. Každá řádka obsahuje jedno číslo. Číslo řádku je parametr `idx` (počítáno od 0). Hodnota `idx=-1` znamená následující řádku (lze použít pro urychlení pro sekvenční čtení více hodnot).

`void SaveValue(fileid, idx, value)`
: Uloží hodnotu `value` do souboru. Význam ostatních parametrů je stejný jako u funkce `LoadValue`.

`void GetSystemTime(time)`
: Vrací hodnotu systémového času. Obvykle je to UTC, ale závisí na nastavení operačního systému. Parametr `time` musí být pole typu `long` o nejméně 8 prvcích. Funkce naplní toto pole hodnotami (po řadě) rok, měsíc, den (v měsíci), den v týdnu, hodina, minuta, sekunda, milisekunda. Na některých platformách milisekundy nejsou k dispozici (funkce vrací vždy 0ms) nebo mají jen omezenou přesnost.

`void Sleep(seconds)`
: Pozastaví vykonávání skriptu na uvedenou dobu (zadává se v parametru jako desetinné číslo v sekundách). Funkci je nutné používat jen ve výjimečných případech, protože se tím pozastaví vykonávání celého tasku/schématu. Doba uspání by neměla přesáhnout 900 milisekund. Nejkratší doba, na kterou lze skript pozastavit, je zhruba 0.01 s. Přesná hodnota závisí na cílové platformě.

`long GetExtInt(ItemID)`
: Vrací hodnotu celočíselného vstupu/výstupu/parametru libovolného bloku v REXu určeného parametrem `ItemID`. Tento parametr je textový a má stejný význam/strukuturu jako parametr `sc` bloku `GETPI`. Pokud hodnotu nelze získat (např. neexistující `ItemID` nebo není typu `long`) blok `REXLANG` skončí chybou a je potřeba ho resetovat.

`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 řetězce/text.

`void SetExt(ItemID, value)`
: Nastaví hodnotu vstupu/výstupu/parametru libovolného bloku v REXu určeného parametrem `ItemID`. Tento parametr je textový a má stejný význam/strukuturu jako parametr `sc` bloku GETPI. 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 umožňuje procházet bloky úlohy. Pokud je `ItemID` identifikátor bloku (`cesta_k_bloku`), `subitems` bude obsahovat jména vstupů, výstupů, parametrů a vnitřních stavů bloku. Návratová hodnota funkce je počet nastavených názvů nebo záporný chybový kod. Význam `kinds` je: 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) libovolný bloku v REXu určeného parametrem `ItemID`. Tento parametr je textový a má stejný význam/strukuturu jako parametr `sc` bloku GETPI. Funkce vrací, to co vrátí volaná funkce (tj. chybový kód REXYGEN). Doporučje se spouštěný blok/subsystém pozastavit pro normální spouštění (zatržítko Halt v záložce Runtime parametrického dilaogu) a umístit do stejného tasku jako volající blok REXLANG.

`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`.

`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 `OpenMemory`.

`long memwr32(hMem, offset, value)`
: Zápis do fyzické paměti. Handle se získá funkcí `OpenMemory`.

- **Komunikační** (v ANSI C nejsou)

  Tato sada funkcí slouží pro práci se sériovou linkou (RS-232 nebo RS-485), sběrnicí I2C nebo SPI a komunikaci po TCP/IP nebo UDP/IP. Zde je uveden jen stručný popis funkcí, které se pro komunikaci používají. Součástí instalace systému REXYGEN jsou příklady, které názorně ukazují způsob použití.

  `long Open(long type, long lclIP, long lclPort, long rmtIP, long rmtPort)`
  : Otevře socket nebo COM port - podle parametru `type`. Pro TCP klient provádí rovnou connect. Vrací identifikační číslo (tzv. handle) socketu nebo portu. Pokud je záporné, otevření/spojení se nezdařilo.

  `long Open(long type, string comname, long baudrate, long parity)`
  : Modifikace příkazu `Open()` pro otevření sériové linky.

  `long Open(long type, string filename)`
  : Modifikace příkazu `Open()` pro otevření souboru.

  `long Open(long type, string localname, long locPort, string remotename, long remP`
  : Modifikace příkazu `Open()` pro otevření TCP nebo UDP socketu.

  `long OpenFile(string filename)`
  : Modifikace příkazu `Open()` pro otevření souboru.

  `long OpenCom(string comname, long baudrate, long parity)`
  : Modifikace příkazu `Open()` pro otevření sériové linky. Nastavení parity: 0=žádná, 1=lichá, 2=sudá.

  `long OpenUDP(string localname, long lclPort, string remotename, long remPort)`
  : Modifikace příkazu `Open()` pro otevření UDP socketu.

  `long OpenTCPsvr(string localname, long lclPort)`
  : Modifikace příkazu `Open()` pro otevření TCP socketu - server, naslouchání.

  `long OpenTCPcli(string remotename, long remPort)`
  : Modifikace příkazu `Open()` pro otevření TCP socketu - klient. POZOR funkce nečeká na navázání spojení To nějakou dobu trvá (na lokální síti jednotky milisekund) a pokud se zavolá `Write()` nebo `Read()` ještě před navázáním spojení vrací funkce chybu -307 (chyba otevření souboru). Je potřeba buď chvíli počkat (např. použít funkci `Sleep()`) nebo zápis/čtení dělat až v dalším tiku.

  `long OpenI2C(string devicename)`
  : Modifikace příkazu `Open()` pro otevření I2C zařízení.

  `long OpenSPI(string devicename)`
  : Modifikace příkazu `Open()` pro otevření SPI zařízení.

  `long OpenDevice(string devicename)`
  : Modifikace příkazu `Open()` pro otevření zařízení. V podstatě stejné jako OpenFile(), ale následné `Read()` a `Write()` je nečekací (tj. vrací -1, pokud není co číst nebo nelze zapsat).

  `long OpenMemory(string devicename, long baseaddr, long size)`
  : Modifikace příkazu `Open()` pro mapování fyzické paměti.

`long OpenSHM(string devicename, long deviceid, long size, long flags)`
: Modifikace příkazu `Open()` pro mapování sdílené paměti (jen na linuxu, volá ftok() a shmget()). První a druhý parametr slouží pro identifikaci oblasti paměti (tj. musí být stejné u všech spolupracujících zařízení); `size` je velikost sdílené oblasti paměti v bajtech; `flags` jsou standartní linuxové flagy/práva (pokud je 0=default, tak se nastaví: vytvořit oblast, pokud neexistuje, všichni mohou číst i zapisovat)

`void Close(long handle)`
: Zavře socket, sériovou linku, soubor nebo jiné zařízení otevřené pomocí funkce `Open` (nebo její varianty).

`void GetOptions(long handle, long params[])`
: Přečte parametry - nastaví aktuální hodnoty do pole `params`; pole musí být dostatečně dlouhé - viz SetOptions

`void SetOptions(long handle, long params[])`
: Nastaví parametry sériové linky nebo socketu. Pole musí být dostatečně dlouhé - aktuálně je

  - 22 parametrů pro sériovou linku (na Windows parametery for SetCommState() a SetCommTimeouts() v pořadí: BaudRate, fParity, Parity, StopBits, ByteSize, fDtrControl, fRtsControl, fAbortOnError, fBinary, fErrorChar, fNull, fDsrSensitivity, fInX, fOutX, fOutxCtsFlow, fOutxDsrFlow, fTXContinueOnXoff, ReadIntervalTimeout, ReadTotalTimeoutConstant, ReadTotalTimeoutMultiplier, WriteTotalTimeoutConstant, WriteTotalTimeoutMultiplier; na linuxu je jiný interface, ale význam parametrů je stejný, pokud lze implementovat)
  - 2 pro soubor (1. prvek je režim: 1=seek begin, 2=seek current, 3=seek end, 4=set file end, 2. prvek je offset pro seek),
  - 3 pro SPI (1. prvek je SPI mode, 2. prvek je bits per word, 3. prvek je max speed Hz),
  - 5 pro I2C (1. prvek je slave address, 2. prvek je 10bits address flag, 3. prvek je Packet Error Checking flag, 4. prvek je nuber of retries, 5. prvek je timeout)

`long Accept(long hListen)`
: Přijme spojení navázané klientem na naslouchací socket `hListen`; vrací handle komunikačního socketu nebo chybu.

`long Read(long handle, long buffer[], long count)`
: Přijme data z linky nebo socketu nebo přečte ze souboru ; v parametru `count` je maximální počet byte, které se mají přečíst; funkce vrací počet skutečně přečtených byte nebo chybový kód; data jsou čtena tak, ze jeden byte z linky odpovídá jednomu prvku typu `long` v poli `buffer`. Ve starších verzích se funkce jmenovala `Recv`, což lze z důvodu zpětné kompatibility stále použít. Funkci lze použít také ve tvaru `long Read(long handle, string data[], long count)` (tj. místo pole na data se použije string;

jeden byte ve vstupním souboru odpovídá jednomu znaku; binární soubory takto číst nelze) Chybové kódy jsou:

- `-1` je třeba počkat na dokončení operace (funkce je totiž tzv. „neblokující“
- `-309` čtení selhalo; chybový kód operačního systému se objevuje v Funkce logu (pokud je zapnuto logování u bloku)
- `-307` soubor/socket není otevřen

může mít ještě jeden (poslední, nepovinný) parametr `offset`, což lze použít pokud je handle vytvořen OpenSHM() nebo OpenMemory().

`long Write(long handle, long buffer[], long count)`
: Odešle data na linku nebo socket; v parametru `count` je počet byte, které se mají poslat; funkce vrací počet skutečně vyslaných byte nebo chybový kód; data jsou zapisována tak, že jeden byte z linky se odpovídá jednomu prvku typu `long` v poli `buffer`. Ve starších verzích se funkce jmenovala `Send`, což lze z důvodu zpětné kompatibility stále použít. Funkci lze použít také ve tvaru `long Write(long handle, string data)` (tj. místo pole dat se použije string; jeden byte ve výstupním souboru odpovídá jednomu znaku; binární soubory takto zapisovat nelze) Chybové kody jsou:
  - `-1` je třeba počkat na dokončení operace (funkce je totiž tzv. „neblokující“
  - `-310` zápis selhal; chybový kód operačního systému se objevuje v logu (pokud je zap
  - `-307` soubor/socket není otevřen

  Funkce může mít ještě jeden (poslední, nepovinný) parametr `offset`, což lze použít pokud je handle vytvořen OpenSHM() nebo OpenMemory().

`long ReadLine(long handle, string data)`
: Přečte jednu řádku z (textového) souboru sériové linky nebo socketu; přečtené znaky jsou v proměnné `data` až do velikosti stringu; vrací skutečnou délku řádky 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)`
: Vrací 1 pokud soubor nebo zařízení existuje (lze jej otevřít pro čtení).

`long I2C(long handle, long addr, long bufW[], long cntW, long bufR[], long cntR)`
: Příkaz pro komunikaci po sběrnici I2C. Funguje jen na zařízeních s operačním systémem Linux, která mají toto rozhraní (např. Raspberry Pi). Provádí současně odeslání i příjem dat na/ze slave zařízení s adresou `addr`. Handle se získá funkcí `OpenI2C`, kde parametr funkce je jméno zařízení (dle operačního systému). Parametr `bufW` je buffer (pole) pro odchozí data, `cntW` je počet odeslaných byte, `bufR` je buffer (pole) pro příchozí data a `cntR` je počet přijímaných byte. Funkce vrací 0 nebo chybový kód.

`long SPI(long handle, 0, long bufW[], long cntW, long bufR[], long cntR)`
: Příkaz pro provedení transakce na sběrnici SPI. Funguje jen na zařízeních s operačním systémem Linux, která mají toto rozhraní (např. Raspberry Pi). Handle se získá funkcí `OpenSPI`, kde parametr funkce je jméno zařízení (dle operačního systému). Druhý parametr je vždy 0 (rezervován pro interní použití). Parametr `bufW` je buffer (pole) pro odchozí data, `cntW` je počet odeslaných byte, `bufR` je buffer (pole) pro příchozí data a `cntR` je počet přijímaných byte. Pamatujte, že SPI komunikace probíhá současně v obou směrech (full-duplex), takže výsledná délka SPI transakce je dána maximem parametrů `cntW` a `cntR`, nikoliv jejich součtem. Funkce vrací 0 nebo chybový kód.

`long Seek(long handle, long mode[], long offset)`
: Příkaz pro nastavení čtecí/zapisovací pozice. Parametr mode má význam: 1=offset od začátku souboru, 2= offset od aktualní pozice, 3=offset od konce souboru.

`long Recv(long handle, long buffer[], long count)`
: *Pouze pro zajištění zpětné kompatibility. Funkce nahrazena funkcí* ***`Read`***.

`long Send(long handle, long buffer[], long count)`
: *Pouze pro zajištění zpětné kompatibility. Funkce nahrazena funkcí* ***`Write`***.

`long crc16(data,length,init,poly,flags,offset)`
: Vypočte kontrolní součet, tak jak jej definují různé komunikační protokoly. `data` pole (long, ale v každém prvku se předpoklád jeden byte) nebo text nad kterým se dělá kontrolní součet. `length` počet platných byte v poly/textu `data` (pro text je možné zadat -1 a pak se uvažuje celý text). `init` počáteční hodnota kontrolního součtu (tzv. inicializační vektor) `poly` tzv. řídící polynom `flags` 1...obrací se pořadí bitů (ve vstupních byte i výsledném crc), 2...na výsledném crc se udělá ještě xor 0xFFFF, 4...zpracovavají se všechny 4 bajty z longu v poli dat (delka i offset je v bajtech), 8... jako 4, ale data v longu se čtou od nejvyššího bajtu `offset` index prvního zpracovávaného bajtu z dat (tj. obvykle 0, ale někdy je potřeba pár bajtů na začátku vynechat, tak sem se napíše kolik) Poznámka: pro jiné délky kodu existuje analogická funkce, např. pro 32-bitové crc použijeme `long crc32(data,length,init,poly,flags)`, pro 8-bitové `long crc8(data,length,init,poly,flags)`, atd. Příklady volání pro časté protokoly: MODBUS: crc16("123456789",-1,0xFFFF,0x8005,1)); DECT-X: crc16("123456789",-1,0,0x0589,0));

Ladění kódu, debugging

Pro ladění kódu je k dispozici příkaz `Trace`, viz výše. Dále lze použít výstupy bloku, které se nouužívají 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 růyné mezivýsledky pomocí funkce `SetExt`.

Poznámky

- Typ vstupů `u0..u15`, výstupů `y0..y15` a parametrů `p0..p15` se určuje až při překladu zdrojového souboru bloku, podle specifikací `input`, `output` a `parameter`.
- Všechny chybové kódy `< -99` vyžadují reset bloku `REXLANG` vstupem `RESET`. Je samozřejmě nutné napřed odstranit příčinu, která chybu způsobila.
- 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.
- Parametr `srcname` je možné udávat s celou cestou. V opačném případě se soubor hledá na aktuálním adresáři a adresářích specifikovaných volbou -I v parametrech příkazové řádky programu REXYGEN Compiler.
- Všechny parametry vektorových funkcí jsou typu `double` (popřípadě vektor typu `double`) kromě parametru `n`, který je typu `long`. Také si všimněme, že funkce, které mají jen jeden vektorový parametr existují ve třech variantách:

  `double funkce(val1,...,valn)`
  : Vektor se zadává jako posloupnost parametrů typu `double`.

  `double funkce(n,val1,...,valn)`
  : Vektor se zadává jako v předchozím případě, ale navíc první parametr udává počet čísel – délku vektoru. Na rozdíl od předchozí varianty, lze v této variantě překládat zdrojový kód bez úprav překladačem jazyka C. Parametr `n` musí být přímo číslo (nikoliv tzv. `const` proměnná) a musí odpovídat počtu následujících parametrů tvořících vektor.

  `double funkce(n,vec)`
  : Parametr `n` je libovolný výraz typu `long` a udává počet prvků vektoru, se kterými funkce počítá.

- Nepovinný parametr `n` u vektorových funkcí se musí uvádět, pokud chceme stejný kód beze změn použít v překladači C/C++. Takové použití vyžaduje implementovat všechny nestandardní funkce, což není velký problém, ale funkce s variantním počtem parametrů musí nějak poznat jejich počet.
- Ve všech případech je třeba mít na paměti, že všechny vektory začínají prvkem s indexem 0 a dále, že program (stejně jako jazyk C) nekontroluje meze polí. Např. při definování `double vec[10], x;` (vektor s deseti prvky s indexy `0` až `9`) není zápis `x=vec[10];` ani syntaktická ani runtime chyba, ale hodnota je nedefinovaná. Dokonce lze i napsat `vec[11]=x;`, což je obzvláště nebezpečné, protože se tím přepíše nějaká jiná proměnná a program nefunguje správně, případně se může i „zaseknout“.

- Při překladu skriptu se často hlásí jen chyba `syntax error` a číslo řádky, kde nastává. Znamená to chybu v syntaxi. Pokud se zdá vše v pořádku, může to být tím, že použitý identifikátor je klíčové slovo jazyka nebo jméno vestavěné funkce.

- Všechny skoky se překládají relativně, tj. příslušný kus kódu je omezen na 32767 instrukcí (v přenositelném formátu na různé platformy).

- Do zásobníku se ukládají všechny aktuálně platné proměnné a mezivýsledky, tj.:

  - Globální proměnné a lokální `static` proměnné (trvale na začátku zásobníku)
  - Návratové adresy funkcí
  - Parametry předávané funkcím (včetně „vestavěných“)
  - Lokální proměnné funkcí
  - Návratová hodnota funkce
  - Mezivýsledky operací (například výraz `a=b+c;` se provádí tak, že se do zásobníku uloží hodnota `b`, pak (na další místo) hodnota `c` pak se provede součet, obě hodnoty se ze zásobníku zruší a vloží se tam hodnota součtu).

  Každá jednoduchá proměnná (tedy `long` i `double` se počítá za jednu položku v zásobníku. Pole se tedy počítá podle jeho délky opět bez ohledu na typ.

- Pole se do funkcí předávají odkazem. To znamená, že v zásobníku se počítá parametr jako jedna hodnota a hlavně se nepracuje s lokální kopií, ale vždy přímo s předaným polem.

- Pokud je zadána nedostatečná velikost zásobníku (méně než potřeba pro globální proměnné plus 10), volí se automaticky jako dvojnásobek potřeby pro globální proměnné (tj. předpoklad, že aktivních lokálních proměnných nebude více než globálních) plus 100 rezerva (na výpočty, parametry funkcí, lokální proměnné, pokud by globálních bylo málo).

- Při základním debug módu se kontroluje (za běhu skriptu), zda jsou všechny čtené hodnoty inicializované, zda index pole nepřesahuje deklarovanou velikost pole, přidává se několik neinicializovaných hodnot před a za každé deklarované pole (další ochrana proti nesprávnému indexu v poli), do kódu se přidávají instrukce `NOP` s argumentem číslo řádku ve zdrojovém souboru (usnadňuje dohledání v *.ill souboru). Pokud je zvolen úplný debug mód tak se navíc kontroluje, zda se program nepokouší přistupovat mimo platnou oblast dat (což je zatím nad `SP` ve stacku – neplatné hodnoty v zásobníku).

- Pod pojmem instrukce se u tohoto bloku nemyslí instrukce procesoru, ale instrukce mezikódu nezávislého na procesoru. Zdrojový kód přeložený do tohoto mezikódu je v souboru *.ill (mnemokódy pro jednotlivé instrukce, co řádka to jedna instrukce).

- Při použití sériové linky funkce `Open()` vždy nastaví binární neblokující režim bez timeoutů, 8 datových bitů, jeden stopbit, bez parity, 19200Bd. Bitovou rychlost a paritu lze nastavit přímo ve funkci `Open()` pomocí nepovinného druhého (bitrate) a třetího (parita) parametru.
- Při čtení a zápisu dat do textového souboru je potřeba počítat s tím, že se musí přečíst/zapsat celý při každém přístupu. Naproti tomu binární soubor má pevnou strukturu, takže přístup je rychlejší. Výhoda textových souborů spočívá v tom, že je lze zobrazovat i měnit bez speciálního programu.
- Na rozdíl od standardních bloků systému REXYGEN se automaticky nevolá funkce `parchange()` v inicializační fázi. Pokud je to potřeba, je nutné ji explicitně zavolat ve funkci `init()`.
- Protože operační systémy na bázi windows i linuxu přistupují k sériové lince stejně jako k souboru, je možné pomocí funkcí `Recv()` a `Send()` číst a zapisovat soubory sekvenčně po bajtech. V tomto případě se ve funkci `Open()` jako parametr `type` používají stejné hodnoty, jako ve funkci `LoadValue()` (parametr `fileid`).
- Ve funkci `WriteRead()` pro případ SPI je potřeba počítat s tím, že se data čtou i během zápisu. Pokud tedy potřebujeme zapsat do zařízení 2byte (např. číslo příkazu) a po jeho předání zařízení posílá 4byte dat, je potřeba číst 6byte, přičemž první a druhý byte (přijatý při zápisu čísla příkazu) neobsahuje platná data. Obecně nelze ale byte přečtené při zápisu vypustit, protože u některých zařízení obsahují platná data.
- Funkce `OpenUDP()`, `OpenTCPsvr()`, `OpenTCPcli()`, podporují i IPv6 socket. Zda použít IPv4 nebo IPv6 se určí automaticky podle formátu adresy popřípadě podle toho co vrátí DNS.
- 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.

Vstupy

`HLD`	Pozastavení – kód bloku se nevykonává, je-li hodnota rovna `on`.	`Bool`
`RESET`	Resetování chyby při náběžné hraně; blok se znovu inicializuje (vynulují se všechny globální proměnné a zavolá se funkce `Init()`)	`Bool`
`u0..u15`	Vstupní signály, jejichž hodnoty jsou přístupné ve skriptu	`Any`

Výstupy

`iE`	Kód chyby při běhu skriptu. Pro chybové `< -99` je provádění algoritmu bloku zastaveno do reinicializace vstupem `RESET` nebo novým spuštěním exekutivy) `0` ..... vše v pořádku, proběhla celá funkce `main()` popř. `init()` `-1` .... provádění programu skončilo příkazem `Suspend()`, tj. vypršel čas pro výpočet; výpočet bude při dalším spuštění bloku pokračovat tam, kde skončil `< -1` .. chybový kód `xxx` systému REXYGEN, více viz příloha C `> 0` ... uživatelské návratové hodnoty, běh algoritmu beze změny	`Error`
`y0..y15`	Výstupní signály, jejichž hodnoty jsou definovány ve skriptu	`Any`

Parametry

`srcname`	Jméno souboru se skriptem	⊙`srcfile.c`	`String`
`srctype`	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)`
`stack`	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)`

`debug`	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). ⊙3 `1` ..... bez kontroly `2` ..... základní kontrola `3` ..... úplná kontrola	`Long (I32)`
`strs`	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)`
`p0..p15`	Parametry, jejichž hodnoty jsou přístupné ze skriptu	`Any`

Příklad

Následující příklad implementuje lineární model procesu definovaný vahovou funkcí (filtr typu FIR) doplněný o saturaci na vstupu. Příklad je napsán tak, aby ukazoval různé konstrukce použitého skriptovacího jazyka. Algoritmus by bylo možné realizovat i jednodušším postupem.

```
double input(0) vstup;  //promenna 'vstup' predstavuje hodnotu u0
double output(0) vystup; //promenna 'vystup' predstavuje prirazeni do y0
double stav[20], param[20];
const long count=20;

long init(void)
{
    long i;
    const double a=0.95;
    param[0]=0.2;param[5]=0.2;param[10]=0.2;param[12]=0.2;param[15]=0.2;
    for(i=0;i<count;i++)
    {
        param[i]=param[i]+exp(-i*a)/a;
        Trace(1,param[i]);
    }
    return 0;
}

long main(void)
{
    long i;
    double soucet=0.0;
    for(i=0;i<count-1;i++)
        stav[i]=stav[i+1];
```

```
    if(fabs(vstup)>1)
        stav[count-1]=(vstup>0)? 1 : -1;
    else
        stav[count-1]=vstup;
    for(i=0;i<count;i++)
    {
        soucet+=stav[i]*param[count-1-i];
        Suspend(0.1);
    }
    vystup=soucet;
    return 0;
}

long exit(void){return 0;}
```

a tentýž příklad v STL syntaxi:

```
VAR_INPUT
  vstup:REAL;  //promenna 'vstup' predstavuje hodnotu u0
END_VAR

VAR_OUTPUT
  vystup:REAL; //promenna 'vystup' predstavuje prirazeni do y0
END_VAR

VAR_PARAMETER
  tt:REAL; //promenna 'tt' predstavuje hodnotu parametru p0
END_VAR

VAR
  param, stav : ARRAY[0 .. 19] OF REAL;
END_VAR

VAR CONSTANT
  count:INT := 20;
END_VAR

FUNCTION init : INT;
VAR
  i:INT;
END_VAR

VAR CONSTANT
    a:REAL := 0.95;
```

```
END_VAR
    param[0]:=0.2; param[5]:=0.2; param[10]:=0.2; param[12]:=0.2; param[15]:=0.2;
    FOR i:=0 TO count-1 DO
        param[i] := param[i] + EXP(-i*a)/a;
        Trace(1,param[i]);
    END_FOR

    init := 0;
END_FUNCTION


FUNCTION main : INT;
VAR
  i:INT;
  soucet:REAL := 0.0;
END_VAR

    FOR i:=0 TO count-2 DO
        stav[i] := stav[i+1];
    END_FOR

    IF abs(vstup)>1 THEN
IF vstup>0.0 THEN
            stav[count-1] := 1;
ELSE
    stav[count-1] := -1;
END_IF
    ELSE
        stav[count-1] := vstup;
    END_IF

    FOR i:=0 TO count-1 DO
        soucet := soucet+stav[i]*param[count-1-i];
     IF tt>0.0 THEN
           Suspend(tt);
        ELSE
           Suspend(0.1);
END_IF
    END_FOR

    vystup := soucet;
    main := 0;
END_FUNCTION
```

```
FUNCTION exit : INT;
    exit := 0;
END_FUNCTION
```

Kapitola 17

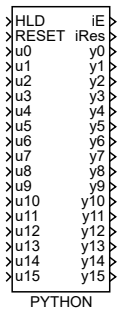
LANG – Speciální bloky

Obsah

PYTHON – Volně programovatelný blok v jazyce Python

Symbol bloku

Licence: REXLANG



Popis funkce

V některých případech se může stát, že je do řídícího 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 interpretr umožňuje načítání modulů/skriptů z různých lokací na cílovém zařízení. Blok `PYTHON` se může odkazovat na jakýkoliv skript dostupný standardnímu interpretru 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 interpretru. 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. Vrácena 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 objet 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 systém pomocí absolutních cest.

Vstupy

`HLD`	Hold – Algoritmus bloku není spuštěn při nastavení vstupu na `on`.	`Bool`
`RESET`	Náběžná hrana resetuje blok. Blok je opět inicializován, takže všechny globální proměnné jsou uvolněny a je zavolána funkce `init()`.	`Bool`
`u0..u15`	Vstupní signály, které jsou přístupné ve skriptu.	`Any`

Výstupy

`iE`	Chybový kód. `0` ..... Žádná chyba nenastala, celá funkce `main()` byla úspěšně spuštěna. (včetně funkce `init()`). `xxx` ... Chybový kód systému REXYGEN, viz Appendix C	`Error`
`iRes`	Výstupní kód exekuce bloku.	`Long (I32)`
`y0..y15`	Výstupní signály, které jsou dostupné ve skriptu.	`Any`

Parametry

`srcname`	Jméno zdrojového souboru	⊙`program.py`	`String`
`embedded`	Vestavění skriptu	⊙`on`	`Bool`
`p0..p15`	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 přidanou 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-pythonblkt
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/). 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

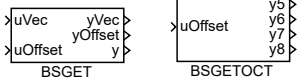
DSP – Zpracování speciálních signálů

Obsah

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

`uVec`	Struktura s daty (pole bajtů)	`Reference`
`uOffset`	Posunutí (v bajtech) dat od začátku struktury	`Long (I32)`

Výstupy

`yVec`	Kopie `uVec` pro řetězení	`Reference`
`y`	Požadovaná hodnota ze struktury (typ hodnoty je definován parametrem)	`Any`
`yOffset`	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	`Long (I32)`

Parametry

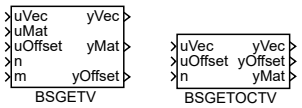
`m`	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	`Long (I32)`

`BE`	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		`Bool`
`type`	Datový typ hodnoty	↓2 ↑10 ⊙2	`Long (I32)`

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

`uVec`	Struktura s daty (pole bajtů)	`Reference`
`uMat`	Matice pro uložení hodnot	`Reference`
`uOffset`	Posunutí (v bajtech) dat od začátku struktury	`Long (I32)`
`n`	Počet sloupců výstupní matice	`Long (I32)`
`m`	Počet řádek výstupní matice	`Long (I32)`

Výstupy

`yVec`	Kopie `uVec` pro řetězení	`Reference`
`yMat`	Kopie `uMat` pro řetězení	`Reference`
`yOffset`	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	`Long (I32)`

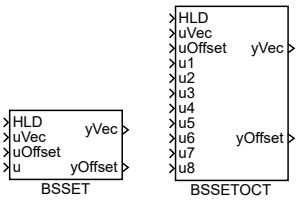
Parametry

`m`	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	`Long (I32)`
`BE`	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		`Bool`
`nmax`	Alokovaná velikost výstupní matice (celkový počet prvků) `yMat`	↓1 ⊙32	`Long (I32)`
`type`	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

`HLD`	Potlačení zápisu	`Bool`
`uVec`	Struktura s daty (pole bajtů)	`Reference`
`uOffset`	Posunutí (v bajtech) dat od začátku struktury	`Long (I32)`
`u`	Zapisovaná hodnota do struktury (typ hodnoty definován parametrem)	`Any`

Výstupy

`yVec`	Kopie `uVec` pro řetězení	`Reference`
`yOffset`	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	`Long (I32)`

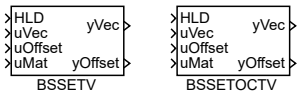
Parametry

`m`	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	`Long (I32)`
`BE`	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		`Bool`
`type`	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

`HLD`	Potlačení zápisu	`Bool`
`uVec`	Struktura pro vyplnění daty z matice (pole bajtů)	`Reference`
`uOffset`	Posunutí (v bajtech) dat od začátku struktury	`Long (I32)`
`uMat`	Matice s daty	`Reference`

Výstupy

`yVec`	Kopie `uVec` pro řetězení	`Reference`
`yOffset`	Posunutí (v bajtech) dat od začátku struktury další hodnoty (pro řetězení)	`Long (I32)`

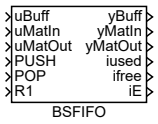
Parametry

`m`	Počet použitých hodnot (u vícenásobných bloků)	↓1 ↑8 ⊙8	`Long (I32)`
`BE`	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		`Bool`
`type`	Datový typ hodnoty	↓2 ↑10 ⊙2	`Long (I32)`

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

`uBuff`	Binární struktura s daty (pole bajtů)	`Reference`
`uMatIn`	Reference matice se vstupními daty (pro PUSH)	`Reference`
`uMatOut`	Reference matice pro uložení výstupních dat (pro POP)	`Reference`
`PUSH`	Povolení vkládání do bufferu	`Bool`
`POP`	Povolení vyčítání z bufferu	`Bool`
`R1`	Reset bufferu (znovu načtení hlaviček z `uBuff`)	`Bool`

Parametry

`OW`	Režim přepisu nejstarších položek	`Bool`
`REV`	Vyčítání posledních vložených položek nejdříve	`Bool`

`col`	Počet vyčítaných sloupců (pro POP)	⊙1	`Long (I32)`

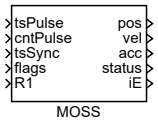
Výstupy

`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`

MOSS – Přesný senzor pohybu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok implementuje pokročilý filtr pro inkrementální (kvadraturní) senzory polohy. Blok vyžaqduje speciální hardware, neboť pro správnou činnost je nutné znát nejen aktuální hodnotu z čidla, ale také časovou značku posledního puslu, 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 vložením 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

`tsPulse`	Časová značka posledního pulsu	`DWord (U32)`
`cntPulse`	Poslední puls (stav čítače)	`DWord (U32)`
`tsSync`	Časová značka synchronizačního pulsu. Čas synchonizačního pulzu je okamžik, ke kterému je určován výstup (obvykle okamžik spuštění tasku systému REXYGEN)	`DWord (U32)`
`flags`	Vstupní příznaky (1: POS, 2: NEG, 4: RUN)	`DWord (U32)`
`R1`		`DWord (U32)`

Parametry

`freq`	Frekvence zdroje časové značky [Hz]	↓`0.0` ⊙`100000000.0`	`Double (F64)`

`stall`	Čas detekce zastavení [s]. Pokud nepřijde pulz více než `stall` sekund, je to vyhodnoceno jako zastavení (tj. výstupní rychlost i zrychlení je 0) ↓`0.0` ⊙`0.08`	`Double (F64)`
`alpha`	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)`
`maxpos`	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 `maxpos` 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)`
`mindivert`	Minimální doba pro omezení odchylky [s]. Když puls nepřijde dlouhou dobu, výstupní poloha "driftuje". Aby se tomu zamezilo, tak pokud pulz nepřijde déle než `mindivert` 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

`pos`	Filtrovaná poloha	`Double (F64)`
`vel`	Filtrovaná rychlost	`Double (F64)`
`acc`	Filtrované zrychlení	`Double (F64)`
`status`	Výstupní stavové příznaky (1: POS, 2: NEG, 4: RUN, 8: INIT, 16: PULSE, 32: STALLED, 64: DIVERT)	`Long (I32)`
`iE`		`Error`

Kapitola 19

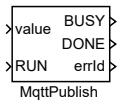
MQTT – Komunikace přes MQTT protokol

Obsah

`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 [7].

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 [8] (pouze anglicky).

Pokud je parametr `RETAIN` nahozen, odchozí pakety zprávy budou označeny příznakem RETAIN. Více informací naleznete ve specifikaci MQTT protokolu [8] (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`.

Vstupy

`value`	Vstupní signál	`String`
`RUN`	Povolení běhu algoritmu	`Bool`

Parametry

`topic`	MQTT topic		`String`
`QoS`	Quality of Service `1` ..... QoS0 (Maximálně jednou) `2` ..... QoS1 (Alespoň jednou) `3` ..... QoS2 (Právě jednou)	⊙`1`	`Long (I32)`
`RETAIN`	Zachovat poslední zprávu	⊙`on`	`Bool`
`defBuffSize`	Výchozí velikost bufferu	↓`1` ⊙`2048`	`Long (I32)`

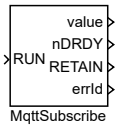
Výstupy

`BUSY`	Příznak probíhající operace	`Bool`
`DONE`	Příznak dokončení transakce	`Bool`
`errId`	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 [7].

Účelem bloku `MqttSubscribe` je přihlášení se k odebírání zpráv ze zařízení typu *MQTT Broker* a 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 [8] (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 [8] (pouze anglicky)..

Přihlašování se k odběrům s využitím zástupných znaků typu *wildcards* není podporováno.

Vstup

`RUN`	Povolení běhu algoritmu		`Bool`

Parametry

`topic`	MQTT topic		`String`
`QoS`	Quality of Service `1` ..... QoS0 (Maximálně jednou) `2` ..... QoS1 (Alespoň jednou) `3` ..... QoS2 (Právě jednou)	⊙`1`	`Long (I32)`
`type`	Očekávaný typ příchozích dat `1` ..... string `2` ..... double `3` ..... long `4` ..... bool `5` ..... byte vector/blob	⊙`1`	`Long (I32)`
`mode`	Způsob bufferování příchozích zpráv `1` ..... poslední hodnota `2` ..... bufferovat hodnoty	⊙`1`	`Long (I32)`
`defBuffSize`	Výchozí velikost bufferu	↓`1` ⊙`2048`	`Long (I32)`

Výstupy

`value`	Výstupní signál		`Any`
`nDRDY`	Počet přijatých zpráv	↓`0` ↑`10`	`Long (I32)`
`errId`	Kód chyby		`Error`

Kapitola 20

MC_SINGLE – Řízení pohybu v jedné ose

Obsah

Tato kategorie bloků zahrnuje bloky pro řízení jedné osy (jednoho motoru), tak jak jsou definovány ve specifikaci PLCopen. Proto zde nejsou dodrženy konvence pro pojmenovávání zavedené v systému REXYGEN. Protože PLCopen používá pro všechny bloky v názvu prefix `MC_`, je tento princip dodržen i zde. Pro řízení pohybu je vhodné, spíše však nutné, bloky dle PLCopen doplnit o další bloky (jedná se o vše, co je v normě označeno jako *vendor specific*, a různá rozšíření). Takové bloky mají v názvu prefix `RM_`. Dále je potřeba si uvědomit, že PLCopen (respektive norma IEC 61131-3, ze které vychází) nerozlišuje vstupy od parametrů (vše na značce bloku vypadá jako vstup). Řídicí systém REXYGEN však parametry a vstupy zpracovává odlišně, zejména z důvodu lepší přehlednosti. Proto skoro všechny bloky existují ve dvojím provedení: s prefixem `MC_`, které jsou plně kompatibilní s PLCopen a mají tedy parametry na vstupech, a bloky s prefixem `MCP_`, které mají své vnitřní parametry jako ostatní bloky řídicího systému REXYGEN. Parametry, které bloky používají, ale nejsou definovány v PLCopen (tzv. *vendor specific* parametry), jsou i u bloků s prefixem `MC_` zadávány jako parametry a výše popsaným změnám nepodléhají.

Specifikace PLCopen říká, že pro činnost bloku je důležitá hodnota parametrů/vstupů při náběžné hraně na vstupu `Execute`. Předtím a potom se mohou parametry/vstupy libovolně měnit a na funkci bloku to nemá vliv. V systému REXYGEN se předpokládá, že parametry se mění jen občas a proto parametry (na rozdíl od vstupů) nejsou „zapamatovány“ při náběžné hraně na vstupu `Execute`. Je proto potřeba dodržovat pravidlo, že parametry bloku se nesmí měnit, během jeho činnosti, tj. pokud je výstup `Busy` ve stavu zapnuto.

Popis bloků v této příručce je dostatečný pro jejich použití, nicméně nejsou zde vysvětleny hlubší souvislosti a motivace. Proto je doporučeno před použitím bloků pro řízení pohybu prostudovat i specifikaci PLCopen.

PLCopen definuje některé signály jako vstupně-výstupní. Je to zejména odkaz na osu (signál `Axis`) a odkazy na další, obvykle *vendor specific*, struktury. Řídicí systém REXYGEN vstupně-výstupní signály nepodporuje. Proto všechny bloky používají místo signálu

`Axis` vstup `uAxis` a výstup `yAxis`. Blok na výstup `yAxis` vždy kopíruje hodnotu (odkaz na strukturu) vstupu `uAxis`. Pro správnou funkci je nutné, aby vstup `uAxis` byl spojen s výstupem `axisRef` příslušného bloku `RM_Axis`, a to buď přímo nebo prostřednictvím výstupu `yAxis` jiného bloku. Výstup `yAxis` není pro bloky důležitý, ale použitá koncepce umožňuje bloky řetězit a tím určit pořadí jejich vykonávání. Ostatní vstupně-výstupní signály (odkazy) používají stejný princip, popřípadě jsou definovány pouze jako vstupní.

`PLCopen` definuje výstupy `Busy`, `Active`, `CommandAborted` jako nepovinné téměř u všech bloků. V řídicím systému REXYGEN jsou tyto signály zavedeny stejným způsobem, ačkoliv v současné implementaci se u některých bloků nepoužívají. Tento způsob byl zvolen z důvodu snadné budoucí rozšiřitelnosti.

Jednotky použité pro polohu a vzdálenost si uživatel může zvolit libovolně. Mohou to být metry, milimetry, pulzy z čidla polohy, úhlové stupně (v případě rotační osy) nebo cokoliv jiného. Je však nutné zvolenou jednotku používat ve všech blocích připojených k příslušné ose a dále je potřeba si uvědomit, že rychlost se ve všech blocích musí zadávat ve zvolených jednotkách polohy za sekundu a analogicky zrychlení a jerk. Časové údaje jsou vždy v sekundách.

Řídicí systém REXYGEN používá více vláken pro spouštění bloků. Normálně se o to uživatel nemusí starat, nicméně použití odkazů (vstupy typu `Reference`) může narušit synchronizační mechanismus. Aby toto nenastalo, je nutné mít blok `RM_Axis` a blok, který je k němu připojen vstupem typu `Reference`, ve stejné úloze, tj. všechny bloky příslušející k jedné ose musí být v jedné úloze řídicího systému REXYGEN (viz bloky `EXEC`, `TASK`, `IOTASK` apod.). Některé bloky již mají synchronizaci ošetřenou, u těch je potřeba se řídit pokyny v této příručce. Momentálně se však jedná pouze o blok `RM_AxisSpline`).

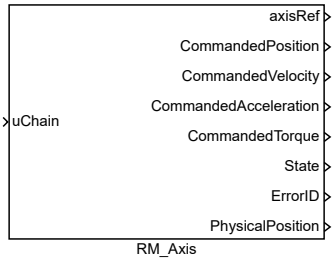
Některé vstupy a parametry jsou typu výběr ze seznamu. V takovém případě jsou v seznamu vždy všechny hodnoty pro daný typ signálu. Pro některé bloky však určité hodnoty nedávají smysl (například blok `MC_MoveVelocity` nepodporuje `Direction = shortest_way`). Platné hodnoty jsou uvedeny v této příručce v popisu jednotlivých bloků.

U všech pohybových bloků platí pravidlo, že pokud parametr určující maximální rychlost, zrychlení, zpomalení nebo jerk není zadán, respektive má hodnotu 0, použije se hodnota nastavená v připojeném bloku `RM_Axis`.

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 přepočet 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 paramtry 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	Long (I32)

Výstupy

`axisRef`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`CommandedPosition`	Požadovaná poloha osy Poloha, která se zadává do pohybových bloků. V případě homingu nebo cyklické osy se liší od `PhysicalPosition`	`Double (F64)`
`CommandedVelocity`	Požadovaná rychlost osy	`Double (F64)`
`CommandedAcceleration`	Požadované zrychlení osy	`Double (F64)`
`CommandedTorque`	Požadovaný moment/síla	`Double (F64)`
`State`	Stav osy pro řízení pohybu `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)	`Long (I32)`
`ErrorID`	Kod chyby ve stavu errorstop `i` ..... obecná chyba systému REXYGEN	`Error`
`PhysicalPosition`	Požadovaná poloha osy Poloha, která se zapojuje do zpětnovazebního regulátoru. V případě homingu nebo cyklické osy se liší od `CommandedPosition`	`Double (F64)`

Parametry

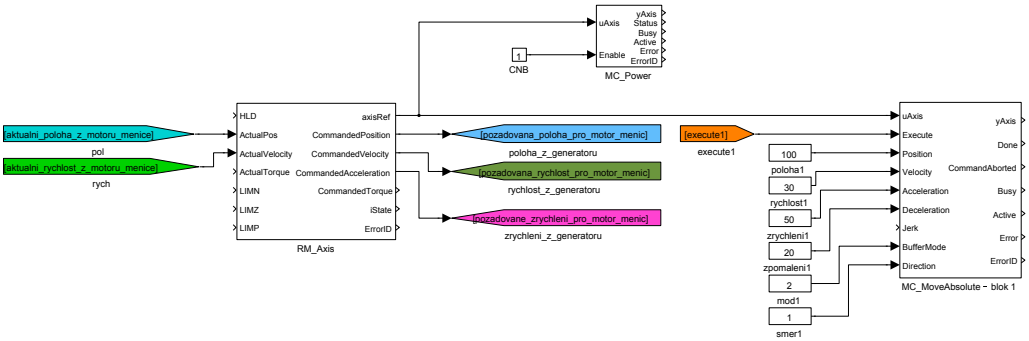
`AxisType`	Druh osy `1` ..... lineární osa `2` ..... cyklická osa s cyklickým polohovým senzorem `3` ..... cyklická osa s lineárním polohovým senzorem	⊙`1`	`Long (I32)`
`EnableLimitPos`	Příznak kontroly maximální polohy v kladném směru (tj. když je zaškrtnuto, je platné `MaxPosAppl`)		`Bool`
`MaxPosAppl`	Maximální hodnota v kladném směru pro pohybové bloky. Hodnota musí být menší než `MaxPosSystem` pro lineární osu. Pro cyklickou osu s lineárním senzorem hodnota omezuje na několik otáček (vhodné pro robotické aplikace) a musí být větší než `MaxPosSystem`.		`Double (F64)`
`MaxPosSystem`	Maximální hodnota v kladném směru (fyzický doraz)		`Double (F64)`
`EnableLimitNeg`	Příznak kontroly maximální polohy v záporném směru (tj. když je zaškrtnuto, je platné `MinPosAppl`)		`Bool`

`MinPosAppl`	Maximální hodnota v záporném směru pro pohybové bloky Hodnota musí být větší než (nastává dříve) `MinPosSystem` 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) `MaxPosSystem`.	`Double (F64)`
`MinPosSystem`	Maximální hodnota v záporném směru (fyzický doraz)	`Double (F64)`
`EnablePosLagMonitor`	Příznak kontroly skluzu (rozdílu mezi požadovanou a skutečnou polohou)	`Bool`
`MaxPositionLag`	Maximální hodnota skluzu	`Double (F64)`
`MaxVelocitySystem`	Maximální hodnota rychlosti (havarijní)	`Double (F64)`
`MaxVelocityAppl`	Maximální hodnota rychlosti (pracovní)	`Double (F64)`
`MaxAccelerationSystem`	Maximální hodnota zrychlení (havarijní)	`Double (F64)`
`MaxAccelerationAppl`	Maximální hodnota zrychlení (pracovní)	`Double (F64)`
`MaxDecelerationSystem`	Maximální hodnota brždění (havarijní)	`Double (F64)`
`MaxDecelerationAppl`	Maximální hodnota brždění (pracovní)	`Double (F64)`
`DefaultJerk`	Maximální doporučená změna zrychlení [unit/s^3] Skutečná změna zrychlení se nekontroluje a může překročit tuto hodnotu.	`Double (F64)`
`MaxTorque`	Maximální hodnota momentu/síly	`Double (F64)`
`TorqueRatio`	Poměr momentu a zrychlení. Pokud moment osy není určen jiným způsobem (např. pomocí bloku `MC_TorqueControl`), určuje se moment ze zrychlení pomocí tohoto koeficientu.	`Double (F64)`
`LoopDelay`	zpoždění mezi požadovanou a skutečnou hodnotou [s] Zpoždění nastává vlivem periody vzorkování, komunikace s externím kontrolérem, zpožděním regulační smyčky. Nastavení umožňuje přesněji kontrolovat PositionLag. (not yet implemented)	`Double (F64)`
`StartMode`	Volby režimu při zapnutí ⊙`1` `1` ..... při zapnutí je pohyb zastaven `2` ..... při zapnutí se pohyb pokračuje aktuální rychlostí motoru	`Long (I32)`
`HomingRequired`	Pokud je zaškrtnuto, tak jsou blokovány všechny pohyby (kromě `MC_Home`), dokud se neprovede homing	`Bool`

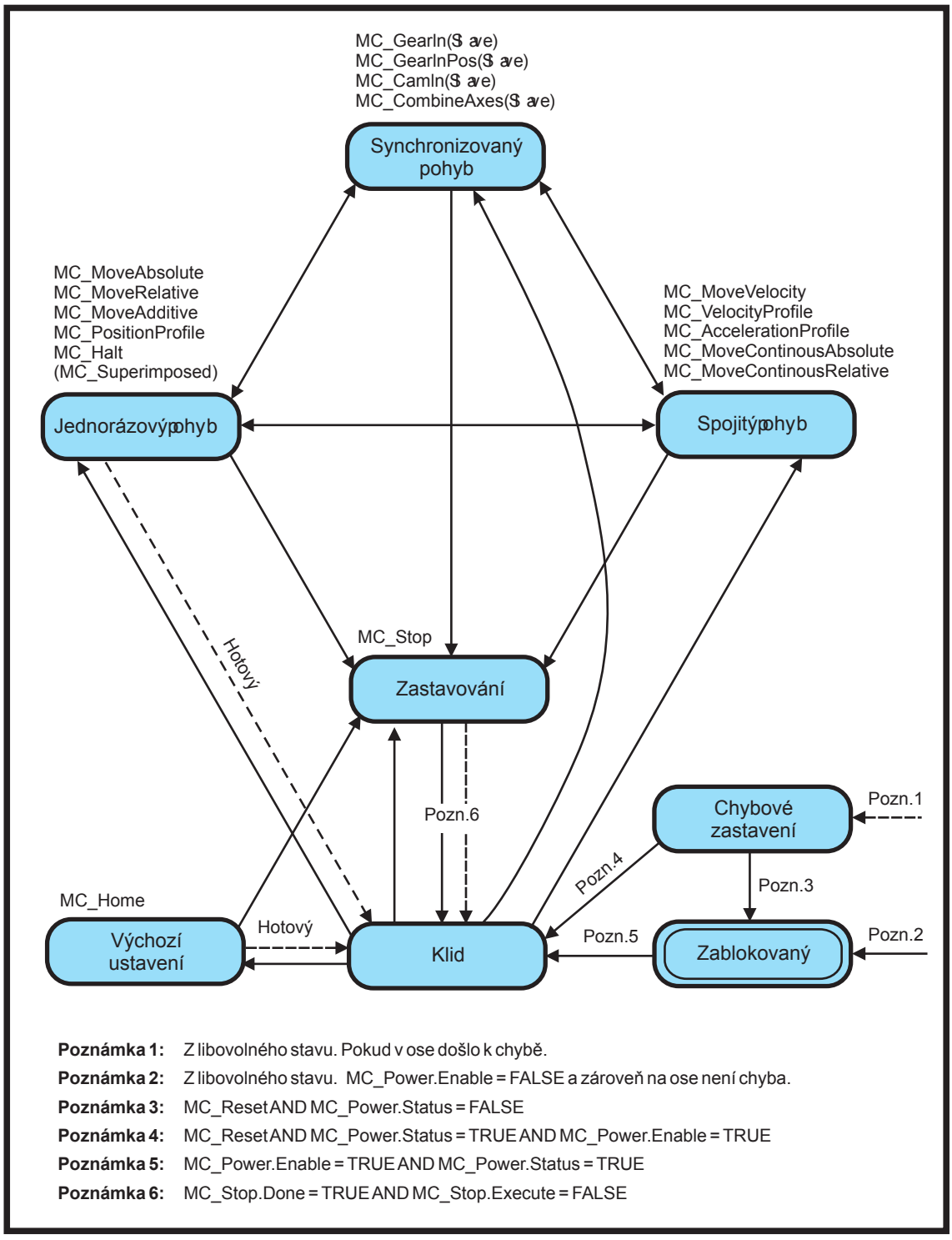
Příklad

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 nepřípustného 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



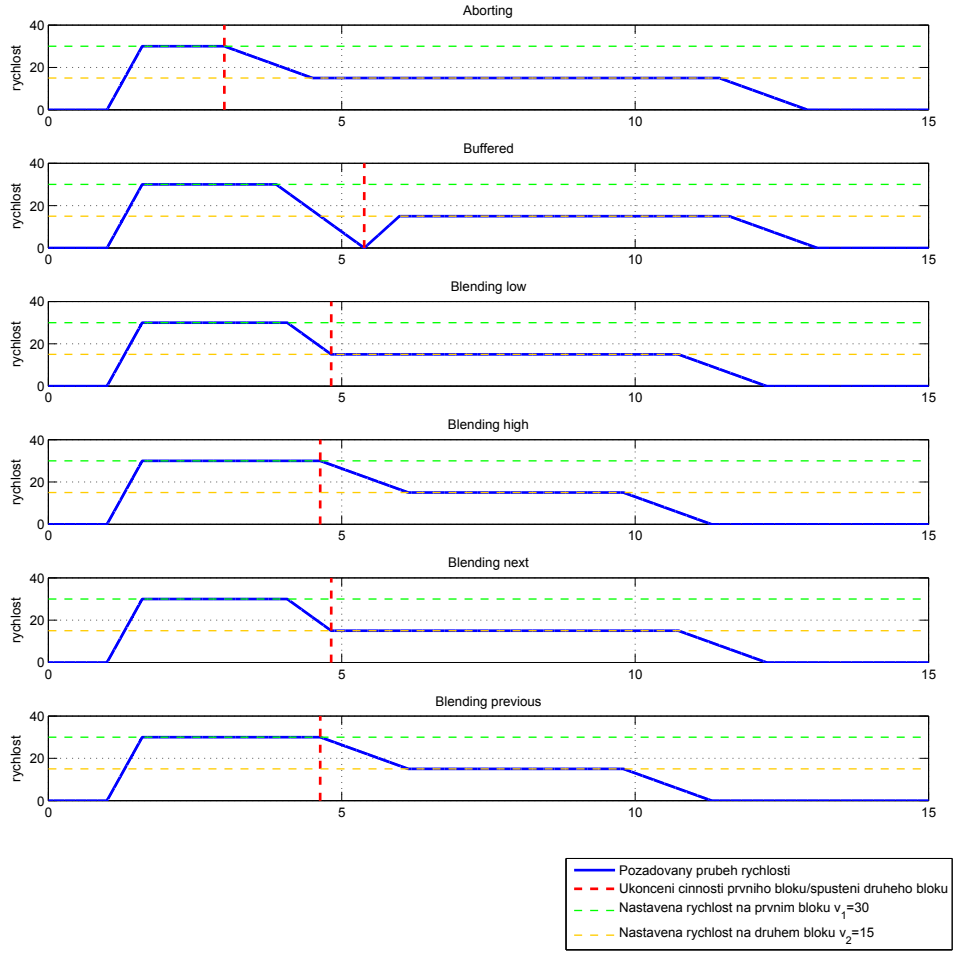
Poznámka 1: Z libovolného stavu. Pokud v ose došlo k chybě.
Poznámka 2: Z libovolného stavu. MC_Power.Enable = FALSE a zároveň na ose není chyba.
Poznámka 3: MC_Reset AND MC_Power.Status = FALSE
Poznámka 4: MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE
Poznámka 5: MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Poznámka 6: MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

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`.

Aborting	nový pohyb okamžitě přerušuje původní příkaz
Buffered	nový pohyb je vykonán po dokončení původního příkazu, neprobíhá žádné míchání
Blending low	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
Blending high	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
Blending previous	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
Blending next	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

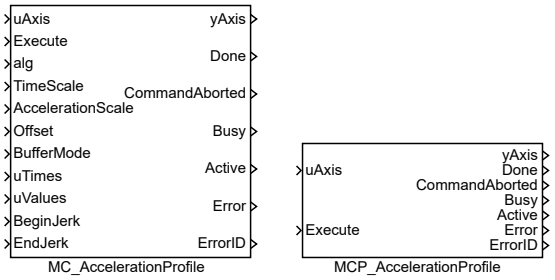
Ilustrace míchání pohybů



MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení)

Symboly 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

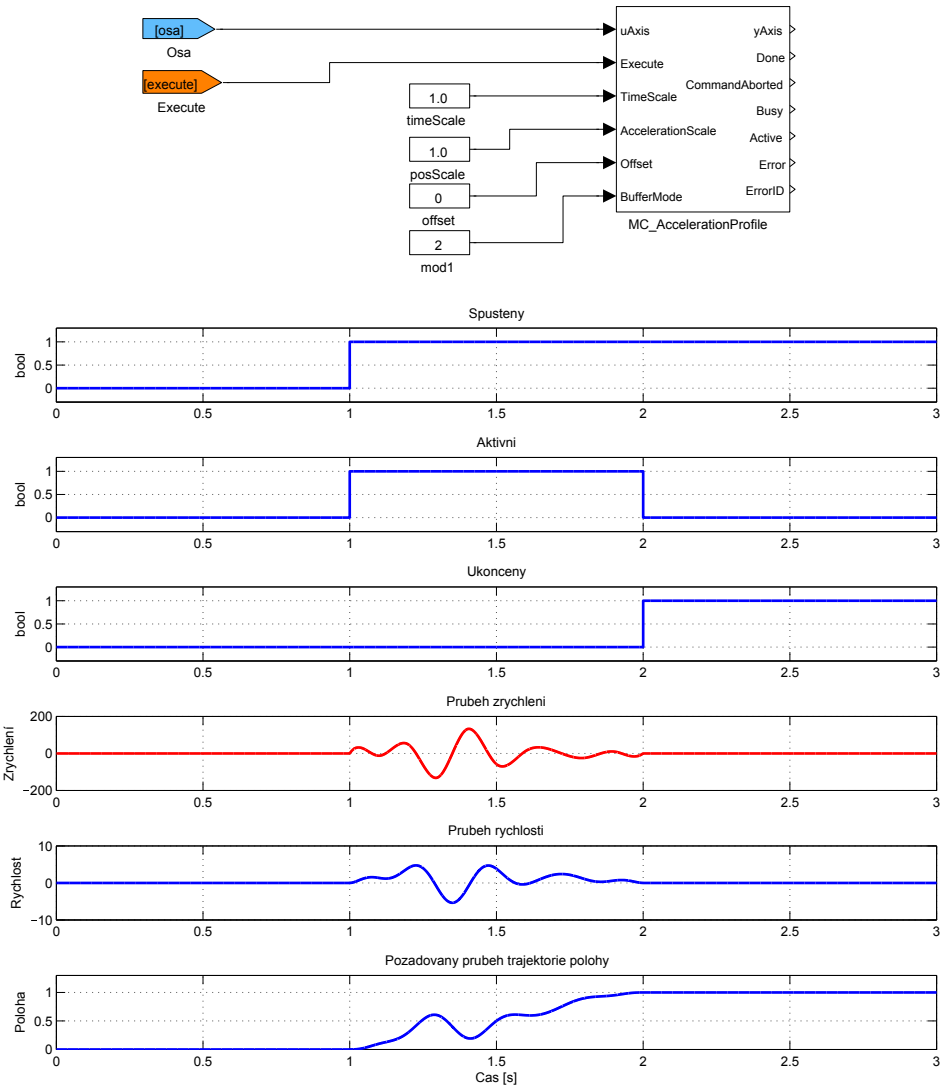
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`alg`	Typ interpolace ⊙2 `1` ..... tabulka čas/hodnota (interpolace polynomem 3. řádu) `2` ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu) `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)	`Long (I32)`
`nmax`	maximalni počet intervalů/bodů profilu ⊙3	`Long (I32)`
`TimeScale`	Konstanta násobení pro přepočet časové osy profilu	`Double (F64)`
`AccelerationScale`	Konstanta násobení pro přepočet hodnotové osy profilu	`Double (F64)`
`Offset`	Aditivní konstanta pro přepočet hodnotové osy profilu	`Double (F64)`
`BufferMode`	Režim převzetí osy	`Long (I32)`
`uTimes`	vektor s časovými hodnotami	`Reference`

`uValues`	vektor s hodnotami zrychlení nebo koeficienty polynomů `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)	`Reference`

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`

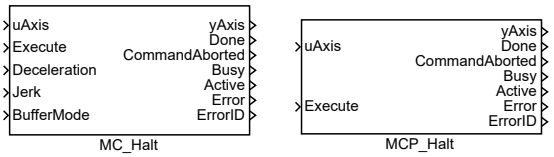
Příklad


[osa]
Osa
[execute]
Execute
1.0
timeScale
1.0
posScale
0
offset
2
mod1
uAxis
Execute
TimeScale
AccelerationScale
Offset
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_AccelerationProfile
Spusteny
Aktivni
Ukonceny
Prubeh zrychleni
Zrychlení
Prubeh rychlosti
Rychlost
Pozadovany prubeh trajektorie polohy
Poloha
bool
Cas [s]

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

`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^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`

Výstupy

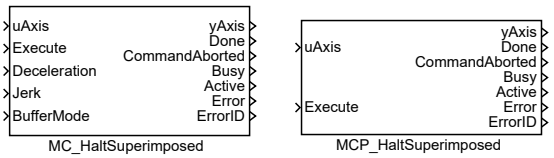
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`

`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)

Symboly 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^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`

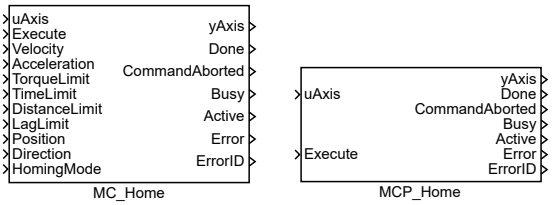
Výstupy

`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 brždění. 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

`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^2]	`Double (F64)`
`TorqueLimit`	Maximální povolený moment/síla	`Double (F64)`
`TimeLimit`	Maximální povolený čas pro celý algoritmus bloku [s]	`Double (F64)`
`DistanceLimit`	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	`Double (F64)`
`Position`	Požadovaná poloha [unit]	`Double (F64)`
`Direction`	Směr pohybu (jen pro cyklické osy nebo speciální případy) `1` ..... kladný `2` ..... nejkratší `3` ..... záporný `4` ..... aktuální	`Long (I32)`
`HomingMode`	Algoritmus hledání výchozí pozice `1` ..... nulový spínač `2` ..... koncové spínače `3` ..... nulový pulz `4` ..... uživatelská hodnota `5` ..... absolutní snímač polohy `6` ..... mechanický doraz	`Long (I32)`

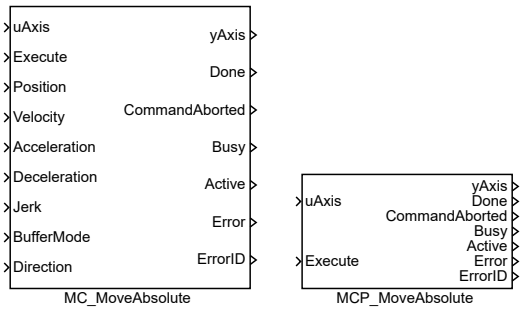
Výstupy

`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_MoveAbsolute`, `MCP_MoveAbsolute` – 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_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.

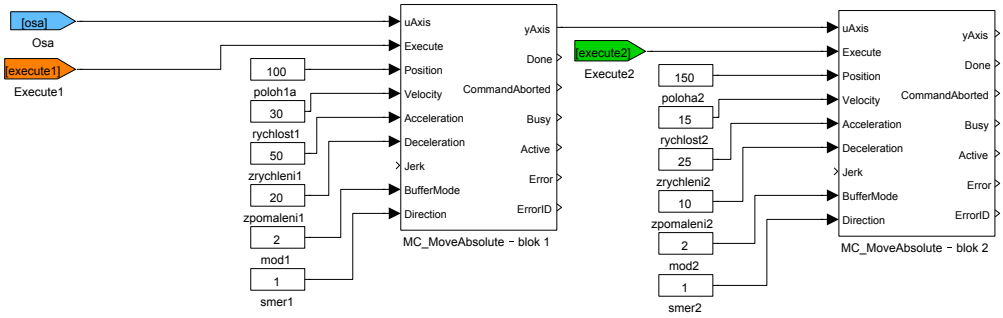
Vstupy

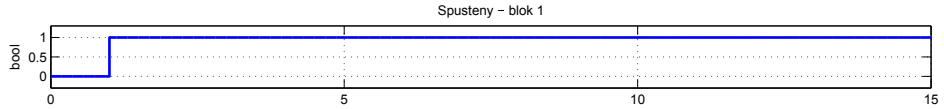
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Position`	Požadovaná poloha [unit]	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`Direction`	Směr pohybu (jen pro cyklické osy nebo speciální případy) `1` ..... kladný `2` ..... nejkratší `3` ..... záporný `4` ..... aktuální	`Long (I32)`

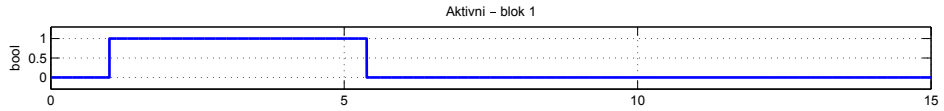
Výstupy

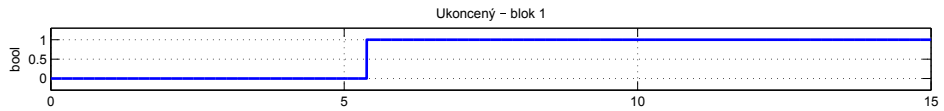
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

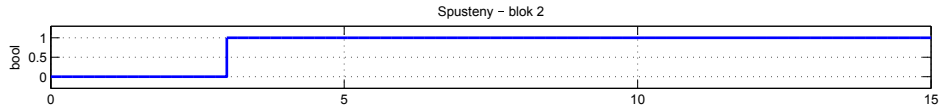
Příklad

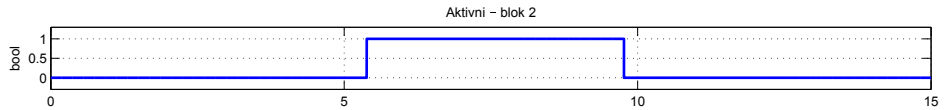

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
1
smer1
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 1
[execute2]
Execute2
150
poloha2
15
rychlost2
25
zrychleni2
10
zpomaleni2
2
mod2
1
smer2
MC_MoveAbsolute - blok 2

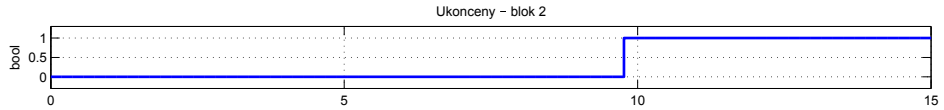

Spusteny - blok 1
bool
1
0.5
0
0
5
10
15

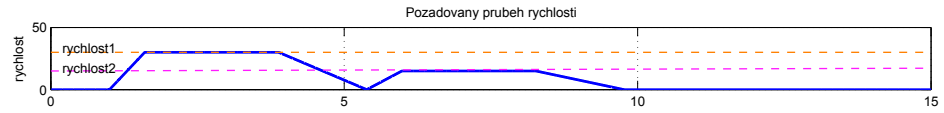

Aktivni - blok 1
bool
1
0.5
0
0
5
10
15

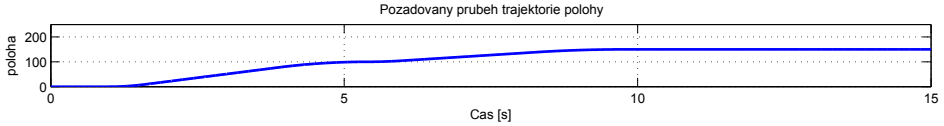

Ukonceny - blok 1
bool
1
0.5
0
0
5
10
15


Spusteny - blok 2
bool
1
0.5
0
0
5
10
15


Aktivni - blok 2
bool
1
0.5
0
0
5
10
15


Ukonceny - blok 2
bool
1
0.5
0
0
5
10
15

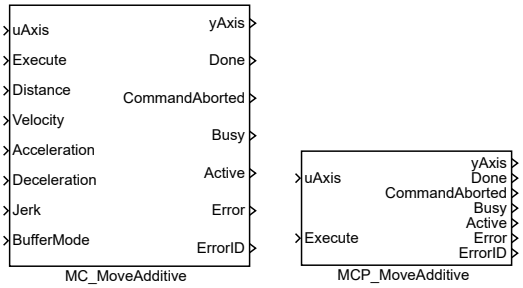

Pozadovany prubeh rychlosti
rychlost
50
0
rychlost1
rychlost2
0
5
10
15


Pozadovany prubeh trajektorie polohy
poloha
200
100
0
0
5
10
15
Cas [s]

MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)

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_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.

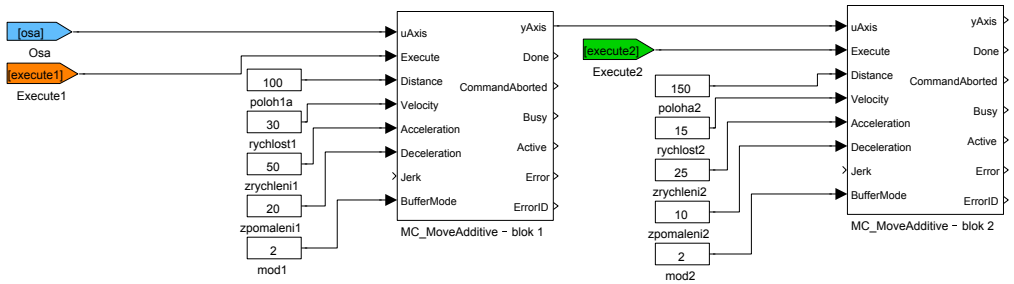
Vstupy

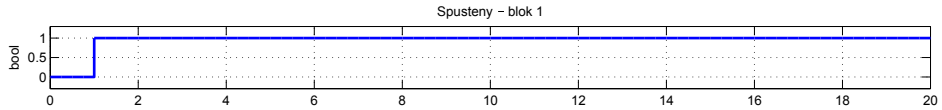
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`

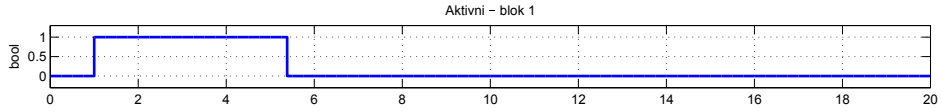
Výstupy

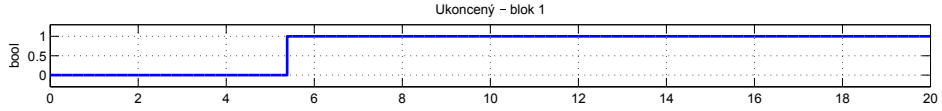
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

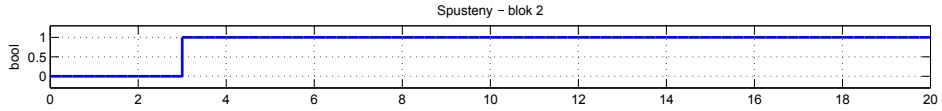
Příklad

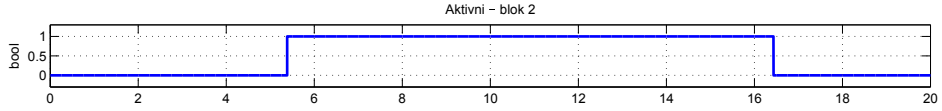

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
uAxis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAdditive - blok 1
[execute2]
Execute2
150
poloha2
15
rychlost2
25
zrychleni2
10
zpomaleni2
2
mod2
MC_MoveAdditive - blok 2

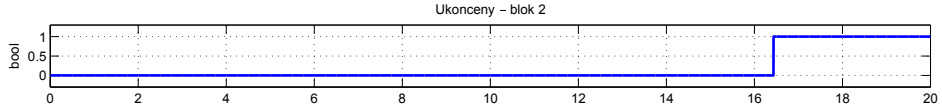

Spusteny - blok 1
bool


Aktivni - blok 1
bool

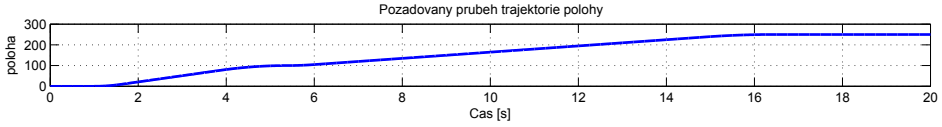

Ukonceny - blok 1
bool


Spusteny - blok 2
bool


Aktivni - blok 2
bool


Ukonceny - blok 2
bool

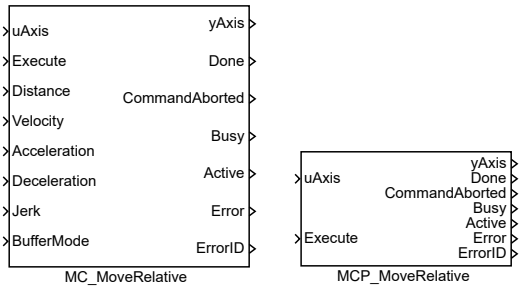

Pozadovany prubeh rychlosti
rychlost
rychlost1
rychlost2


Pozadovany prubeh trajektorie polohy
poloha
Cas [s]

`MC_MoveRelative`, `MCP_MoveRelative` – Pohyb do pozice (relativně k okamžiku spuštění)

Symboly 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.

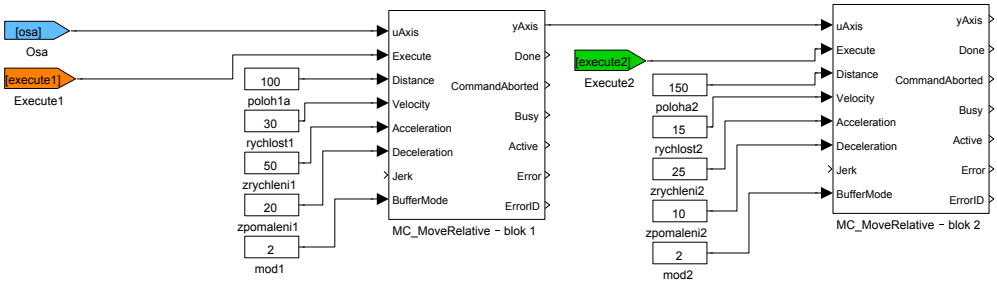
Vstupy

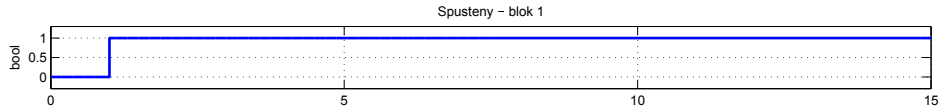
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`

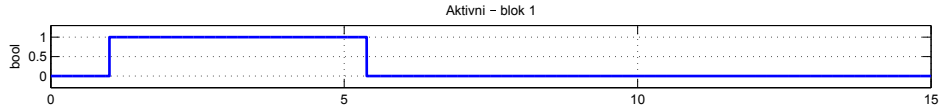
Výstupy

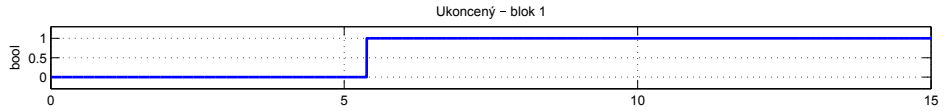
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

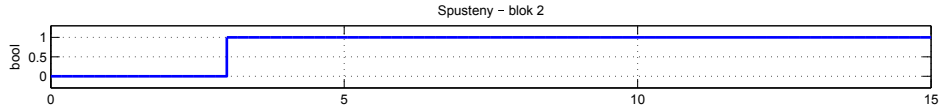
Příklad

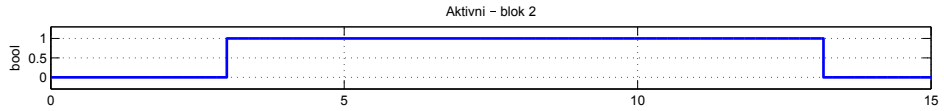

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
uAxis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveRelative – blok 1
[execute2]
Execute2
150
poloha2
15
rychlost2
25
zrychleni2
10
zpomaleni2
2
mod2
MC_MoveRelative – blok 2

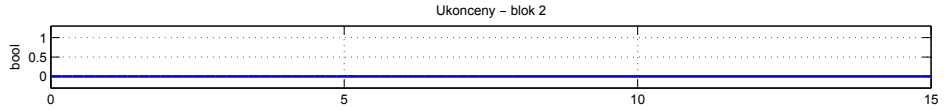

Spusteny – blok 1
bool
1
0.5
0
0
5
10
15

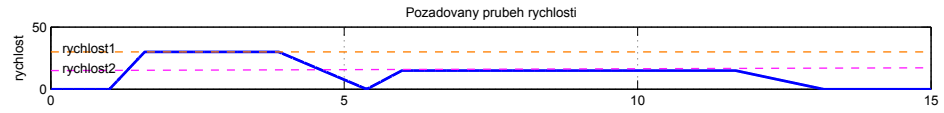

Aktivni – blok 1
bool
1
0.5
0
0
5
10
15

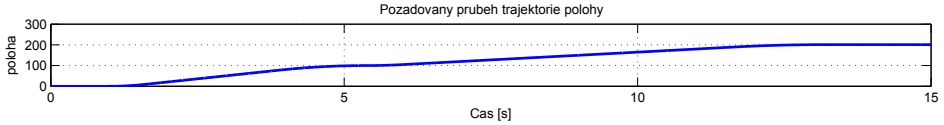

Ukoncený – blok 1
bool
1
0.5
0
0
5
10
15


Spusteny – blok 2
bool
1
0.5
0
0
5
10
15


Aktivni – blok 2
bool
1
0.5
0
0
5
10
15


Ukonceny – blok 2
bool
1
0.5
0
0
5
10
15

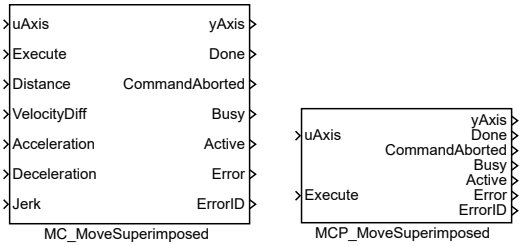

Pozadovany prubeh rychlosti
rychlost
50
0
rychlost1
rychlost2
0
5
10
15


Pozadovany prubeh trajektorie polohy
poloha
300
200
100
0
0
5
10
15
Cas [s]

MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)

Symboly 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).

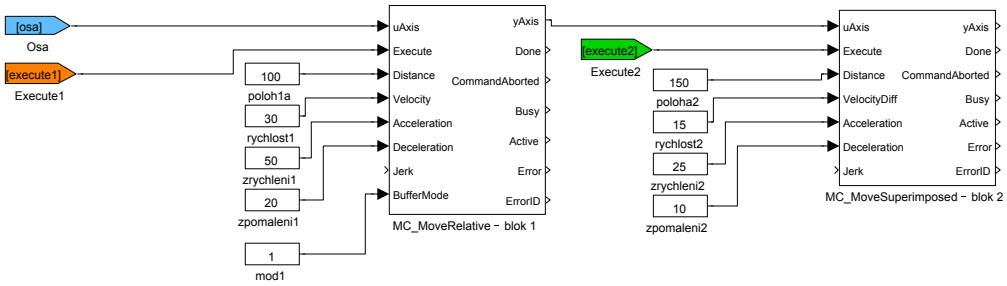
Vstupy

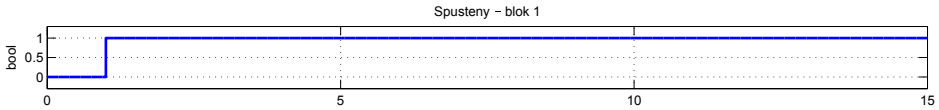
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	`Double (F64)`
`VelocityDiff`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`

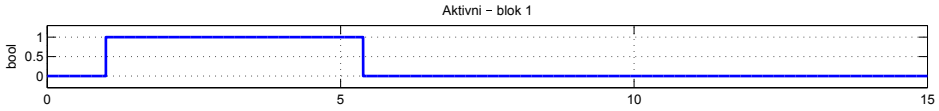
Výstupy

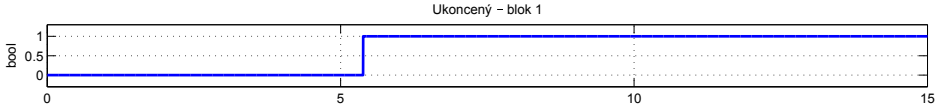
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

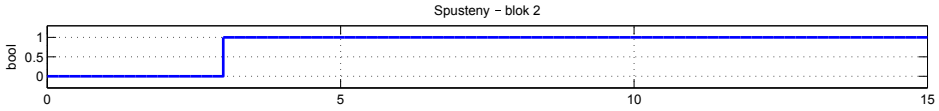
Příklad

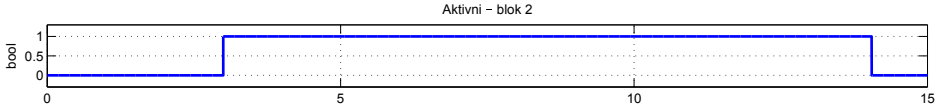

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
1
mod1
uAxis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveRelative - blok 1
[execute2]
Execute2
150
poloha2
15
rychlost2
25
zrychleni2
10
zpomaleni2
uAxis
Execute
Distance
VelocityDiff
Acceleration
Deceleration
Jerk
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveSuperimposed - blok 2

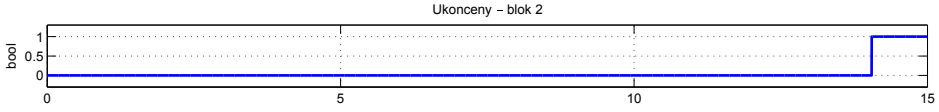

Spusteny - blok 1
bool
1
0.5
0
0
5
10
15

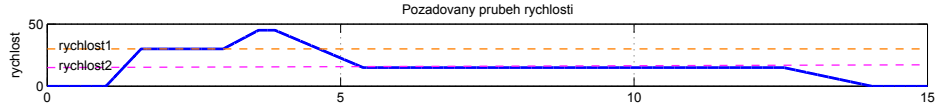

Aktivni - blok 1
bool
1
0.5
0
0
5
10
15

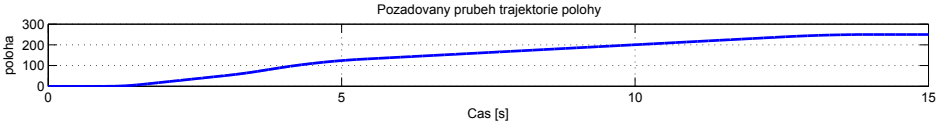

Ukonceny - blok 1
bool
1
0.5
0
0
5
10
15


Spusteny - blok 2
bool
1
0.5
0
0
5
10
15


Aktivni - blok 2
bool
1
0.5
0
0
5
10
15


Ukonceny - blok 2
bool
1
0.5
0
0
5
10
15

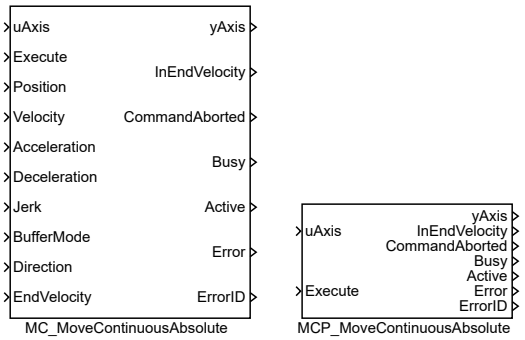

Pozadovany prubeh rychlosti
rychlost
50
0
rychlost1
rychlost2
0
5
10
15


Pozadovany prubeh trajektorie polohy
poloha
300
200
100
0
0
5
10
15
Cas [s]

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

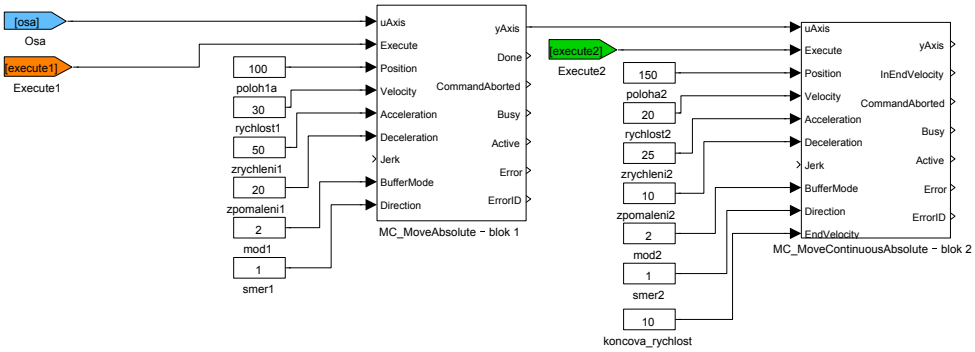
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Position`	Požadovaná poloha [unit]	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`Direction`	Směr pohybu (jen pro cyklické osy nebo speciální případy) `1` ..... kladný `2` ..... nejkratší `3` ..... záporný `4` ..... aktuální	`Long (I32)`
`EndVelocity`	Koncová rychlost	`Double (F64)`

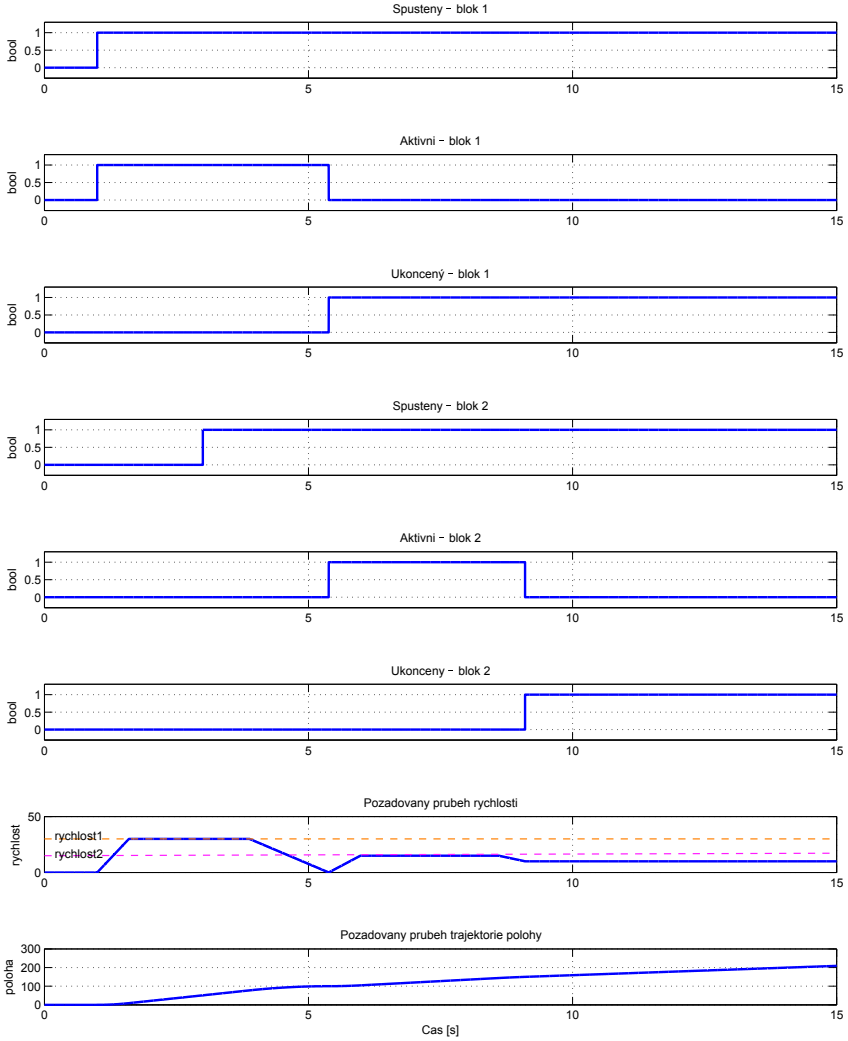
Výstupy

`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`InEndVelocity`	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

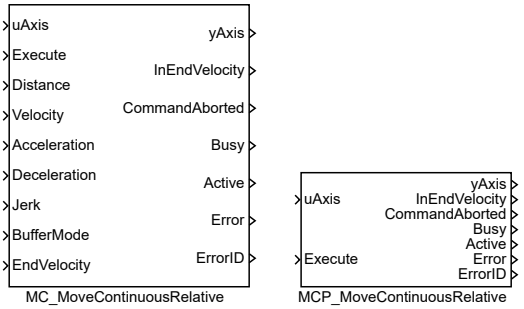

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
1
smer1
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 1
[execute2]
Execute2
150
poloha2
20
rychlost2
25
zrychleni2
10
zpomaleni2
2
mod2
1
smer2
10
koncova_rychlost
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
EndVelocity
yAxis
InEndVelocity
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveContinuousAbsolute - blok 2


Spusteny - blok 1
Aktivni - blok 1
Ukonceny - blok 1
Spusteny - blok 2
Aktivni - blok 2
Ukonceny - blok 2
bool
Pozadovany prubeh rychlosti
rychlost
rychlost1
rychlost2
Pozadovany prubeh trajektorie polohy
poloha
Cas [s]

MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symboly 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

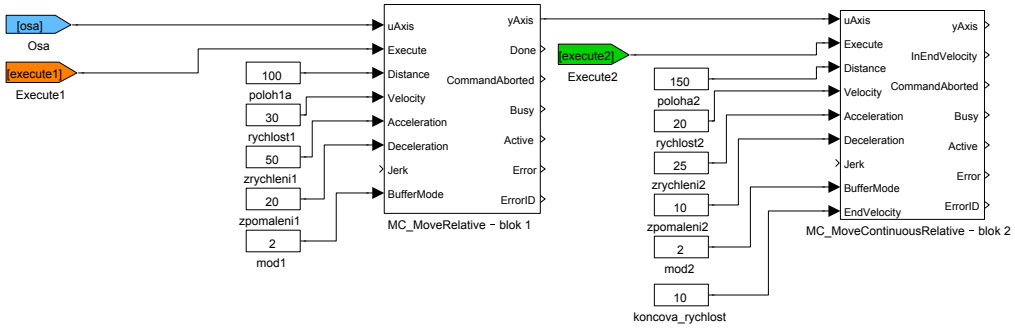
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`EndVelocity`	Koncová rychlost	`Long (I32)`

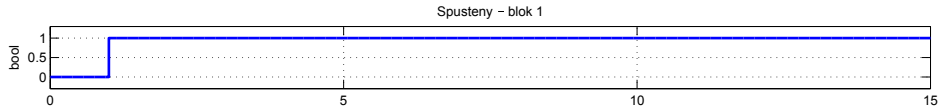
Výstupy

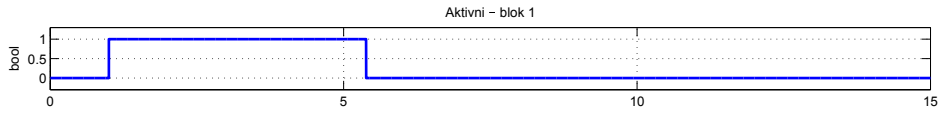
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`InEndVelocity`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`

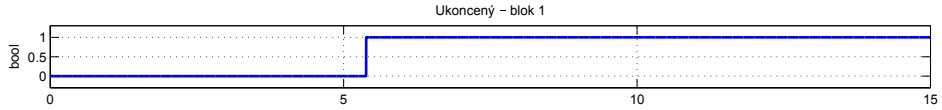
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

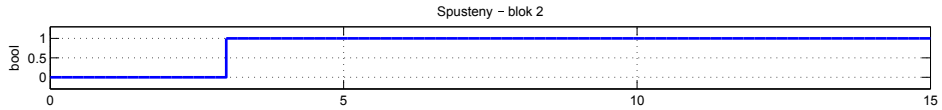
Příklad

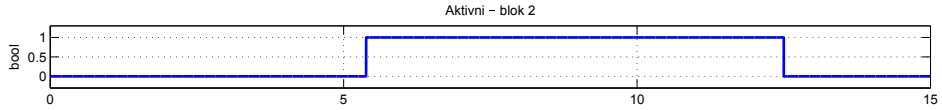

[osa]
Osa
[execute1]
Execute1
100
poloh1a
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
uAxis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveRelative - blok 1
[execute2]
Execute2
150
poloha2
20
rychlost2
25
zrychleni2
10
zpomaleni2
2
mod2
10
koncova_rychlost
uAxis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
EndVelocity
yAxis
InEndVelocity
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveContinuousRelative - blok 2

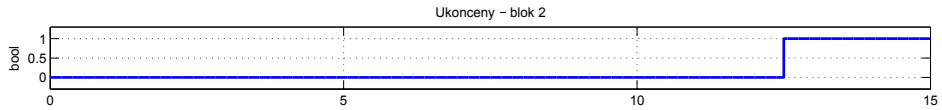

Spusteny - blok 1
bool
1
0.5
0
0
5
10
15

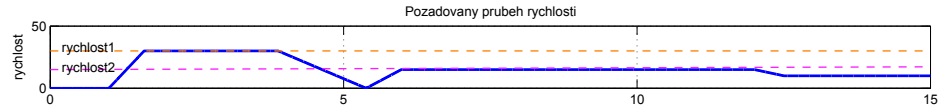

Aktivni - blok 1
bool
1
0.5
0
0
5
10
15

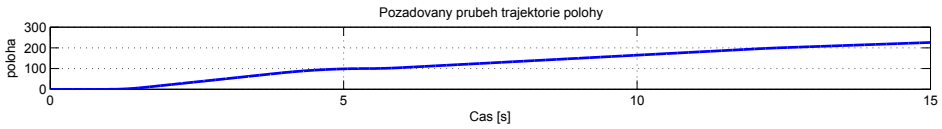

Ukonceny - blok 1
bool
1
0.5
0
0
5
10
15


Spusteny - blok 2
bool
1
0.5
0
0
5
10
15


Aktivni - blok 2
bool
1
0.5
0
0
5
10
15


Ukonceny - blok 2
bool
1
0.5
0
0
5
10
15

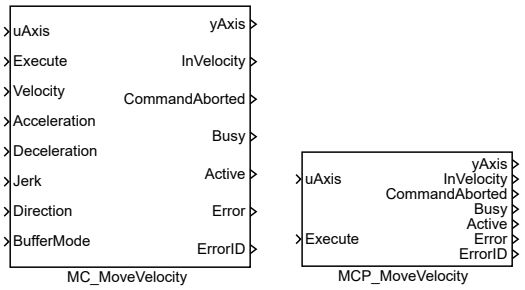

Pozadovany prubeh rychlosti
rychlost
50
0
rychlost1
rychlost2
0
5
10
15


Pozadovany prubeh trajektorie polohy
poloha
300
200
100
0
0
5
10
15
Cas [s]

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

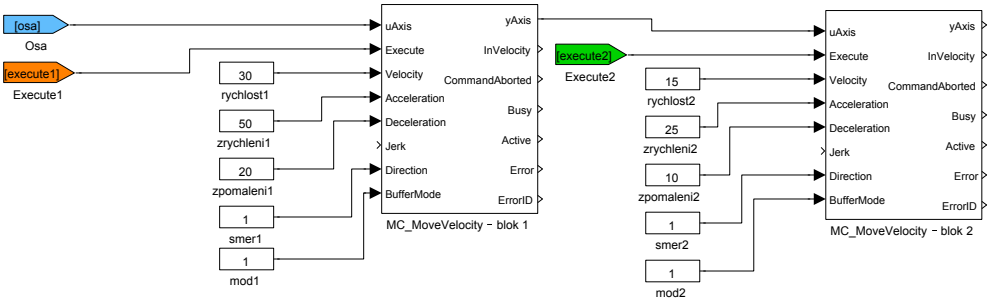
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`Direction`	Směr pohybu (jen pro cyklické osy nebo speciální případy) `1` ..... kladný `2` ..... nejkratší `3` ..... záporný `4` ..... aktuální	`Long (I32)`

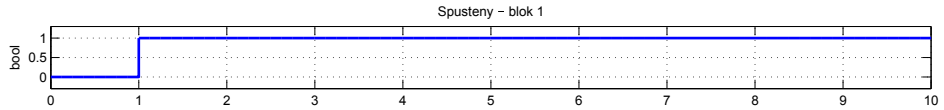
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	`Long (I32)`

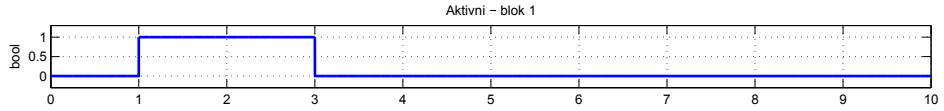
Výstupy

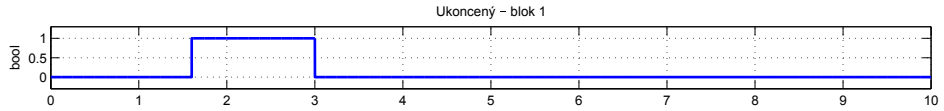
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`InVelocity`	Příznak dosažení požadované rychlosti	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

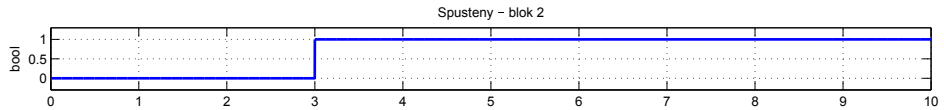
Příklad

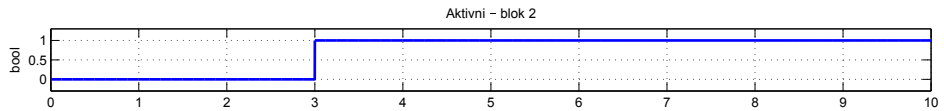

[osa]
Osa
[execute1]
Execute1
30
rychlost1
50
zrychleni1
20
zpomaleni1
1
smer1
1
mod1
uAxis
Execute
Velocity
Acceleration
Deceleration
Jerk
Direction
BufferMode
yAxis
InVelocity
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveVelocity - blok 1
[execute2]
Execute2
15
rychlost2
25
zrychleni2
10
zpomaleni2
1
smer2
1
mod2
MC_MoveVelocity - blok 2

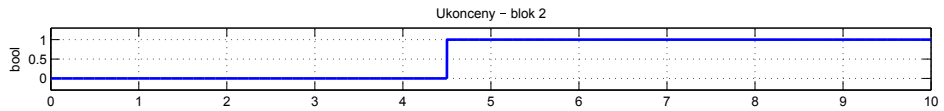

Spusteny - blok 1
bool

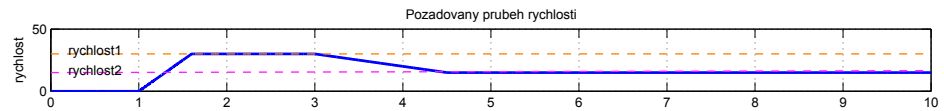

Aktivni - blok 1
bool

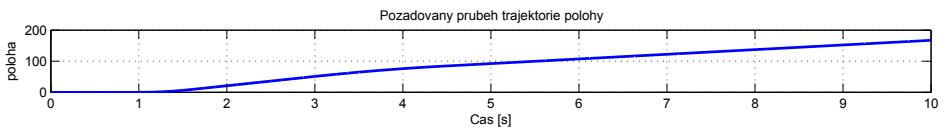

Ukoncený - blok 1
bool


Spusteny - blok 2
bool


Aktivni - blok 2
bool


Ukonceny - blok 2
bool

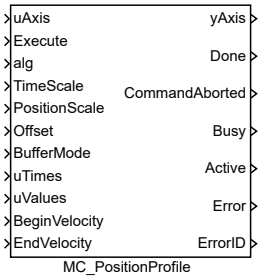

Pozadovany prubeh rychlosti
rychlost
rychlost1
rychlost2

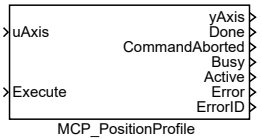

Pozadovany prubeh trajektorie polohy
poloha
Cas [s]

`MC_PositionProfile`, `MCP_PositionProfile` – Generování trajektorie (poloha)

Symboly 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 rzchlosti, čtvrtý (nepovinný) sloupec zrychlení.

Vstupy

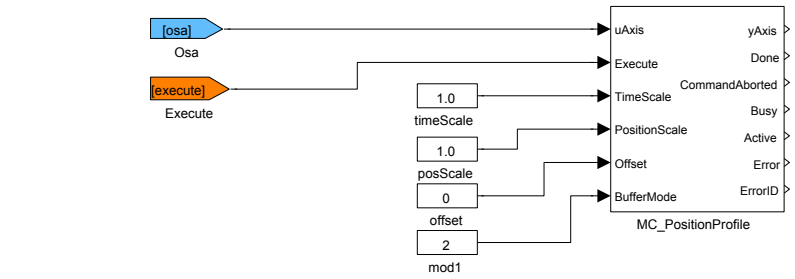
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`alg`	Typ interpolace ⊙2 `1` ..... tabulka čas/hodnota (interpolace polynomem 3. řádu) `2` ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu) `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)	`Long (I32)`
`nmax`	maximalni počet intervalů/bodů profilu ⊙3	`Long (I32)`
`TimeScale`	Konstanta násobení pro přepočet časové osy profilu	`Double (F64)`
`PositionScale`	Konstanta násobení pro přepočet hodnotové osy profilu	`Double (F64)`
`Offset`	Aditivní konstanta pro přepočet hodnotové osy profilu	`Double (F64)`
`uTimes`	vektor s časovými hodnotami	`Reference`
`uValues`	vektor s hodnotami polohy nebo koeficienty polynomů	`Reference`

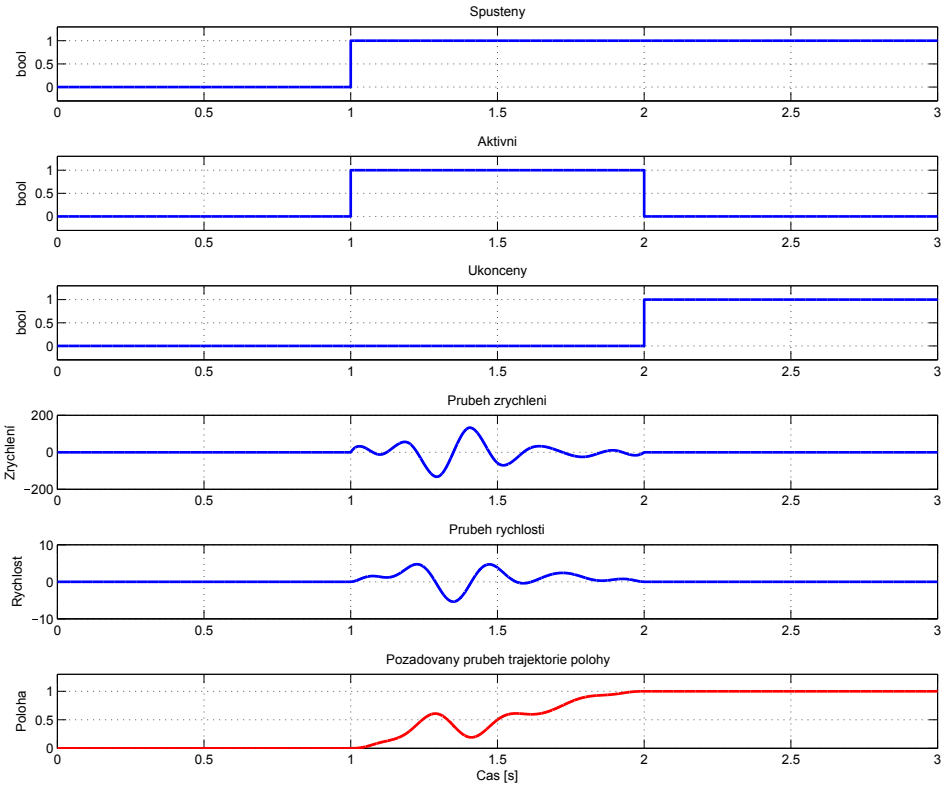
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	`Long (I32)`
`BeginVelocity`	počáteční hodnota rychlosti (jen alg=5 nebo 6)	`Double (F64)`
`EndVelocity`	koncová hodnota rychlosti (jen alg=5 nebo 6)	`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`

Příklad

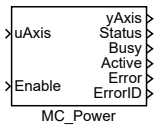

[osa]
Osa
[execute]
Execute
1.0
timeScale
1.0
posScale
0
offset
2
mod1
uAxis
Execute
TimeScale
PositionScale
Offset
BufferMode
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_PositionProfile


Spusteny
Aktivni
Ukonceny
Prubeh zrychleni
Prubeh rychlosti
Pozadovany prubeh trajektorie polohy
bool
Zrychlení
Rychlost
Poloha
Cas [s]

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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

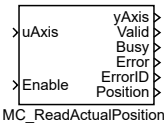
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Status	Skutečný stav napájení osy	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_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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

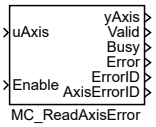
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error
Position	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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

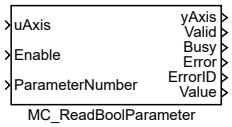
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error
AxisErrorID	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

`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`
`ParameterNumber`	Číslo požadovaného parametru `4` ..... kontrola kladného omezení polohy `5` ..... kontrola záporného omezení polohy `6` ..... kontrola regulační odchylky (poloha)	`Long (I32)`

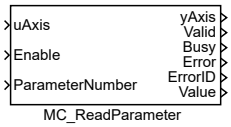
Výstupy

`yAxis`	OOdkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`
`Value`	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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
ParameterNumber	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

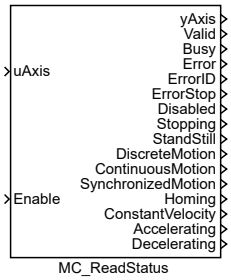
Výstupy

`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`
`Value`	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

`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`

Výstupy

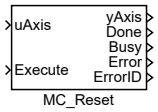
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`
`ErrorStop`	Osa je ve stavu CHYBA	`Bool`
`Disabled`	Osa je ve stavu VYPNUTO	`Bool`
`Stopping`	Osa je ve stavu STOP	`Bool`
`StandStill`	Osa je ve stavu PŘIPRAVEN	`Bool`
`DiscreteMotion`	Osa je ve stavu JEDNORÁZOVÝ POHYB	`Bool`
`ContinuousMotion`	Osa je ve stavu TRVALÝ POHYB	`Bool`
`SynchronizedMotion`	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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

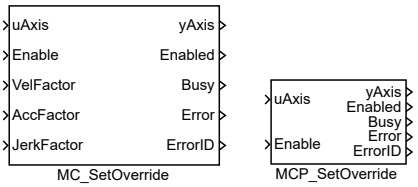
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	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

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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
VelFactor	Faktor násobení pro rychlost	Double (F64)
AccFactor	Faktor násobení pro zrychlení	Double (F64)
JerkFactor	Faktor násobení pro změnu zrychlení	Double (F64)

Výstupy

`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`Enabled`	Povolení funkce bloku (aktivace výstupů)	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

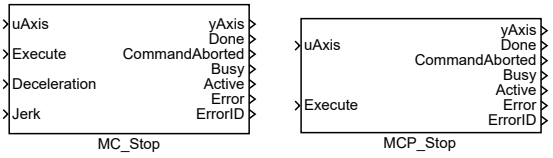
Parametr

`diff`	Pásmo necitlivosti (pro přepočet 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

`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^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`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`

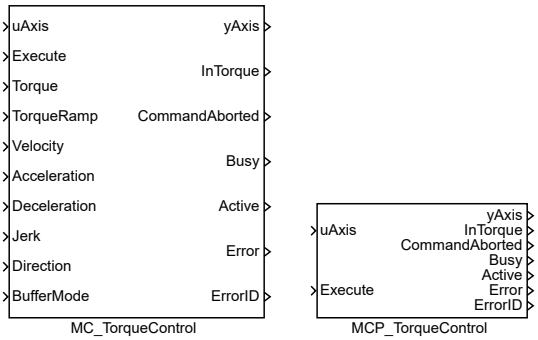
`ErrorID` Výsledek poslední operace `Error`

`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^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [uunit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`Direction`	Směr pohybu (jen pro cyklické osy nebo speciální případy) `1` ..... kladný `2` ..... nejkratší `3` ..... záporný `4` ..... aktuální	`Long (I32)`

`BufferMode`	Režim převzetí osy `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)	`Long (I32)`

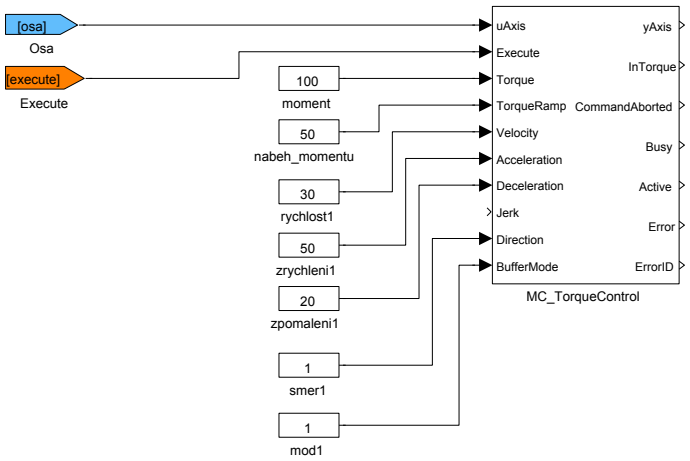
Výstupy

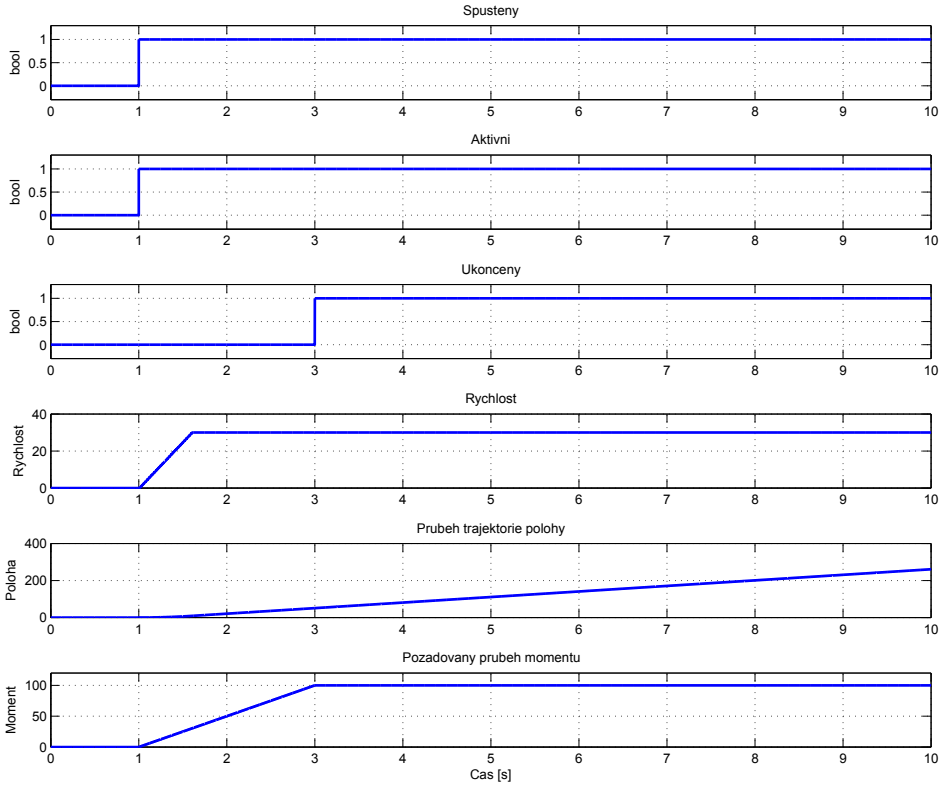
`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`InTorque`	Příznak dosažení požadovaného momentu/síly	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

Parametr

`kma`	Poměr mezi sílou/momentem a zrychlením	`Double (F64)`

Příklad

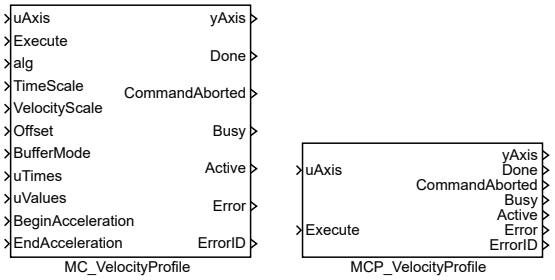

[osa]
Osa
[execute]
Execute
100
moment
50
nabeh_momentu
30
rychlost1
50
zrychleni1
20
zpomaleni1
1
smer1
1
mod1
uAxis
Execute
Torque
TorqueRamp
Velocity
Acceleration
Deceleration
Jerk
Direction
BufferMode
yAxis
InTorque
CommandAborted
Busy
Active
Error
ErrorID
MC_TorqueControl


Spusteny
bool
Aktivni
bool
Ukonceny
bool
Rychlost
Rychlost
Prubeh trajektorie polohy
Poloha
Pozadovany prubeh momentu
Moment
Cas [s]

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 $x = 1$. 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 `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: Block 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

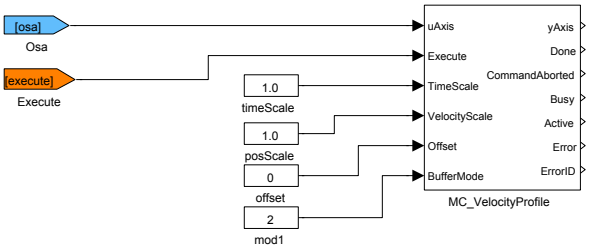
`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`alg`	Typ interpolace ⊙2 `1` ..... tabulka čas/hodnota (interpolace polynomem 3. řádu) `2` ..... hodnoty ve stejném intervalu (interpolace polynomem 3. řádu) `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)	`Long (I32)`
`nmax`	maximalni počet intervalů/bodů profilu ⊙3	`Long (I32)`
`TimeScale`	Konstanta násobení pro přepočet časové osy profilu	`Double (F64)`
`VelocityScale`	Konstanta násobení pro přepočet hodnotové osy profilu	`Double (F64)`
`Offset`	Aditivní konstanta pro přepočet hodnotové osy profilu	`Double (F64)`
`BufferMode`	Režim převzetí osy	`Long (I32)`
`uTimes`	vektor s časovými hodnotami	`Reference`

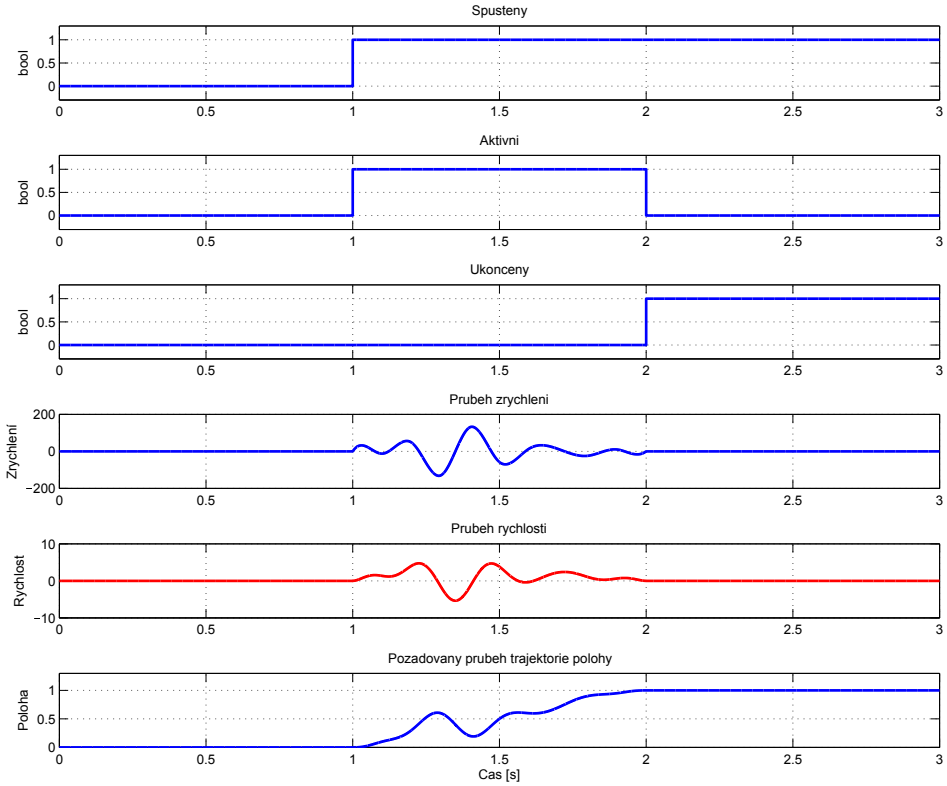
`uValues`	vektor s hodnotami rychlosti nebo koeficienty polynomů `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `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)	`Reference`
`BeginAcceleration`	počáteční hodnota zrychlení (jen alg=5 nebo 6)	`Double (F64)`
`EndAcceleration`	koncová hodnota (jen alg=5 nebo 6)	`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`

Příklad

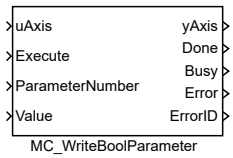



Spusteny
bool
Aktivni
bool
Ukonceny
bool
Prubeh zrychleni
Zrychlení
Prubeh rychlosti
Rychlost
Pozadovany prubeh trajektorie polohy
Poloha
Cas [s]

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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
ParameterNumber	Číslo požadovaného parametru 4 ..... kontrola kladného omezení polohy 5 ..... kontrola záporného omezení polohy 6 ..... kontrola regulační odchylky (poloha)	Long (I32)
Value	Hodnota parametru	Bool

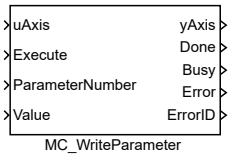
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	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

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

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
ParameterNumber	Číslo požadovaného parametru 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	Long (I32)
Value	Hodnota parametru	Double (F64)

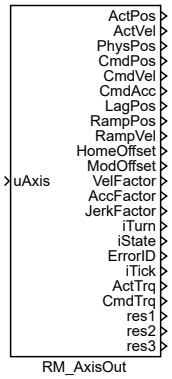
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	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

RM_AxisOut – Výstupní blok osy

Symbol bloku

Licence: MOTION CONTROL



Popis funkce

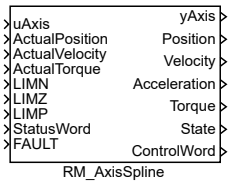
Blok RM_AxisOut zpřístupňuje důležité stavy bloku RM_Axis. Některé z výstupů jsou přímo na bloku RM_Axis, ale ty jsou o krok zpožděné. Aby nebyly výstupy zpožděné i na tomto bloku, je potřeba jej zařadit jako poslední.

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 řídícím 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`.

Měnič 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` (zadává 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.

Řídící obvody motoru obvykle pracují s celočíselnými typy (respektive v celočíselných typech předávají hodnoty nadřízenému 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 (řídící obvody motoru).

Měniče 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`. 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 Bspline 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 neprojíždí 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

`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef`–`uAxis` nebo `yAxis`–`uAxis`)	`Reference`
`ActualPosition`	Skutečná poloha osy [drive unit]	`Double (F64)`
`ActualVelocity`	Skutečná rychlost osy [drive unit/s]	`Double (F64)`
`ActualTorque`	Skutečný moment/síla osy	`Double (F64)`
`LIMN`	Koncový spínač v záporném směru	`Bool`
`LIMZ`	Koncový spínač pro nulovou polohu	`Bool`
`LIMP`	Koncový spínač v kladném směru	`Bool`
`StatusWord`	Stavové slovo servoměniče dle CiA402 standardu	`Long (I32)`
`FAULT`	Vynucení vypnutí osy (obvykle chyba měniče nebo komunikace s měničem)	`Bool`

Výstupy

`yAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`Position`	Požadovaná (interpolovaná) poloha [drive unit]	`Double (F64)`
`Velocity`	Požadovaná (interpolovaná) rychlost [drive unit/s]	`Double (F64)`
`Acceleration`	Požadované (interpolované) zrychlení [drive unit/s/s]	`Double (F64)`
`Torque`	Požadovaný (interpolovaný) moment/síla	`Double (F64)`
`State`	Stav interpolátoru (záporné hodnoty znamenají chybu a pokud to není nesprávnou periodou, je potřeba zkontrolovat časování obou úloh) `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`)	`Long (I32)`
`ControlWord`	Řídící slovo servoměniče dle CiA402 standardu	`Long (I32)`

Parametry

`InterpolationMode` Algoritmus pro interpolaci ⊙2 `Long (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 aproximation (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 aproximation (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

`ReverseLimit` Negace významu vstupů LIMN, LIMZ a LIMP `Bool`

`InterpolationMode` Režim řízení servoměniče ⊙2 `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)

`DriveTimeout` Čas pro odpověď servoměniče [s] (for Strict CiA402 mode only) `Double (F64)`

`DriveBits` Počet platných bitů (záporné číslo znamená číslo se znaménkem) v polohovém vstupu `ActualPosition` a výstupu `Position` `Long (I32)`

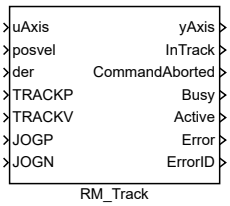
↓-64 ↑63 ⊙-32

`DriveUnits`	Vzdálenost v jednotkách měniče (pro transformaci polohy, vzdálenost odpovídající `AxisUnits`)	`Double (F64)`
`AxisUnits`	Vzdálenost v jednotkách osy (pro transformaci polohy, vzdálenost odpovídající `DriveUnits`)	`Double (F64)`
`VelocityCalculate`	když je zaškrtnuto, vstup `ActualVelocity` se neuplatňuje a rychlost je dopočítávána z rozdílu polohy (v aktuálním a předchozím kroku)	`Bool`

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

`uAxis`	Odkaz na osu (přípustné je jen spojení `RM_Axis.axisRef–uAxis` nebo `yAxis–uAxis`)	`Reference`
`posvel`	Požadovaná poloha nebo rychlost [unit, unit/s]	`Double (F64)`
`TRACKP`	Zapnutí režimu sledování polohy	`Bool`
`TRACKV`	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^2]		Double (F64)
pd	Maximální povolené zpomalení [unit/s^2]		Double (F64)
pj	Maximální povolená změna zrychlení [unit/s^3]		Double (F64)
iLen	Počet kroků pro odhad rychlosti	⊙10	Long (I32)

Kapitola 21

MC_MULTI – Řízení pohybu více os

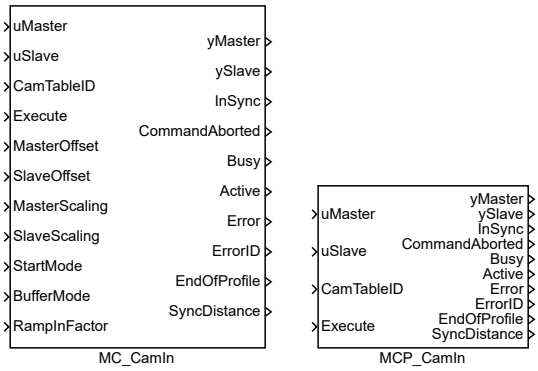
Obsah

Tato kategorie bloků zahrnuje bloky pro synchronizovaný pohyb více os, jak jsou definovány ve specifikaci PLCopen. Jde zejména o tzv. elektronické vačky a elektronické převodovky. Pro bloky této kategorie platí stejné obecné zásady, jaké byly uvedeny v kapitole 20 (knihovna MC_SINGLE, bloky pro řízení jedné osy).

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):

$$PosS = Cam((PosM - \texttt{MasterOffset})/\texttt{MasterScaling}) * \texttt{SlaveScaling} + \texttt{SlaveOffset}$$

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

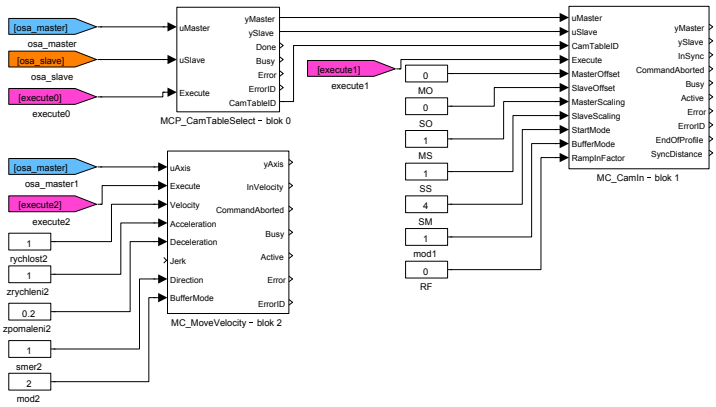
`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`CamTableID`	Odkaz na vačku (spojit s `MCP_CamTableSelect.CamTableID`)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`MasterOffset`	Posunutí v definici vačky na straně hlavní osy [unit]	`Double (F64)`
`SlaveOffset`	Posunutí v definici vačky na straně podřízené osy [unit]	`Double (F64)`
`MasterScaling`	Měřítko pro definici vačky (strana hlavní osy)	`Double (F64)`
`SlaveScaling`	Měřítko pro definici vačky (strana podřízené osy)	`Double (F64)`
`StartMode`	Volba absolutního nebo relativního profilu vačky `1` ..... hlavní osa relativní podřízená osa absolutní `2` ..... hlavní osa absolutní podřízená osa relativní `3` ..... obě osy relativní `4` ..... obě osy absolutní	`Long (I32)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`RampIn`	RampIn faktor (0 = RampIn režim se nepoužívá); odpovídá přibližně rychlosti synchronizace [unit/s] podřízené osy na polohu vačky	`Double (F64)`

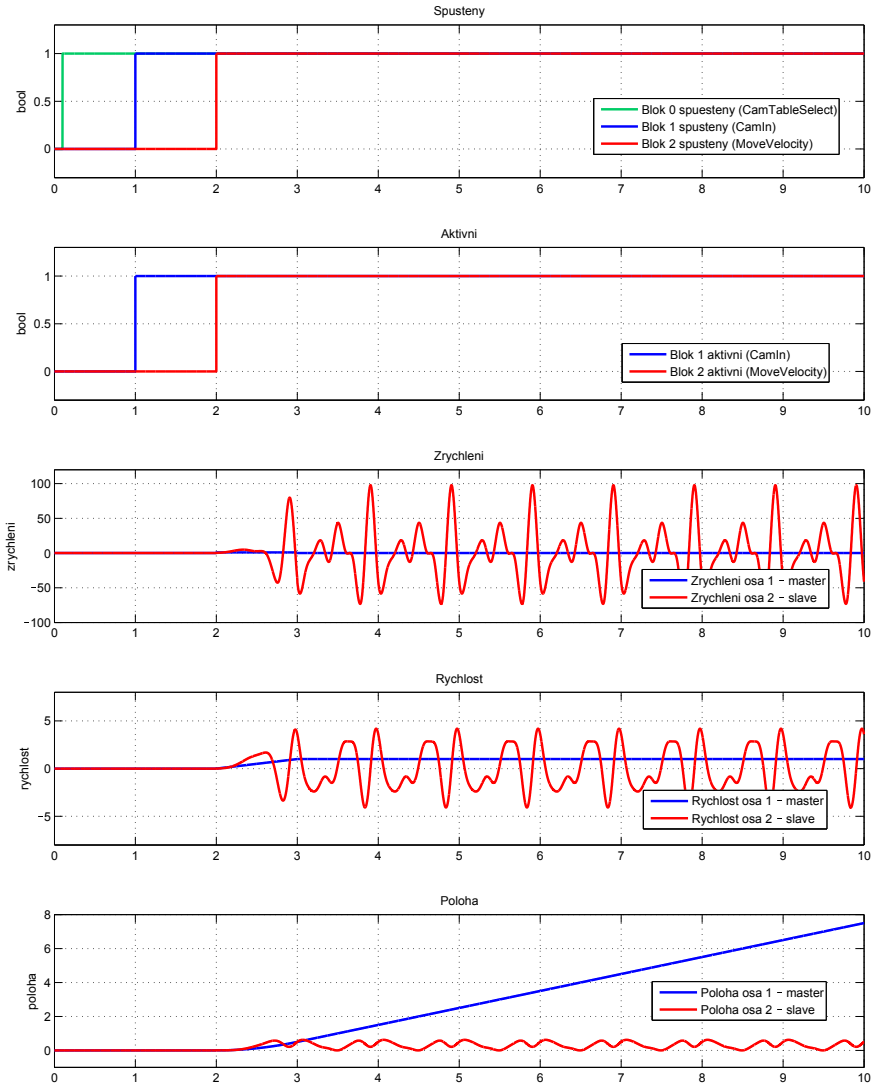
Výstupy

`yMaster`	Odkaz na 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` ..... 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

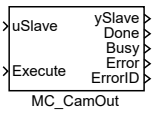



Spusteny
bool
Blok 0 spuesteny (CamTableSelect)
Blok 1 spusteny (CamIn)
Blok 2 spusteny (MoveVelocity)
Aktivni
bool
Blok 1 aktivni (CamIn)
Blok 2 aktivni (MoveVelocity)
Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave
Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave
Poloha
poloha
Poloha osa 1 - master
Poloha osa 2 - slave

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).

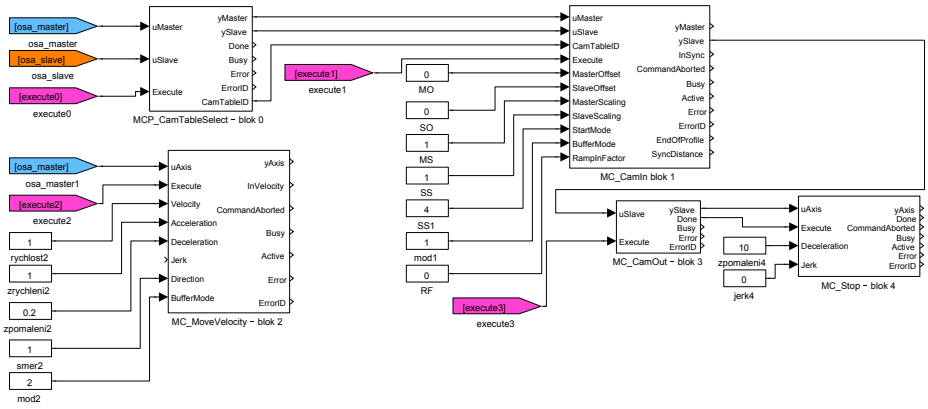
Vstupy

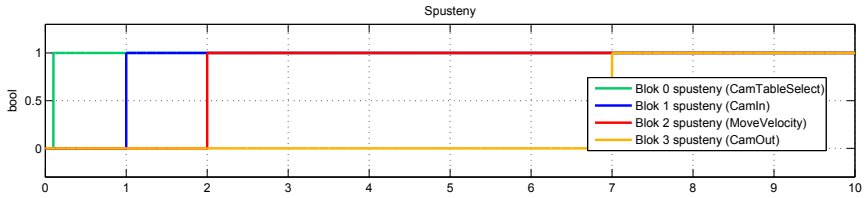
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

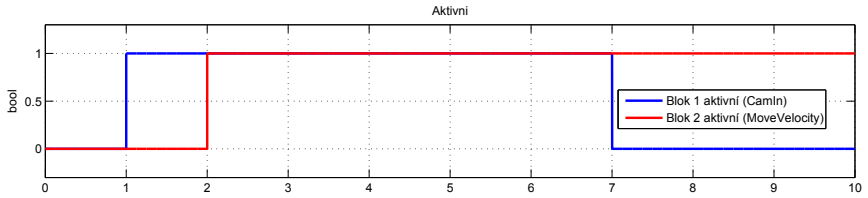
Výstupy

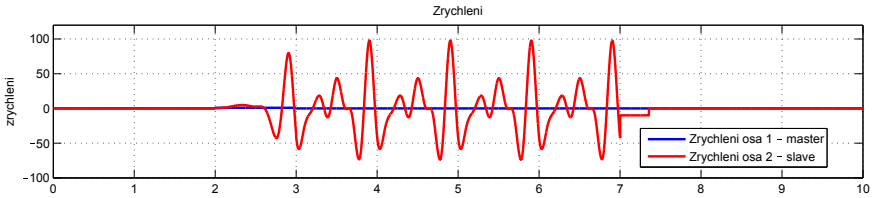
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	

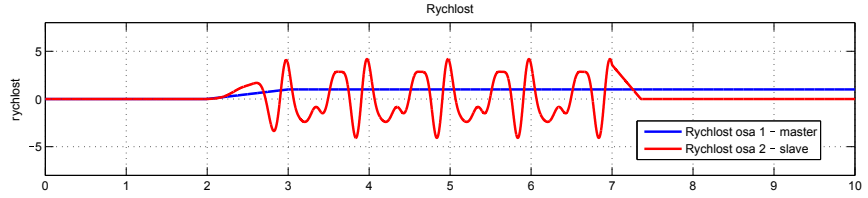
Příklad

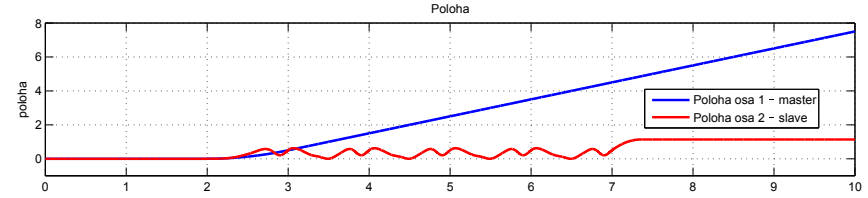

[osa_master]
osa_master
[osa_slave]
osa_slave
[execute0]
execute0
uMaster
uSlave
Execute
yMaster
ySlave
Done
Busy
Error
ErrorID
CamTableID
MCP_CamTableSelect - blok 0
[execute1]
execute1
0
MO
0
SO
1
MS
1
SS
4
SS1
1
mod1
0
RF
uMaster
uSlave
CamTableID
Execute
MasterOffset
SlaveOffset
MasterScaling
SlaveScaling
StartMode
BufferMode
RampInFactor
yMaster
ySlave
InSync
CommandAborted
Busy
Active
Error
ErrorID
EndOfProfile
SyncDistance
MC_CamIn blok 1
[osa_master]
osa_master1
[execute2]
execute2
1
rychlost2
1
zrychleni2
0.2
zpomaleni2
1
smer2
2
mod2
uAxis
Execute
Velocity
Acceleration
Deceleration
Jerk
Direction
BufferMode
yAxis
InVelocity
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveVelocity - blok 2
[execute3]
execute3
uSlave
Execute
ySlave
Done
Busy
Error
ErrorID
MC_CamOut - blok 3
10
zpomaleni4
0
jerk4
uAxis
Execute
Deceleration
Jerk
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_Stop - blok 4


Spusteny
bool
Blok 0 spusteny (CamTableSelect)
Blok 1 spusteny (CamIn)
Blok 2 spusteny (MoveVelocity)
Blok 3 spusteny (CamOut)


Aktivni
bool
Blok 1 aktivní (CamIn)
Blok 2 aktivní (MoveVelocity)


Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave

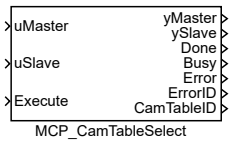

Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave


Poloha
poloha
Poloha osa 1 - master
Poloha osa 2 - slave

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 zvoloit, 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

`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	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 `MC_CamIn.CamTableID`) `i` ..... obecná chyba systému REXYGEN	`Reference`

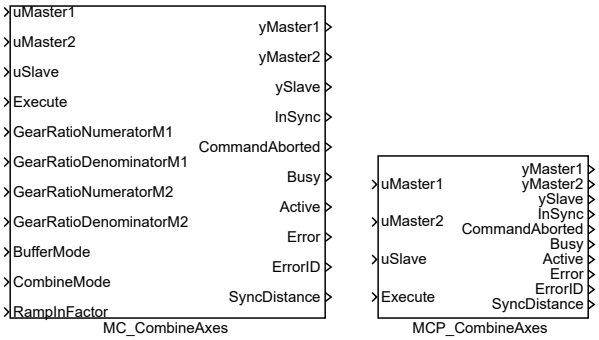
Parametry

`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 `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)	`Long (I32)`
`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)`

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

$$\texttt{SlavePosition} = \texttt{Master1Position} \cdot \frac{\texttt{GearRatioNumeratorM1}}{\texttt{GearRatioDenominatorM1}} + \\ + \texttt{Master2Position} \cdot \frac{\texttt{GearRatioNumeratorM2}}{\texttt{GearRatioDenominatorM2}}$$

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

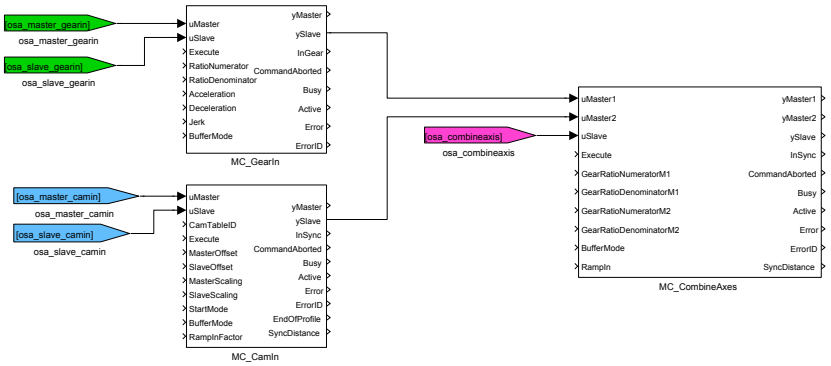
`uMaster1`	Odkaz na první hlavní osu	`Reference`
`uMaster2`	Odkaz na druhou hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`GearRatioNumeratorM1`	Čitatel převodového poměru master osy 1	`Long (I32)`
`GearRatioDenominatorM1`	Jmenovatel převodového poměru master osy 1	`Long (I32)`
`GearRatioNumeratorM2`	Čitatel převodového poměru master osy 2	`Long (I32)`

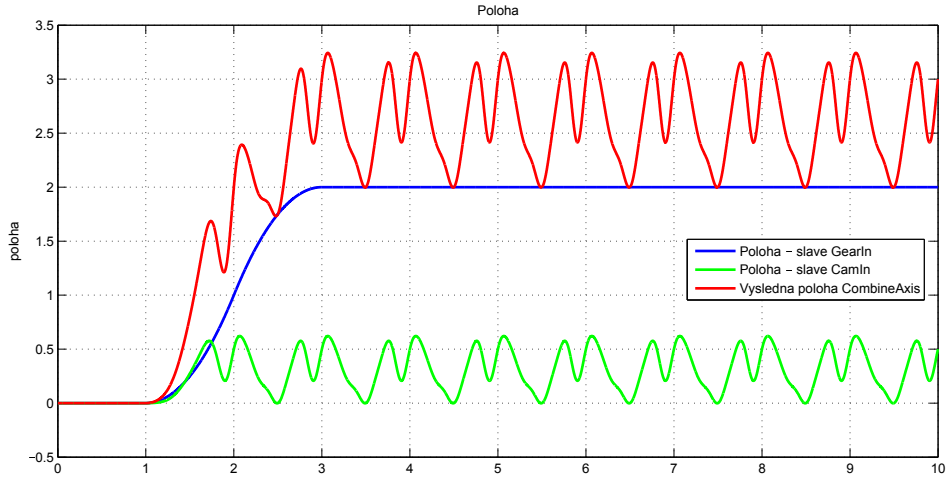
`GearRatioDenominatorM2`	Jmenovatel převodového poměru master osy 2	`Long (I32)`
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní 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)	`Long (I32)`
`CombineMode`	Režim sloučení os ⊙`1` `1` ..... součet `2` ..... rozdíl	`Long (I32)`
`RampInFactor`	RampIn faktor (0 = RampIn režim se nepoužívá); rychlost synchronizace [unit/s] podřízené osy na polohu vačky odpovídá přibližně `RampIn * MaxVelocityAppl`, kde `MaxVelocityAppl` se zadává v bloku `RM_Axis` připojeného ke vstupu `uSlave`; analogicky zrychlení	`Double (F64)`

Výstupy

`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` ..... obecná chyba systému REXYGEN	`Error`
`SyncDistance`	Odchylka v poloze podřízené osy od synchronizované polohy	`Double (F64)`

Příklad

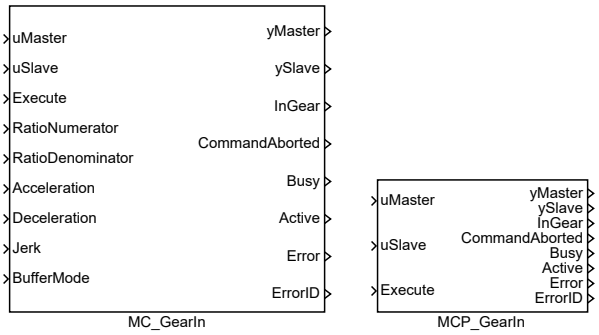

[osa_master_gearin]
osa_master_gearin
[osa_slave_gearin]
osa_slave_gearin
uMaster
uSlave
Execute
RatioNumerator
RatioDenominator
Acceleration
Deceleration
Jerk
BufferMode
yMaster
ySlave
InGear
CommandAborted
Busy
Active
Error
ErrorID
MC_GearIn
[osa_master_camin]
osa_master_camin
[osa_slave_camin]
osa_slave_camin
CamTableID
MasterOffset
SlaveOffset
MasterScaling
SlaveScaling
StartMode
RampInFactor
InSync
EndOfProfile
SyncDistance
MC_CamIn
[osa_combineaxis]
osa_combineaxis
uMaster1
uMaster2
GearRatioNumeratorM1
GearRatioDenominatorM1
GearRatioNumeratorM2
GearRatioDenominatorM2
RampIn
yMaster1
yMaster2
MC_CombineAxes


Poloha
poloha
Poloha – slave GearIn
Poloha – slave CamIn
Vysledna poloha CombineAxis
3.5
3
2.5
2
1.5
1
0.5
0
−0.5
0
1
2
3
4
5
6
7
8
9
10

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):

$$VelS = VelM \cdot \frac{\texttt{RatioNumerator}}{\texttt{RatioDenominator}}$$

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 spojité, tj. `jerk<>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.

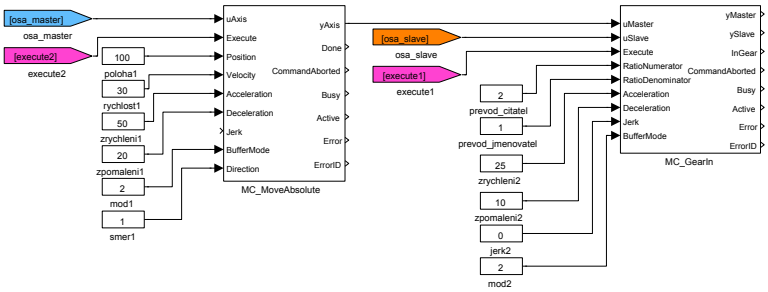
Vstupy

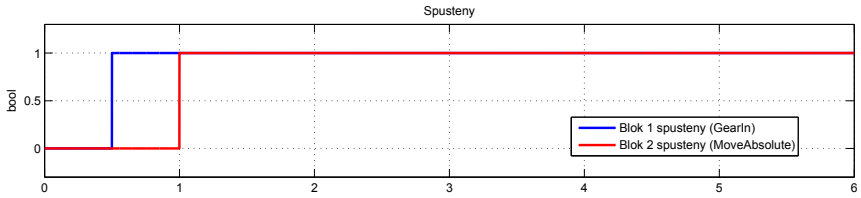
`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`RatioNumerator`	Převodový poměr - čitatel (podřízená osa)	`Long (I32)`
`RatioDenominator`	Převodový poměr - jmenovatel (hlavní osa)	`Long (I32)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	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)	

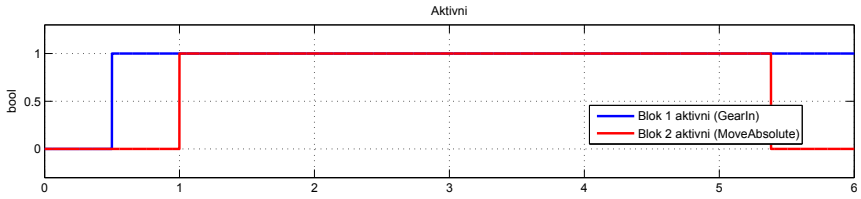
Výstupy

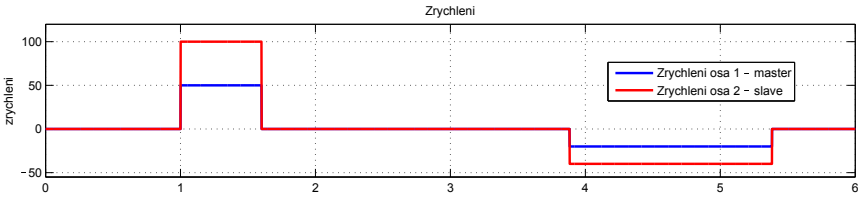
`yMaster`	Odkaz na hlavní osu	`Reference`
`ySlave`	Odkaz na podřízenou osu	`Reference`
`InGear`	Příznak dosažení požadované rychlosti řízenou osou	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace	`Error`
	`i` ..... obecná chyba systému REXYGEN	

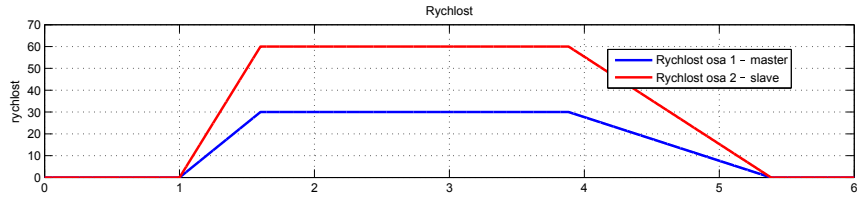
Příklad

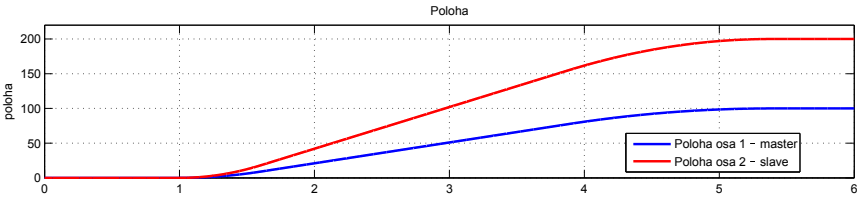

[osa_master]
osa_master
[execute2]
execute2
100
poloha1
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
1
smer1
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute
[osa_slave]
osa_slave
[execute1]
execute1
2
prevod_citatel
1
prevod_jmenovatel
25
zrychleni2
10
zpomaleni2
0
jerk2
2
mod2
uMaster
uSlave
Execute
RatioNumerator
RatioDenominator
Acceleration
Deceleration
Jerk
BufferMode
yMaster
ySlave
InGear
CommandAborted
Busy
Active
Error
ErrorID
MC_GearIn


Spusteny
bool
Blok 1 spusteny (GearIn)
Blok 2 spusteny (MoveAbsolute)


Aktivni
bool
Blok 1 aktivni (GearIn)
Blok 2 aktivni (MoveAbsolute)


Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave

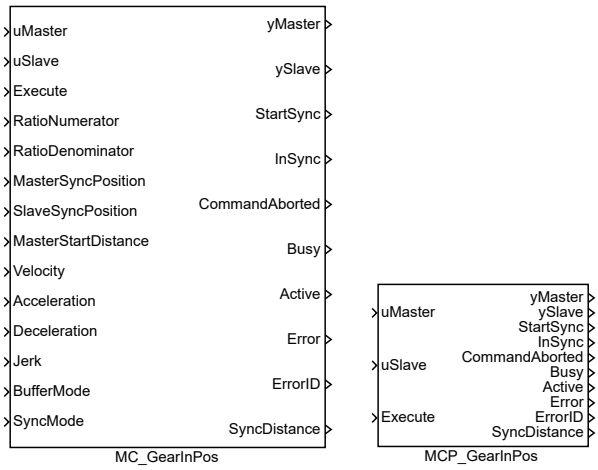

Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave


Poloha
poloha
Poloha osa 1 - master
Poloha osa 2 - slave

MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici

Symboly 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 zachováván i poměr vzdáleností a je určena i poloha obou os v okamžiku zasynchronizování, tzn. platí

$$\frac{SlavePosition - SlaveSyncPosition}{MasterPosition - MasterSyncPosition} = \frac{\texttt{RatioNumerator}}{\texttt{RatioDenominator}}$$

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 rychlosti, 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

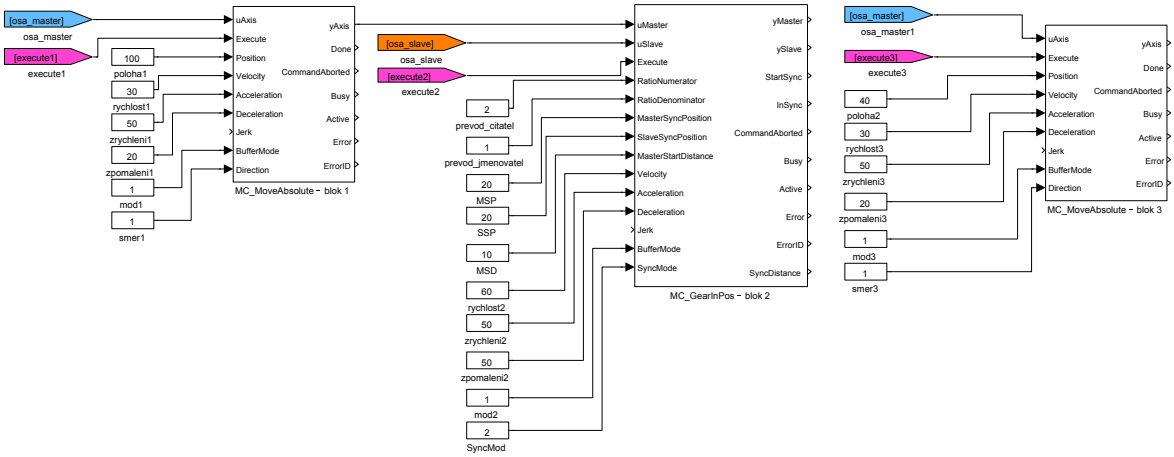
`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`RatioNumerator`	Převodový poměr - čitatel (podřízená osa)	`Long (I32)`
`RatioDenominator`	Převodový poměr - jmenovatel (hlavní osa)	`Long (I32)`
`MasterSyncPosition`	Poloha 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^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`

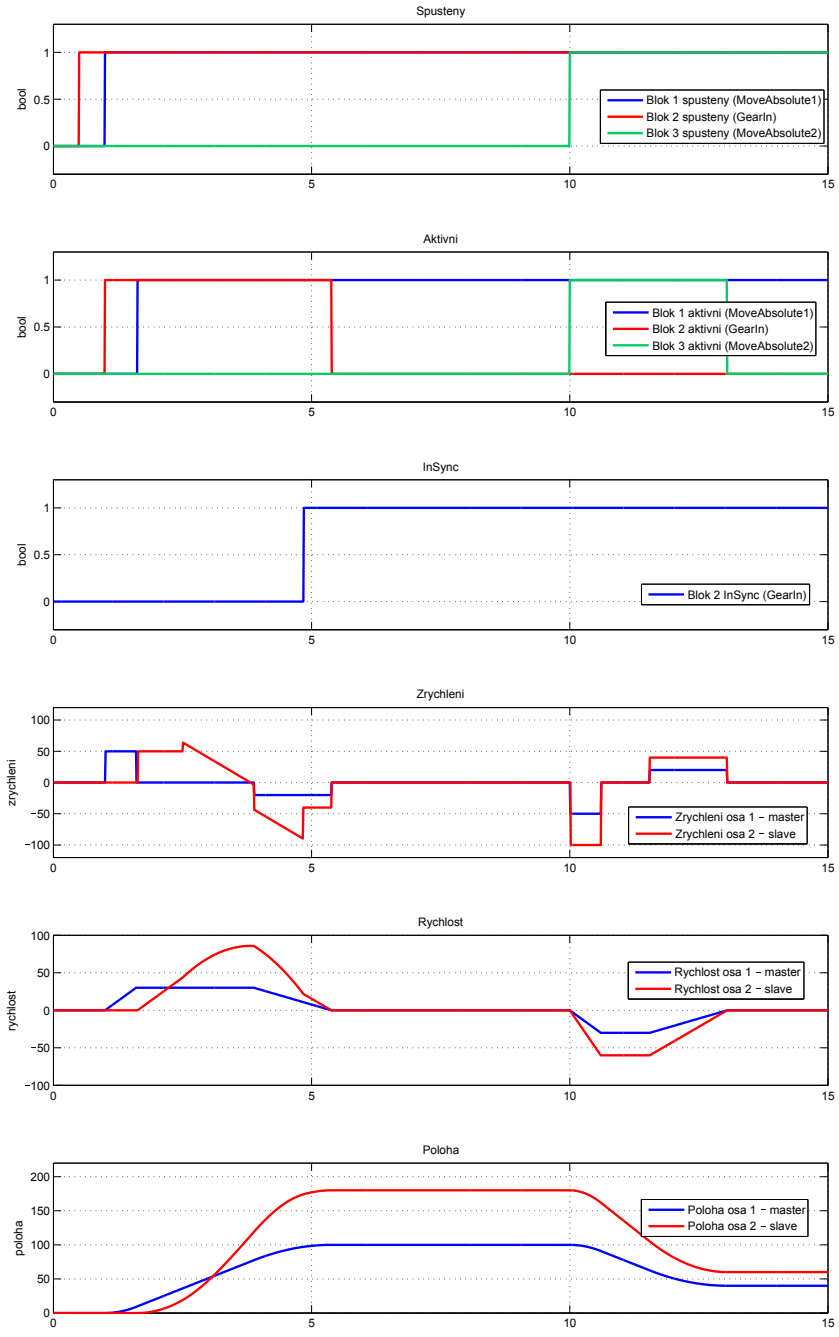
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	`Long (I32)`
`SyncMode`	Režim synchronizace (pouze cyklické osy) `1` ..... synchronizovat zrychlením `2` ..... synchronizovat na nejbližší bod `3` ..... synchronizovat zpomalením	`Long (I32)`

Výstupy

`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 `i` ..... obecná chyba systému REXYGEN	`Error`
`SyncDistance`	Odchylka v poloze podřízené osy od synchronizované polohy [unit/s]	`Double (F64)`

Příklad

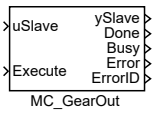

[osa_master]
osa_master
[execute1]
execute1
100
poloha1
30
rychlost1
50
zrychleni1
20
zpomaleni1
1
mod1
1
smer1
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 1
[osa_slave]
osa_slave
[execute2]
execute2
2
prevod_citatel
1
prevod_jmenovatel
20
MSP
20
SSP
10
MSD
60
rychlost2
50
zrychleni2
50
zpomaleni2
1
mod2
2
SyncMod
uMaster
uSlave
Execute
RatioNumerator
RatioDenominator
MasterSyncPosition
SlaveSyncPosition
MasterStartDistance
Velocity
Acceleration
Deceleration
Jerk
BufferMode
SyncMode
yMaster
ySlave
StartSync
InSync
CommandAborted
Busy
Active
Error
ErrorID
SyncDistance
MC_GearInPos - blok 2
[osa_master]
osa_master1
[execute3]
execute3
40
poloha2
30
rychlost3
50
zrychleni3
20
zpomaleni3
1
mod3
1
smer3
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 3


Spusteny
bool
Blok 1 spusteny (MoveAbsolute1)
Blok 2 spusteny (GearIn)
Blok 3 spusteny (MoveAbsolute2)
Aktivni
bool
Blok 1 aktivni (MoveAbsolute1)
Blok 2 aktivni (GearIn)
Blok 3 aktivni (MoveAbsolute2)
InSync
bool
Blok 2 InSync (GearIn)
Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave
Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave
Poloha
poloha
Poloha osa 1 - master
Poloha osa 2 - slave

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).

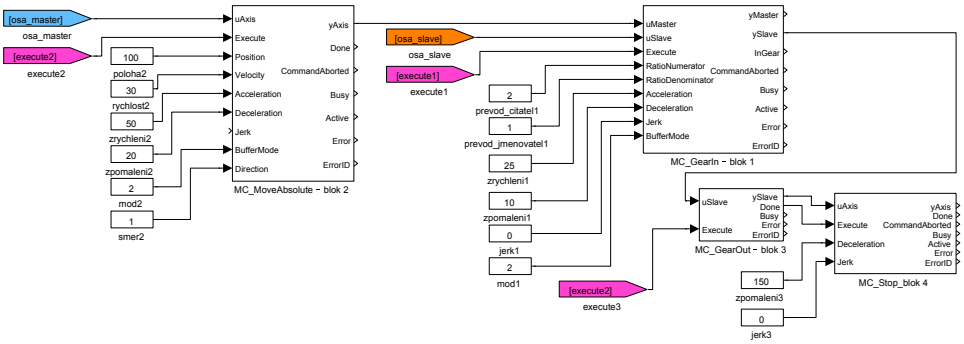
Vstupy

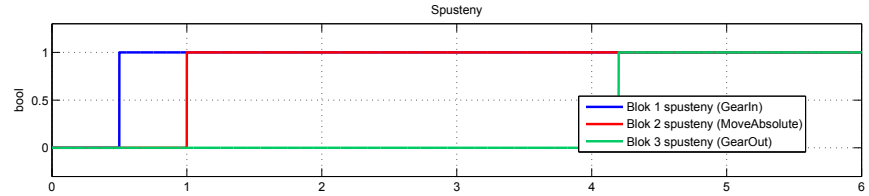
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

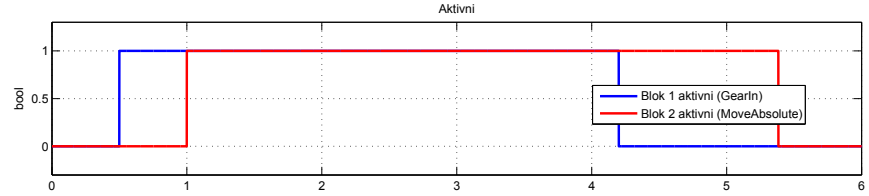
Výstupy

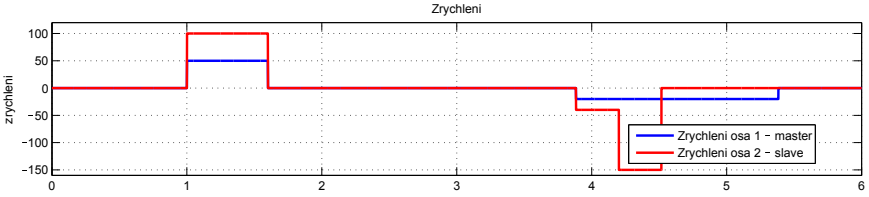
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

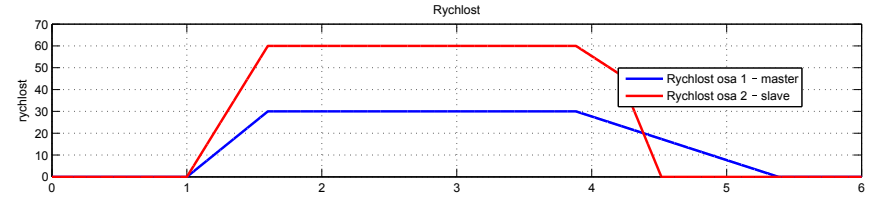
Příklad

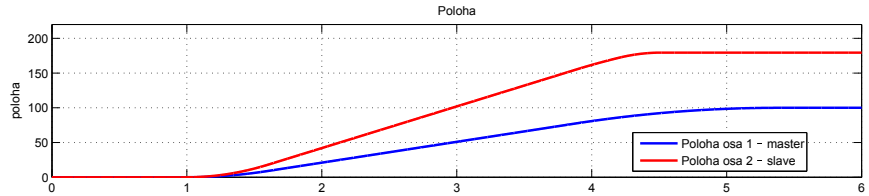

[osa_master]
osa_master
[execute2]
execute2
100
poloha2
30
rychlost2
50
zrychleni2
20
zpomaleni2
2
mod2
1
smer2
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 2
[osa_slave]
osa_slave
[execute1]
execute1
2
prevod_citatel1
1
prevod_jmenovatel1
25
zrychleni1
10
zpomaleni1
0
jerk1
2
mod1
uMaster
uSlave
Execute
RatioNumerator
RatioDenominator
Acceleration
Deceleration
Jerk
BufferMode
yMaster
ySlave
InGear
CommandAborted
Busy
Active
Error
ErrorID
MC_GearIn - blok 1
[execute2]
execute3
uSlave
Execute
ySlave
Done
Busy
Error
ErrorID
MC_GearOut - blok 3
150
zpomaleni3
0
jerk3
uAxis
Execute
Deceleration
Jerk
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_Stop_blok 4


Spusteny
bool
Blok 1 spusteny (GearIn)
Blok 2 spusteny (MoveAbsolute)
Blok 3 spusteny (GearOut)


Aktivni
bool
Blok 1 aktivni (GearIn)
Blok 2 aktivni (MoveAbsolute)


Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave

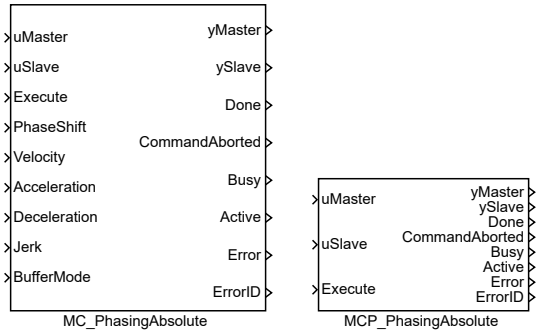

Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave


Poloha
poloha
Poloha osa 1 - master
Poloha osa 2 - slave

MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)

Symboly 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

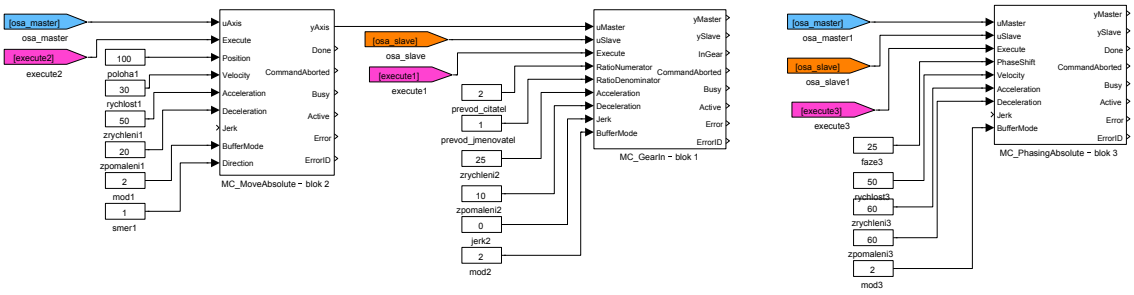
`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`PhaseShift`	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	`Double (F64)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`

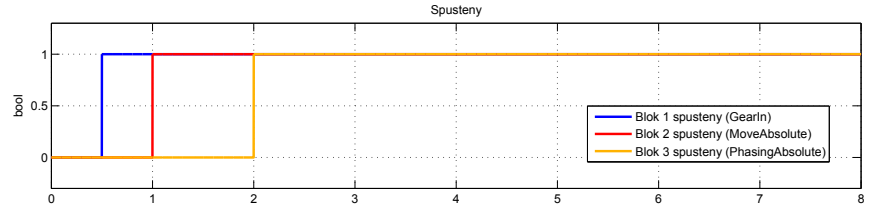
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `1` ..... aborting `2` ..... buffered	`Long (I32)`

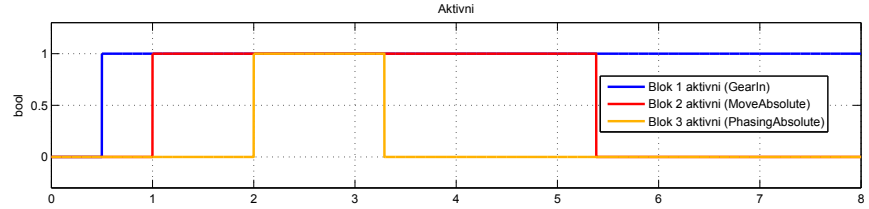
Výstupy

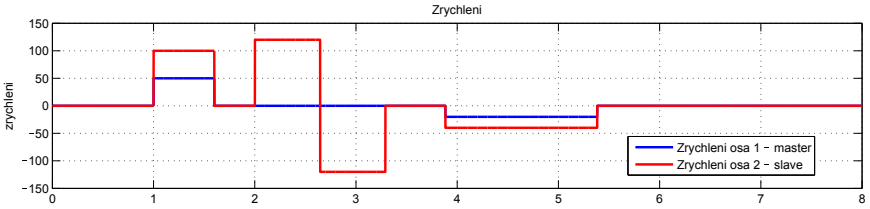
`yMaster`	Odkaz na hlavní osu	`Reference`
`ySlave`	Odkaz na podřízenou osu	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

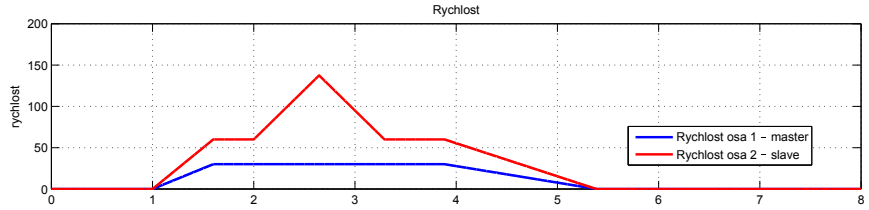
Příklad

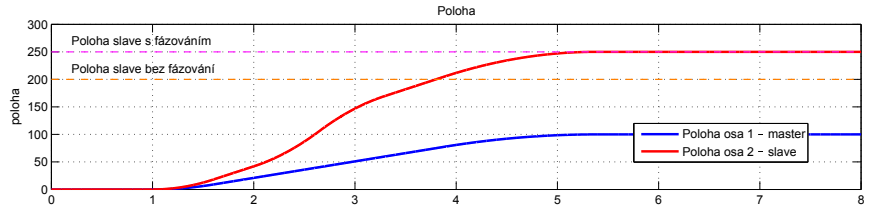

[osa_master]
osa_master
[execute2]
execute2
100
poloha1
30
rychlost1
50
zrychleni1
20
zpomaleni1
2
mod1
1
smer1
uAxis
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Direction
yAxis
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveAbsolute - blok 2
[osa_slave]
osa_slave
[execute1]
execute1
2
prevod_citatel
1
prevod_jmenovatel
25
zrychleni2
10
zpomaleni2
0
jerk2
2
mod2
uMaster
uSlave
Execute
RatioNumerator
RatioDenominator
Acceleration
Deceleration
Jerk
BufferMode
yMaster
ySlave
InGear
CommandAborted
Busy
Active
Error
ErrorID
MC_GearIn - blok 1
[osa_master]
osa_master1
[osa_slave]
osa_slave1
[execute3]
execute3
25
faze3
50
rychlost3
60
zrychleni3
60
zpomaleni3
2
mod3
uMaster
uSlave
Execute
PhaseShift
Velocity
Acceleration
Deceleration
Jerk
BufferMode
yMaster
ySlave
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_PhasingAbsolute - blok 3


Spusteny
bool
Blok 1 spusteny (GearIn)
Blok 2 spusteny (MoveAbsolute)
Blok 3 spusteny (PhasingAbsolute)


Aktivni
bool
Blok 1 aktivni (GearIn)
Blok 2 aktivni (MoveAbsolute)
Blok 3 aktivni (PhasingAbsolute)


Zrychleni
zrychleni
Zrychleni osa 1 - master
Zrychleni osa 2 - slave

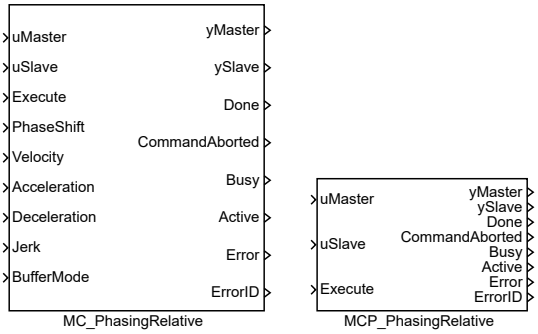

Rychlost
rychlost
Rychlost osa 1 - master
Rychlost osa 2 - slave


Poloha
poloha
Poloha slave s fázováním
Poloha slave bez fázování
Poloha osa 1 - master
Poloha osa 2 - slave

MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)

Symboly 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

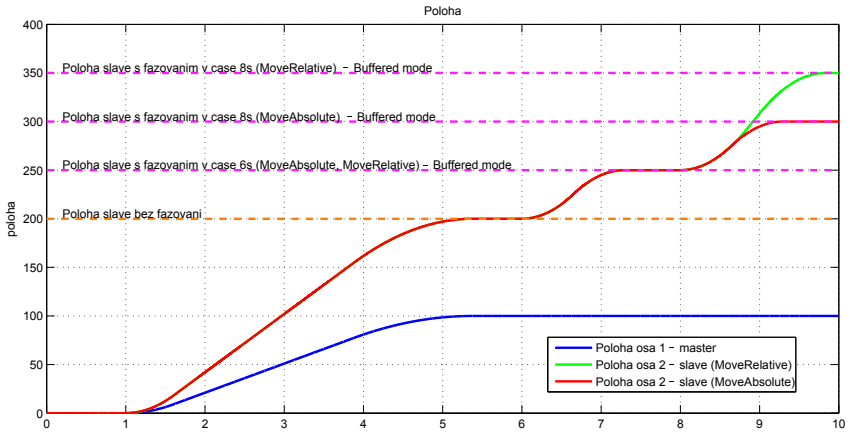
`uMaster`	Odkaz na hlavní osu	`Reference`
`uSlave`	Odkaz na podřízenou osu	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`PhaseShift`	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	`Double (F64)`

`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`BufferMode`	Režim převzetí osy `1` ..... aborting `2` ..... buffered	`Long (I32)`

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 `i` ..... obecná chyba systému REXYGEN	`Error`

Příklad



Kapitola 22

MC_COORD – Koordinované řízení pohybu

Obsah

Tato kategorie bloků zahrnuje bloky pro koordinovaný pohyb více os, jak jsou definovány ve specifikaci PLCopen. Pro bloky této kategorie platí stejné obecné zásady, jaké byly uvedeny v kategorii MC_SINGLE (bloky pro řízení pohybu v jedné ose). Dále platí, že sdílená struktura `AXES_GROUP_REF` popisovaná v PLCopen je reprezentována blokem RM_AxesGroup.

Pohybové bloky jsou opět ve variantě s parametry na vstupech a v parametrech, tj. s prefixem `MC_` a `MCP_`. Protože nelze dopředu říci, kolik bude souřadnic (je to parametr bloku RM_AxesGroup), musí být údaje o poloze vektorové parametry, respektive vstupy (předpokládá to i norma PLCopen). Pro variantu `MC_` se na příslušný vstup připojí blok RTOV (popřípadě řetězec těchto bloků, pokud je vektor delší), který umožňuje složit vektor ze skalárních vstupů. Pro výstupní vektory se analogicky použije blok VTOR. Oba tyto bloky se nacházejí v knihovně MATRIX. V případě `MCP_` varianty se příslušný vektor zapisuje přímo do bloku jako jeho parametr.

Rychlost, zrychlení, zpomalení a jerk se zadává skalárně (jedno číslo, nikoliv vektor) a je to vždy ve smyslu tečném k výsledné trajektorii. Hodnota se kontroluje jen při spuštění bloku (zda není větší, než zadané maximum v bloku RM_AxesGroup), takže v některých případech může být rychlost vyšší. Tyto limity se zadávají buď v absolutních jednotkách nebo relativně k parametrům v bloku `RM_AxesGroup`. Řídí se to hodnotou parametru `LimitMode`.

Transformace a souřadné systémy

Transformace mezi polohou v kartézských souřadnicích a polohou motorů je obecně nelineární a není tak jednoduše možné určit, zda naplánovaná trajektorie nepřekračuje limity na rychlost a zrychlení v jednotlivých osách (motorech). Proto se provede vygenerování

trajektorie dle zadaných parametrů a její spuštění. Pokud dojde k překročení limitu polohy, rychlosti nebo zrychlení v ose, skupina přejde do chybového stavu a spustí se zastavovací sekvence po původní trajektorii. Pokud přesto dojde k překročení systémového limitu polohy, rychlosti nebo zrychlení na některé z os, jsou všechny osy skupiny brzděny s maximální možnou intenzitou samostatně a tedy již ne po požadované trajektorii.

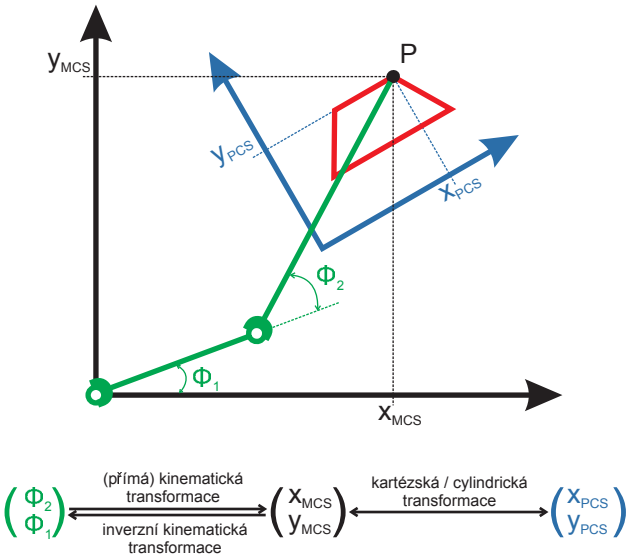
Norma PLCopen dále rozlišuje tři souřadné systémy. To, jaký souřadný systém se použije (tj. ve kterém je zadávána poloha) určuje parametr `CoordSystem` příslušného pohybového bloku. Možnosti jsou:

- `ACS` (Axis Coordinate System) – V tomto souřadném systému jsou jednotlivé souřadnice přímo polohy jednotlivých motorů/připojených os.

- `MCS` (Machine Coordinate System) – Pravoúhlý souřadný systém spojený se strojem. Předpokládá se, že tento souřadný systém obsahuje tři (popř. dvě pro planární stroje) na sebe navzájem kolmé osy pro určení polohy (tzv. kartézký souřadný systém; obvykle se používá pravotočivý) a dále až 3 souřadnice pro určení natočení koncového efektoru stroje. Pro reprezentaci směru/úhlu natočení se obvykle používá směrový vektor (tři souřadnice polohy, přičemž směr od bodu [0, 0, 0] k zadanému bodu určuje směr koncového efektoru) nebo tzv. eulerovy úhly (obdoba zeměpisné šířky, zeměpisné délky a azimutu na zeměkouli). Vazbu mezi MCS a ACS určuje kinematická transformace (tj. některý z bloků `MC_SetKinTransform_xxx`), která tím pak také určuje jak je MCS definován.

- `PCS` (Product Coordinate System) – Pravoúhlý souřadný systém spojený s výrobkem. Je to vlastně posunutý a otočený MCS. Někdy se PCS oproti MCS pohybuje. Vazbu mezi MCS a PCS zajišťuje (pro statický případ) blok `MC_SetCartesianTransform`

Níže uvedený obrázek ukazuje reprezentaci bodu P (ležícím na obrobku – červený lichoběžník) v různých souřadných systémech. V PCS je dán polohou $P_{PCS} = (x_{PCS}, y_{PCS})$. Translací a rotací lze převést do MCS jako $P_{MCS} = (x_{MCS}, y_{MCS})$. Nakonec může být popsán v ACS jako dvojice úhlů natočení os $P_{ACS} = (\phi_1, \phi_2)$



Pokud stroj nepracuje jen s polohou, ale i orientací/natočením, pro reprezentaci lze použít:

- `eulerovy úhly` – Obdoba souřadnic na zeměkouli (tj. zeměpisná délka, zeměpisná šířka, azimut). Geometricky to představuje otočení souřadného systému podle osy Z o 1. úhel, potom podle nové osy Y o 2. úhel a nakonec podle nové osy X o 3. úhel. Výhoda je, že pro reprezentaci máme pouze 3 nezávislá čísla (ostatní reprezentace vyžadují více čísel, která mezi sebou mají nějakou vazbu). Nevýhoda je nejednoznačnost representace (zejména „na pólu“, ale někdy také vadí, že 3615^o je totéž co 15^o).

- `kvaternion` – Obdoba komplexního čísla, ale má celkem 4 složky (1 reálnou a 3 imaginární). Pro vyjádření orientace se používají jen kvaterniony velikosti 1, tj. $q_1^2+q_2^2+q_3^2+q_4^2 = 1$. Lze ukázat, že libovolnou orientaci lze získat jedním otočením z výchozí polohy o určitý úhel α okolo určité osy definované jednotkovým směrovým vektorem $o = (o1, o2, o3)$. Geometrický význam kvaternionu je právě toto otočení ze základní polohy, přičemž $q = (cos\frac{\alpha)}{2}, o1 * sin\frac{\alpha)}{2}, o2 * sin\frac{\alpha)}{2}, o3 * sin\frac{\alpha)}{2})$. Tato reprezentace se vždy používá interně. Výhoda je, že skládání otočení je násobení kvaternionu.

- `Rodriguez vektor` – Vzhledem k tomu, že kvaternion je normován (jinak by to nebyla jednoznačná reprezentace), lze 1. složku vždy dopočítat ze zbylých 3, což je právě tato reprezentace. Je definována jako $r = (o1 * \alpha, o2 * \alpha, o3 * \alpha)$, přičemž některé implementace zadávají úhel ve stupních a jiné v radiánech.

- `matice rotace` – Matice rozměru 3x3, kde sloupce představují po řadě jednotkové vektory ve směru otočené osy X, otočené osy Y, otočené osy Z. Matice rotace se interně používá pro některé výpočty, protože skládání transformací představuje

násobení matic. Nevýhoda je samozřejmě redundance zápisu a možnost vzniku nekonzistence.

Druhy pohybů

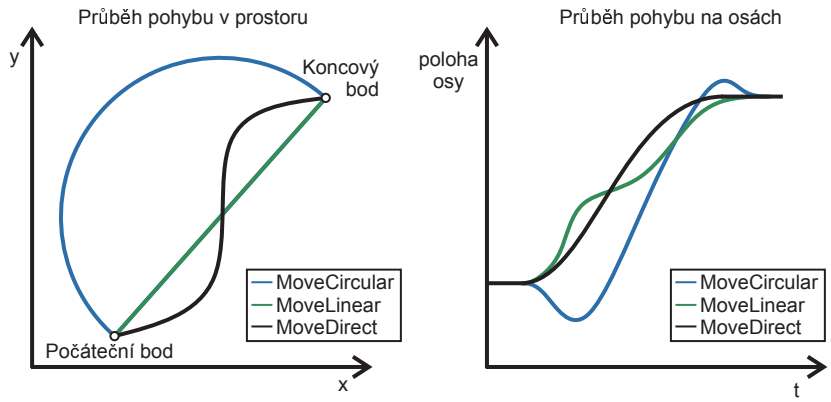
Generované pohyby lze rozdělit na dva základní druhy:

Pohyb z bodu do bodu (Point - to - Point movements) – Při tomto pohybu se chceme dostat do zadané pozice co nejrychleji. To se provede tak, že se pomocí inverzní kinematiky spočte požadované natočení pohonu, které je nutné pro dosažení zadané pozice. Každý pohon se pak bude do této pozice řídit nezávisle na ostatních a to s maximální rychlostí a zrychlením nastavenými jako limity na osách. Pokud se budou takto pohybovat všechny osy, tak výsledná trajektorie mezi počátečním a koncovým bodem v MCS není definována. Je tedy nutné dát pozor, jestli je bezpečné tento blok použít. Použitelné bloky pro tento pohyb jsou:

- MC_MoveDirectAbsolute
- MC_MoveDirectRelative

Pohyb po dané trajektorii (Cartesian Path Movement) – Podporované trajektorie jsou pohyb po přímce, po kružnici a po spline křivce. Trajektorie je vygenerována obecně v prostoru a následně převedena inverzní kinematikou do souřadnicového systému pohonů ACS. Takto přepočtenou trajektorii je pak již možné přivést jako požadovanou hodnotu na regulátory pohonů. Použitelné bloky pro tento pohyb jsou:

- MC_MoveLinearAbsolute
- MC_MoveLinearRelative
- MC_MoveCircularAbsolute
- MC_MoveCircularRelative
- MC_MovePath

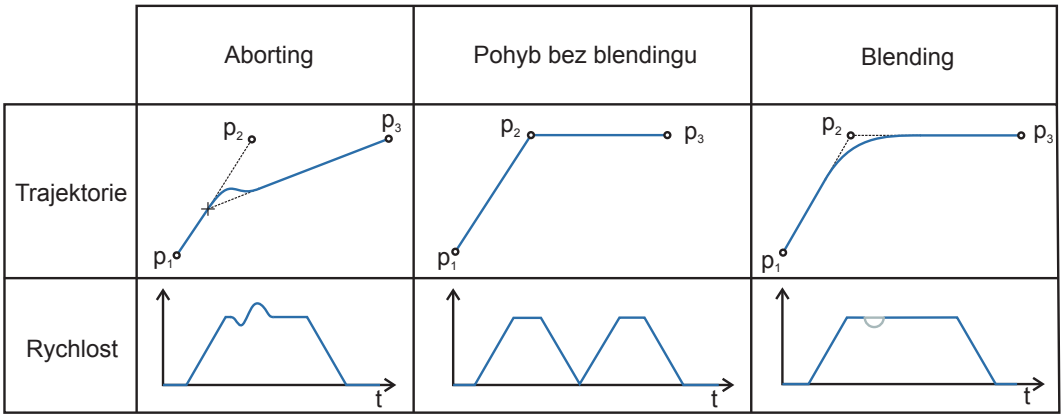


Míchání pohybů

Důležitá část při generování pohybu je navazování po sobě jdoucích pohybových příkazů na skupinu os (blending). Bez zapnutí blending módů provede skupina os vždy pohyb do požadované pozice, kde zastaví přesně na cílovém místě a uvede se do klidu. Následující pohyb se do této doby nespustí. Po zastavení je tedy nutné se opět znovu rozjet. V mnoha aplikacích může být toto neustálé zastavování a rozjíždění nežádoucí a pohyby je nutné nějak prokládat. Blending je vhodné použít například pro:

- Zrychlení výroby.
- Generování hladších trajektorií a tím zmenšení mechanického namáhání.
- Některé aplikace vyžadují pohyby s konstantní rychlostí (např. nanášení lepidla, barvení, svařování, ...)

Všechny výše uvedené příklady lze vyřešit použitím různých druhů blendingů. Společné pro všechny druhy je modifikace původní trasy v hladkou trajektorii bez rohů (zastavování). Na obrázku níže jsou uvedeny příklady některých možných navazování pohybu.



Přehled režimů převzetí osy

Tabulka režimů převzetí osy:

režim	funkce
`1: Aborting`	Start následujícího bloku okamžitě.
`2: Buffered`	Start následujícího bloku ihned po dokončení předcházejícího (dojde k zastavení pohybu).
`3: BlendingLow`	Rychlost je proložena s menší z rychlostí obou bloků.
`4: BlendingHigh`	Rychlost je proložena s vetší z rychlostí obou bloků.
`5: BlendingPrevious`	Rychlost je proložena s rychlostí prvního bloku.
`6: BlendingNext`	Rychlost je proložena s rychlostí druhého bloku.

Přehled režimů míchání pohybu

Tabulka režimů míchání pohybu:

režim	funkce
`1: TMNone`	Nevkládá žádnou přechodovou křivku.
`2: TMStartVelocity`	Přechod s danou počáteční rychlostí.
`3: TMConstantVelocity`	Přechod s danou konstantní rychlostí.
`4: TMCornerDistance`	Přechod s daným corner distance.
`5: TMMaxCornerDeviation`	Přechod s daným maximálním corner deviation.
`11: Milanův režim`	Nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku.

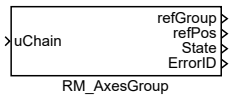
Synchronizace

Pro správnou synchronizaci je nutné, aby se všechny osy (jejich bloky `RM_Axis`) přiřazené do jedné skupiny (k jednomu bloku `RM_AxesGroup`) byly buď před blokem `RM_AxesGroup` nebo až za ním. Aby byla jistota ohledně pořadí zpouštění bloků, jsou bloky `RM_AxesGroup` i `RM_Axis` vybaveny vstupem `uChain`. Tento vstup blok nijak nezpracovává, ale lze tam napojit libovolný signál a tím definovat pořadí zpouštění bloků. To je totiž v systému REXYGENvždy podle toku signálu. Pokud jsou bloky `RM_AxesGroup` a `RM_Axis` v jiném tasku (s jinou periodou) funguje podobný interpolační mechanismus jako mezi bloky `RM_Axis` a `RM_AxisSpline`. Pak pořadí zpouštění nelze zajistit. Konzistence dat je zajištěna jiným mechanismem, ale dochází ke zpoždění mezi generovanou hodnotou a její aplikací do motoru.

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, přepočet 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 `PowwerOff`) a `MC_UngroupAllAxis` (ruší všechna nastavení, tj. odebere osy, odebere kinematickou transformaci, přepne do stavu `PowwerOff`).

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 rzchlosti. 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

zadávání úhlu (např. na úhlové stupně nebo otáčky) a pak se rychlost zadává v těchto jednotkách za sekundu. Stejné pravidlo platí pro zrychlení a jerk, tj. parametry `AccRot` a `JerkRot`.

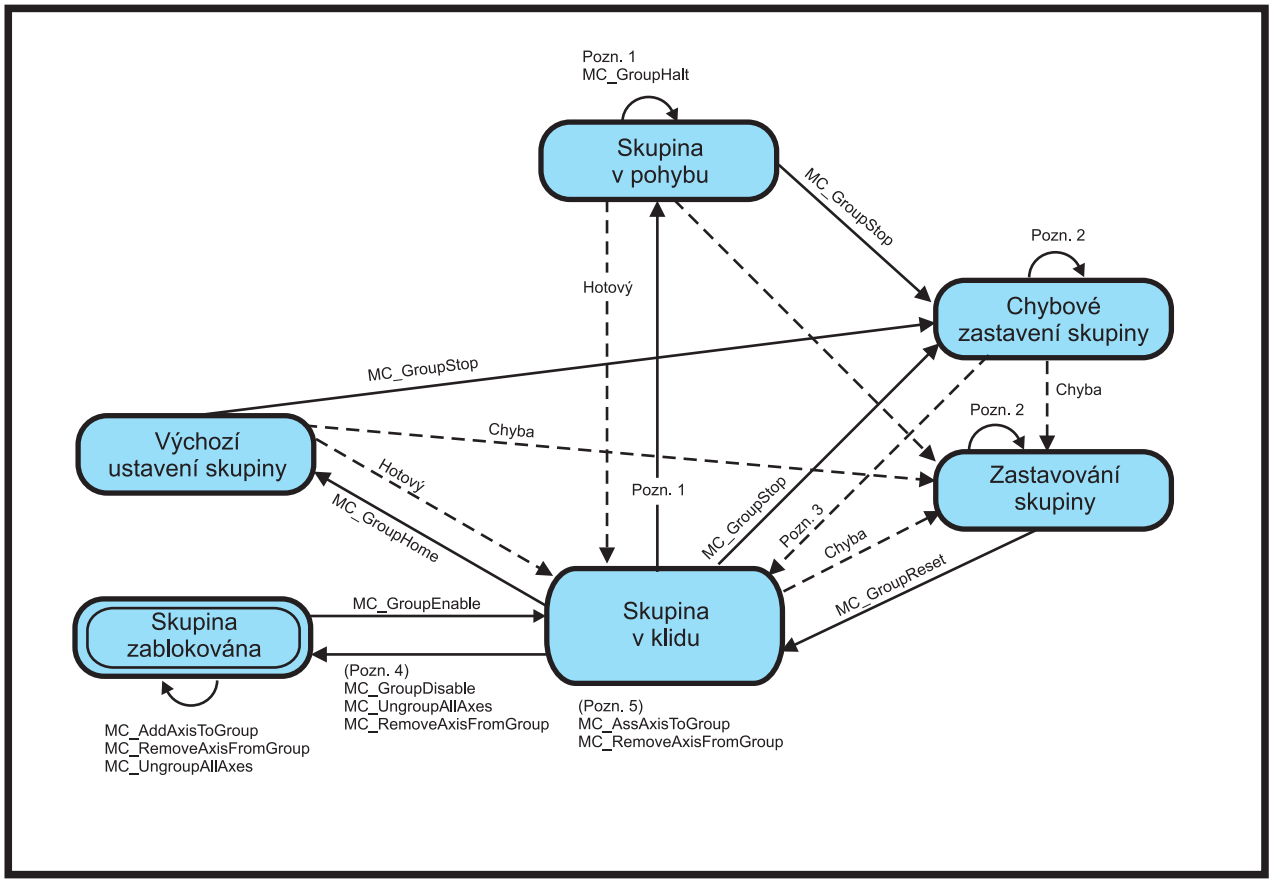
Parametry

`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`VelRot`	Maximální povolená rychlost otáčení [rotunit/s]	`Double (F64)`
`AccRot`	Maximální povolené zrychlení v otáčení [rotunit/s^2]	`Double (F64)`
`JerkRot`	Maximální povolená změna zrychlení v otáčení [rotunit/s^3]	`Double (F64)`
`InterpolationMode`	Algoritmus pro interpolaci ⊙2	`Long (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 aproximation (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 aproximation (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` ..... reservováno pro pozdější použití	
	`9` ..... reservováno pro pozdější použití	
`min`	Minimální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu `ErrorStop` ⊙[]	`Double (F64)`
`max`	Maximální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu `ErrorStop` ⊙[]	`Double (F64)`

Výstupy

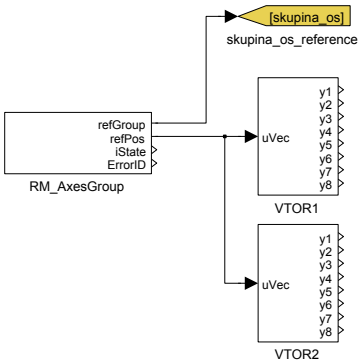
`refGroup`	Odkaz na skupinu os	`Reference`
`refPos`	Vektor generovaných poloh a rychlostí v MCS	`Reference`
`iState`	Stav skupiny	`Long (I32)`
	`0` ..... Disabled (skupina blokována)	
	`1` ..... Standby (skupina připravena)	
	`2` ..... Homing (hledání výchozí polohy)	
	`6` ..... Moving (koordinovaný 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



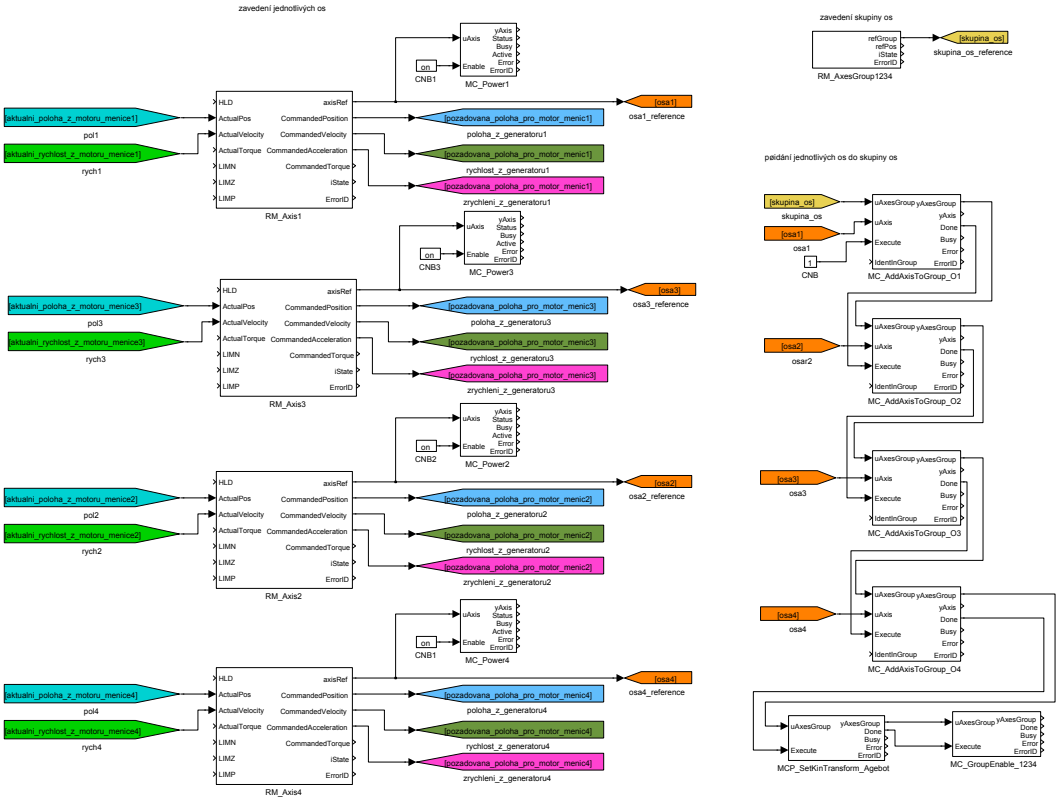
Čtení z výstupu refPos

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 MATRIX. 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

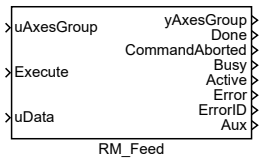
Pro správné spuštění bloku generujícího víceosý pohyb je nutné definovat jednotlivé osy RM_Axis a postupně je všechny přidat do skupiny os pomocí bloku MC_AddAxisToGroup. Dále je třeba nastavit kinematickou transformaci (dle typu stroje jeden z bloků **MC_SetKinTransform_xxx**). Nakonec je třeba aktivovat skupinu os blokem MC_GroupEnable. Pro pohyby v PCS je nutné ještě přidat kartézkou 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.


zavedení jednotlivých os
zavedení skupiny os
RM_AxesGroup1234
[skupina_os]
skupina_os_reference
MC_Power1
CNB1
[osa1]
osa1_reference
[aktualni_poloha_z_motoru_menice1]
pol1
[aktualni_rychlost_z_motoru_menice1]
rych1
RM_Axis1
[pozadovana_poloha_pro_motor_menic1]
poloha_z_generatoru1
rychlost_z_generatoru1
zrychleni_z_generatoru1
MC_Power3
CNB3
[osa3]
osa3_reference
[aktualni_poloha_z_motoru_menice3]
pol3
[aktualni_rychlost_z_motoru_menice3]
rych3
RM_Axis3
poloha_z_generatoru3
rychlost_z_generatoru3
zrychleni_z_generatoru3
MC_Power2
CNB2
[osa2]
osa2_reference
[aktualni_poloha_z_motoru_menice2]
pol2
[aktualni_rychlost_z_motoru_menice2]
rych2
RM_Axis2
poloha_z_generatoru2
rychlost_z_generatoru2
zrychleni_z_generatoru2
MC_Power4
CNB1
[osa4]
osa4_reference
[aktualni_poloha_z_motoru_menice4]
pol4
[aktualni_rychlost_z_motoru_menice4]
rych4
RM_Axis4
poloha_z_generatoru4
rychlost_z_generatoru4
zrychleni_z_generatoru4
přidání jednotlivých os do skupiny os
[skupina_os]
skupina_os
[osa1]
osa1
CNB
MC_AddAxisToGroup_O1
[osa2]
osa2
MC_AddAxisToGroup_O2
[osa3]
osa3
MC_AddAxisToGroup_O3
[osa4]
osa4
MC_AddAxisToGroup_O4
MCP_SetKinTransform_Agebot
MC_GroupEnable_1234

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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`uData`	Odkaz na matici s daty	`Reference`

Parametry

`VelFactor`	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 ↓`0.01` ↑`100.0` ⊙`1.0`	`Double (F64)`
`Relative`	Pokud je zatrženo, poloha v datech se přičte k poloze na startu (tj. první sloupec v matici by měl být 0)	`Bool`

`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	↓1 ↑3 ⊙2	`Long (I32)`
`BufferMode`	Režim převzetí osy `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)	↓1 ↑6 ⊙1	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	↓0 ↑15 ⊙1	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		`Double (F64)`

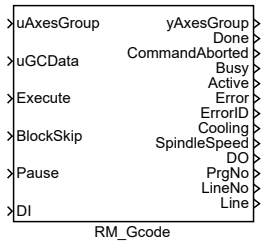
Výstupy

`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	`Error`
`Aux`	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 parametem `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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`uGCData`	Odkaz na specialni blok generujici `G-code`	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`BlockSkip`	Přeskočení bloku v `G-code` (pokud program přijde na řádku, kde je znak lomítko a je BlockSkip=true, tak se řádka přeskočí)	`Bool`

`Pause`	Pozastavení vykonávání programu (pokud je Pause=true dokončí se aktuální příkaz/řádka a další se nezpracovávají; slouží hlavně k ladění programu)	`Bool`
`DI`	Celé číslo, které slouží jako bitové pole vstupních logických signálů, které lze v `G-code` testovat a podle toho větvit program ↓0 ↑1	`Long (I32)`

Parametry

`BaseDir`	Adresář se soubory G-kódu	`String`
`MainFile`	Číslo hlavního programu	`Long (I32)`
`CoordSystem`	Volba souřadného systému ↓1 ↑3 ⊙3 `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`
`BufferMode`	Režim převzetí osy ⊙1 `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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu ⊙1 `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)	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`
`params`	Pole dodatečných parametrů pro speciální interpretery ⊙`[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]`	`Double (F64)`

`workOffsets`	Sady počátečních souřadnic ⊙[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	`Double (F64)`
`toolOffsets`	Sady korekcí délky nástroje ⊙[0 0 0]	`Double (F64)`
`cutterOffsets`	Poloměry nástroje ⊙[0 0 0]	`Double (F64)`

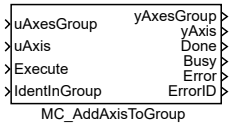
Výstupy

`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`
`Cooling`	Chlazení zapnuto	`Bool`
`SpindleSpeed`	Rychlost otáčení vřetene	`Double (F64)`
`DO`	Celé číslo, které slouží jako bitové pole výstupních logických signálů, které lze v `G-code` nastavovat	`Long (I32)`
`PrgNo`	Číslo právě vykonávaného programu	`Long (I32)`
`LineNo`	Číslo právě vykonávané řádky programu (parametr N v `G-code`)	`Long (I32)`
`Line`	Právě prováděná řádka G-kódu	`String`

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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`uAxis`	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`IdentInGroup`	Pořadí osy ve skupině (0=první nepřiřazená)	`Long (I32)`

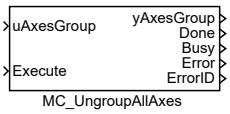
Výstupy

`yAxesGroup`	Odkaz na skupinu os	`Reference`
`yAxis`	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	`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`

MC_UngroupAllAxes – Odebrání všech ose 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, kartézké 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

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

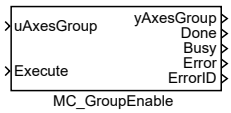
Výstupy

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	Error
	i ..... obecná chyba systému REXYGEN	

`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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`

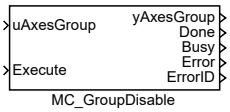
Výstupy

`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`

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

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

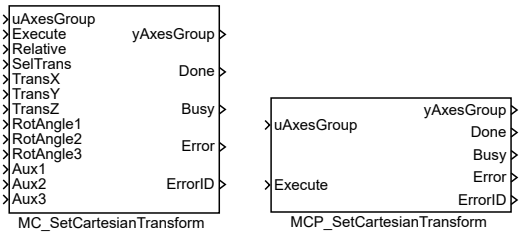
Výstupy

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	Error
	i ..... obecná chyba systému REXYGEN	

MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace

Symboly 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 kartézkých 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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Relative`	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 transforemační matici)	`Bool`
`SelTrans`	Výběr transformace pro nastavení ⊙1 `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)	`Long (I32)`

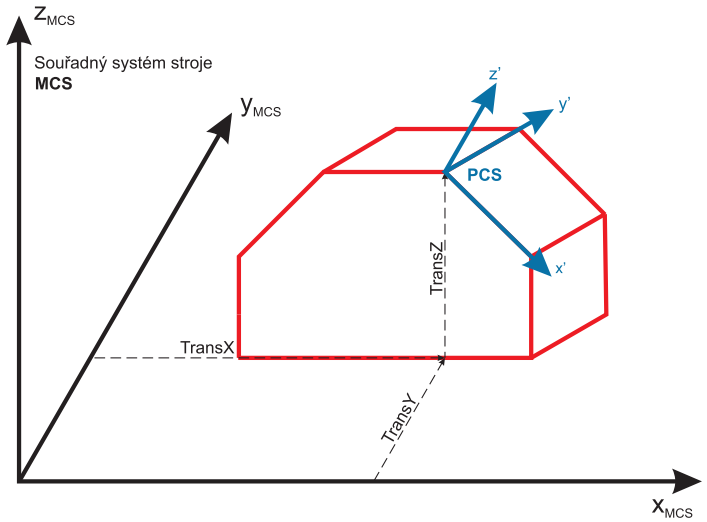
`TransX`	Posun v ose X	`Double (F64)`
`TransY`	Posun v ose Y	`Double (F64)`
`TransZ`	Posun v ose Z	`Double (F64)`
`RotAngle1`	Úhel rotace podél osy Z	`Double (F64)`
`RotAngle2`	Úhel rotace podél osy Y	`Double (F64)`
`RotAngle3`	Úhel rotace podél osy X	`Double (F64)`
`Aux1`	posun v 1. doplňkové ose	`Double (F64)`
`Aux2`	posun v 2. doplňkové ose	`Double (F64)`
`Aux3`	posun v 3. doplňkové ose	`Double (F64)`

Výstupy

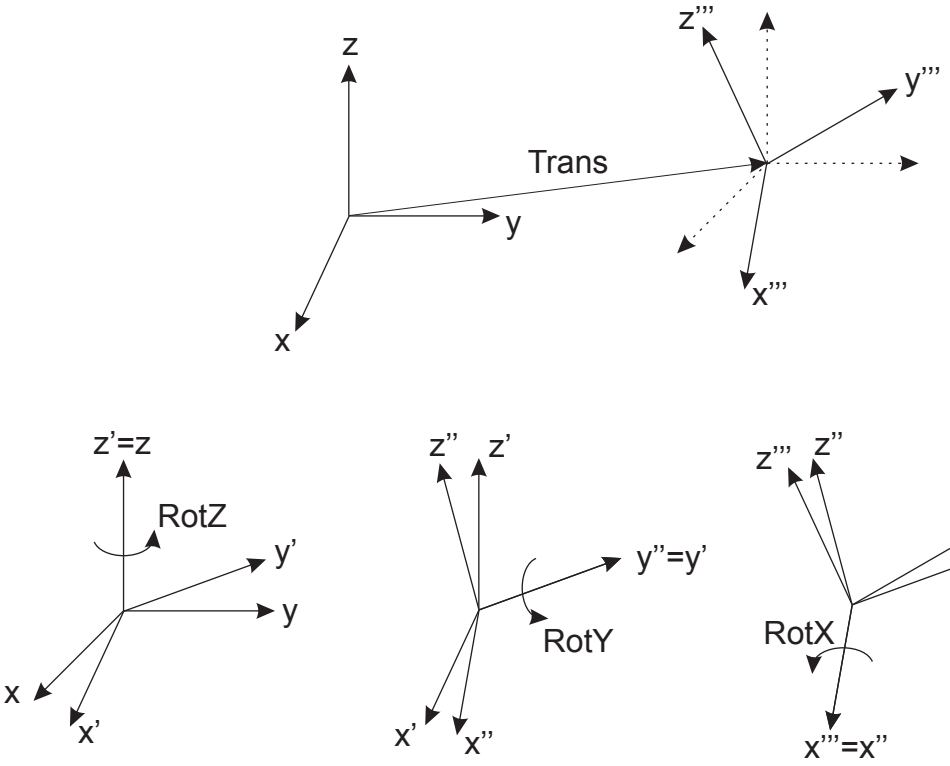
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

Definice translace a rotace

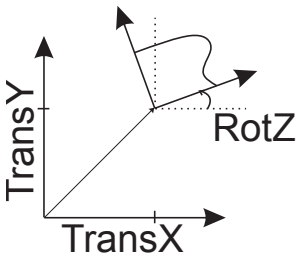
Na prvním obrázku je uveden příklad translace ze souřadného systému MCS do souřadného systému PCS.



Na druhém obrázku je pak uveden příklad rotace. Kdy celková rotace je docílena postupným provedením dílčích rotací kolem jednotlivých os.



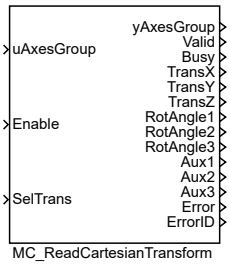
Ukázka zadání kartézské transformace (v rovině). Pro dosažení transformace na obrázku níže se spustí blok `MC_SetCartesianTransform` s nastavenou transformací {50,50,0,30,0,0}



MC_ReadCartesianTransform – Přečtení 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 kartézkých 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

`uAxesGroup`	Odkaz na skupinu os		`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)		`Bool`
`SelTrans`	Výběr transformace pro zobrazení `1` ..... PCS offset `2` ..... Tool offset `3` ..... Machine Base offset	⊙`1`	`Long (I32)`

Výstupy

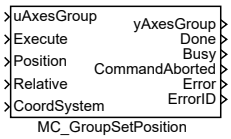
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`TransX`	Posun v ose X	`Double (F64)`
`TransY`	Posun v ose Y	`Double (F64)`
`TransZ`	Posun v ose Z	`Double (F64)`
`RotAngle1`	Úhel rotace podél osy X	`Double (F64)`

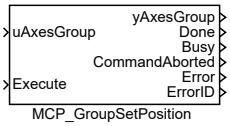
`RotAngle2`	Úhel rotace podél osy Y	`Double (F64)`
`RotAngle3`	Úhel rotace podél osy Z	`Double (F64)`
`Aux1`	posun v 1. doplňkové ose	`Double (F64)`
`Aux2`	posun v 2. doplňkové ose	`Double (F64)`
`Aux3`	posun v 3. doplňkové ose	`Double (F64)`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os

Symboly 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 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 predchozí možnosti nebo změnit transformaci MCS na PCS (řeší `MC_SetCartesianTransform`), v TCS nedava smysl

Vstupy

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Position`	Pole souřadnic (pozic a orientací)	`Reference`
`Relative`	Výběr absolutních (=false) nebo relativních (=true) souřadnic `off` ... absolute `on` .... relative	`Bool`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`

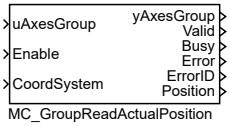
Výstupy

`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` ..... obecná chyba systému REXYGEN	

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 vzdy obsahuje 3 polohové souradnice (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

uAxesGroup	Odkaz na skupinu os	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
CoordSystem	Volba souřadného systému	Long (I32)
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

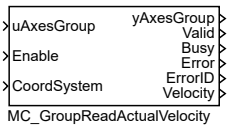
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i ..... obecná chyba systému REXYGEN	
Position	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 vzdy obsahuje 3 polohové souradnice (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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`

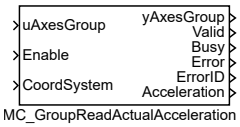
Výstupy

`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`
`Velocity`	Pole aktuálních rychlostí skupiny os	`Reference`

`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 vzdy obsahuje 3 polohové souradnice (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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`

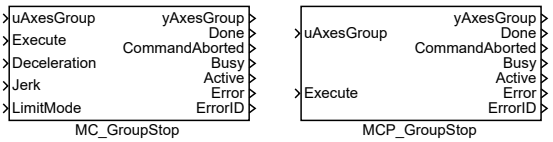
Výstupy

`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`
`Acceleration`	Pole aktuálních zrychlení skupiny os	`Reference`

MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu

Symboly 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^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`

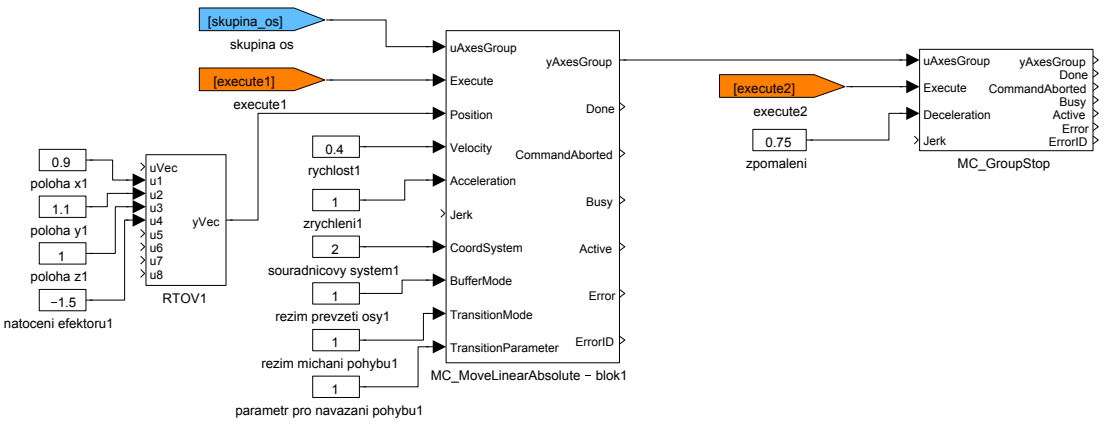
Výstupy

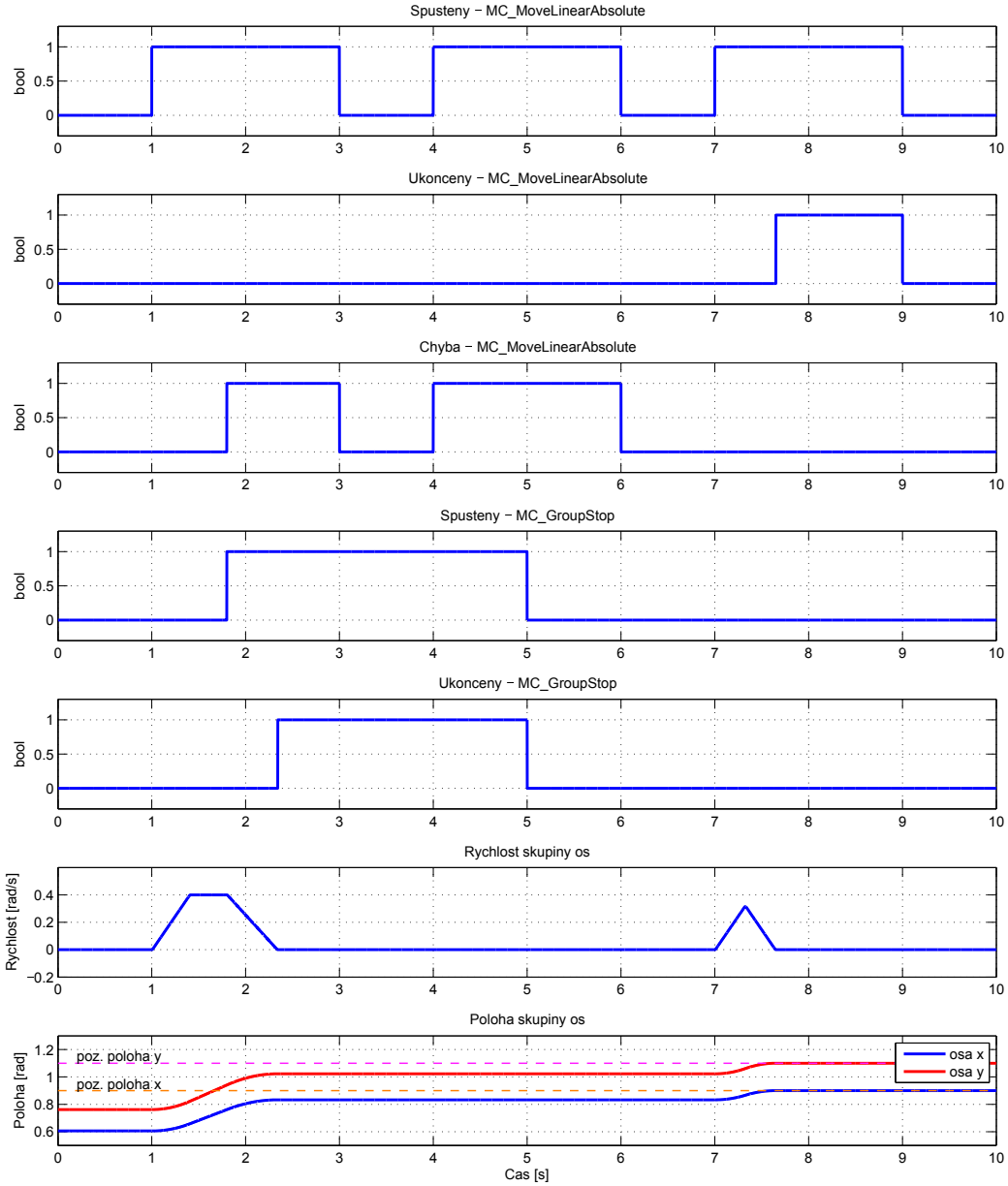
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`

`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. Následné 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.

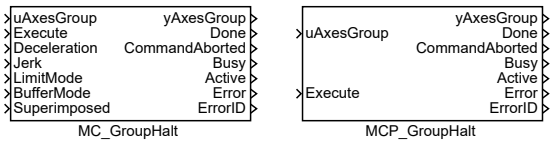



Spusteny − MC_MoveLinearAbsolute
Ukonceny − MC_MoveLinearAbsolute
Chyba − MC_MoveLinearAbsolute
Spusteny − MC_GroupStop
Ukonceny − MC_GroupStop
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha y
poz. poloha x
osa x
osa y
Cas [s]

MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)

Symboly 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

`uAxesGroup`	Odkaz na skupinu os		`Reference`
`Execute`	Náběžná hrana aktivuje blok		`Bool`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]		`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]		`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Deceleration, Jerk)	⊙1	`Long (I32)`
	`1` ..... Relative [part of group limit]		
	`2` ..... Absolute [unit/s, unit/s^2, unit/s^3]		

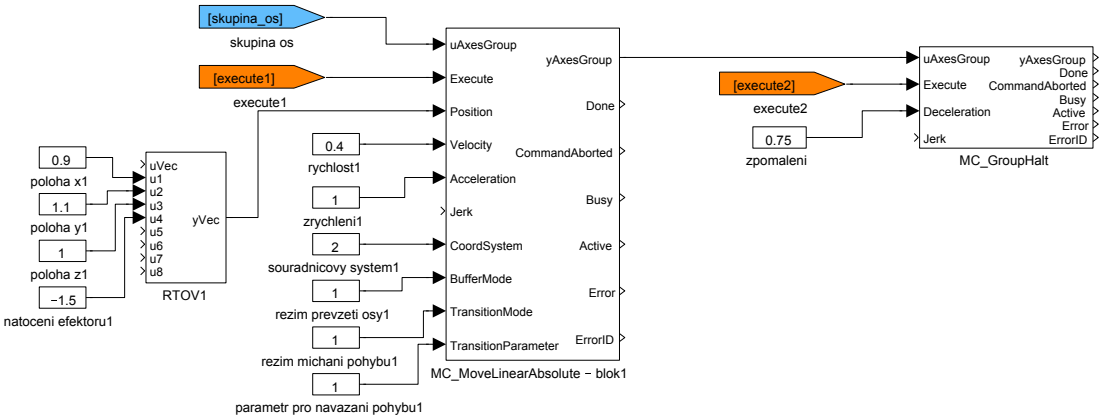
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	⊙`1`	`Long (I32)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb		`Bool`

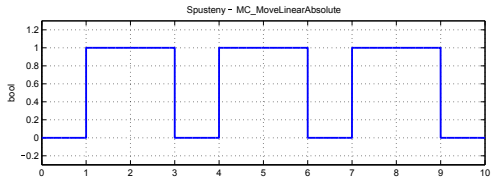
Výstupy

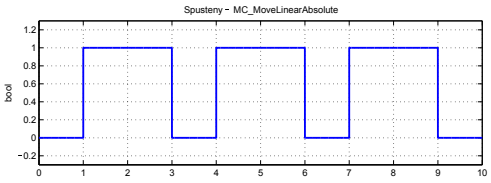
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`
`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

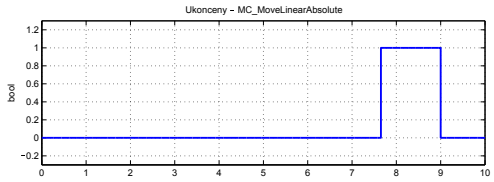
Příklad

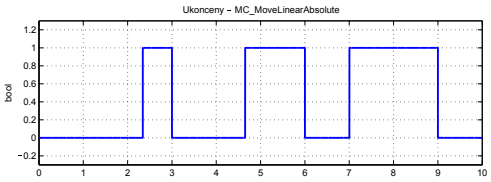
Pro porovnání je následující příklad totožný jako u bloku `MC_GroupStop`. Časové průběhy v levém sloupci odpovídají bloku `MC_GroupStop`. Průběhy v pravém sloupci bloku `MC_GroupHalt`. Je vidět, že při použití bloku `MC_GroupHalt` dojde k dojetí do požadované polohy již na druhé spuštění exekutivy bloku `MC_MoveLinearAbsolute`.

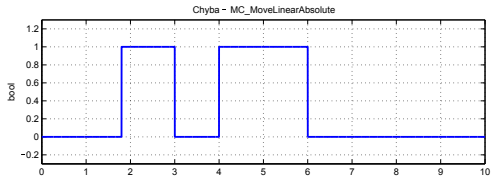

[skupina_os]
skupina os
[execute1]
execute1
0.9
poloha x1
1.1
poloha y1
1
poloha z1
−1.5
natoceni efektoru1
uVec
u1
u2
u3
u4
u5
u6
u7
u8
yVec
RTOV1
0.4
rychlost1
1
zrychleni1
2
souradnicovy system1
1
rezim prevzeti osy1
1
rezim michani pohybu1
1
parametr pro navazani pohybu1
uAxesGroup
Execute
Position
Velocity
Acceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
yAxesGroup
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveLinearAbsolute − blok1
[execute2]
execute2
0.75
zpomaleni
uAxesGroup
Execute
Deceleration
Jerk
yAxesGroup
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_GroupHalt

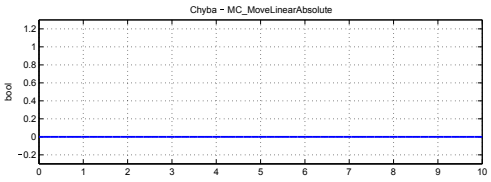

Spusteny - MC_MoveLinearAbsolute
bool

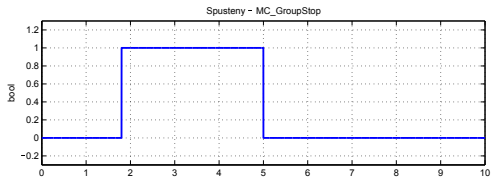

Spusteny - MC_MoveLinearAbsolute
bool

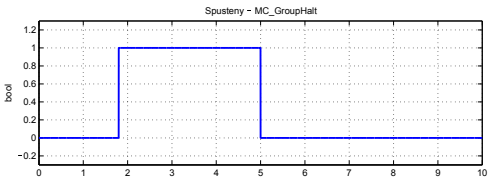

Ukonceny - MC_MoveLinearAbsolute
bool

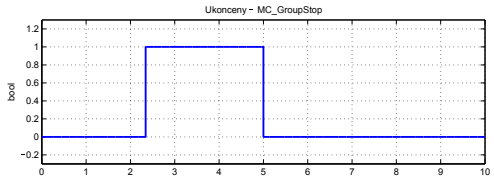

Ukonceny - MC_MoveLinearAbsolute
bool

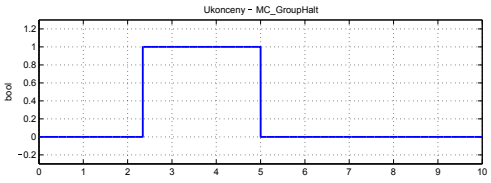

Chyba - MC_MoveLinearAbsolute
bool

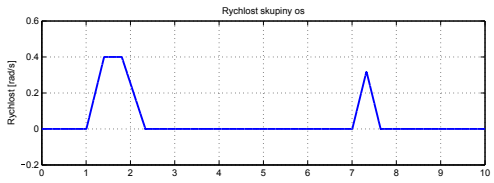

Chyba - MC_MoveLinearAbsolute
bool

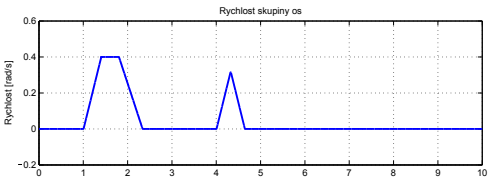

Spusteny - MC_GroupStop
bool

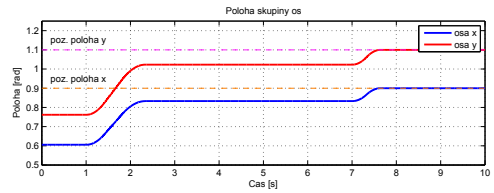

Spusteny - MC_GroupHalt
bool

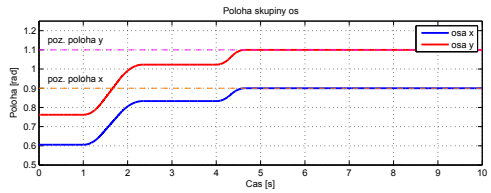

Ukonceny - MC_GroupStop
bool


Ukonceny - MC_GroupHalt
bool

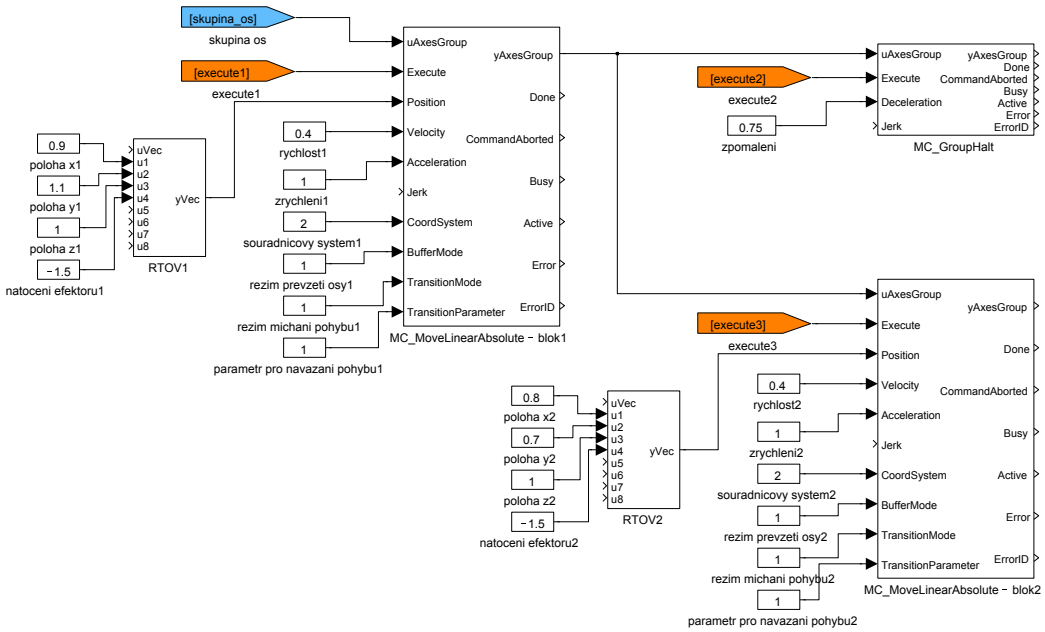

Rychlost skupiny os
Rychlost [rad/s]

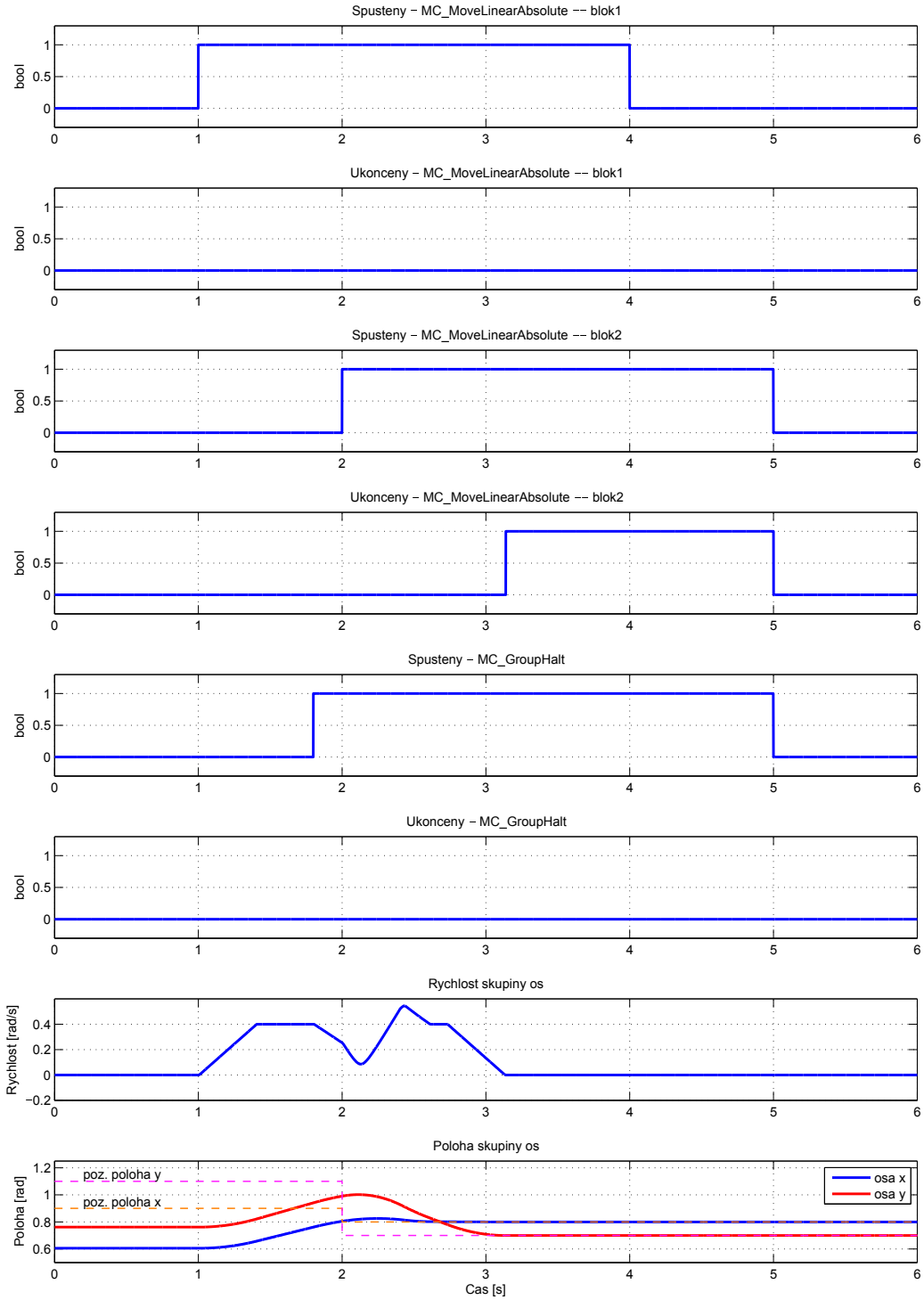

Rychlost skupiny os
Rychlost [rad/s]


Poloha skupiny os
poz. poloha y
poz. poloha x
osa x
osa y
Poloha [rad]
Cas [s]


Poloha skupiny os
poz. poloha y
poz. poloha x
osa x
osa y
Poloha [rad]
Cas [s]

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.

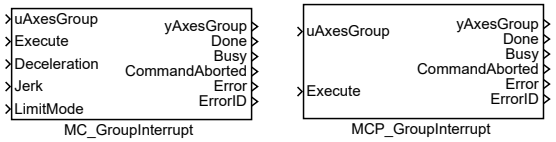



Spusteny – MC_MoveLinearAbsolute –– blok1
Ukonceny – MC_MoveLinearAbsolute –– blok1
Spusteny – MC_MoveLinearAbsolute –– blok2
Ukonceny – MC_MoveLinearAbsolute –– blok2
Spusteny – MC_GroupHalt
Ukonceny – MC_GroupHalt
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha y
poz. poloha x
osa x
osa y
Cas [s]

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

`uAxesGroup`	Odkaz na skupinu os		`Reference`
`Execute`	Náběžná hrana aktivuje blok		`Bool`
`Deceleration`	Maximální povolené zpomalení [unit/s^2]		`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]		`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Deceleration, Jerk)	⊙`1`	`Long (I32)`
	`1` ..... Relative [part of group limit]		
	`2` ..... Absolute [unit/s, unit/s^2, unit/s^3]		

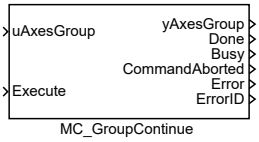
Výstupy

`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` ..... obecná chyba systému REXYGEN	

`MC_GroupContinue` – Pokračování v přerušeném pohybu

Symbol bloku

Licence: COORDINATED MOTION



Popis funkce

Blok `MC_GroupContinue` ruší působení bloku MC_GroupInterrupt. Skupina os dokončí původně prováděný pohyb před přerušením. Výstup `Done` určuje, zda se pohyb vrátil do stavu před přerušením.

Vstupy

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`

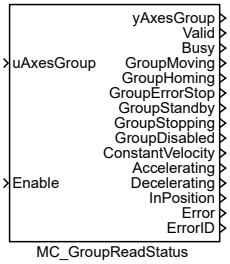
Výstupy

`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 `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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`

Výstupy

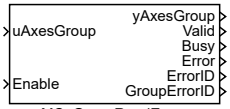
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Valid`	Příznak platnosti výstupní hodnoty	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`GroupMoving`	Stav GroupMoving	`Bool`
`GroupHoming`	Stav GroupHoming	`Bool`
`GroupErrorStop`	Stav ErrorStop	`Bool`
`GroupStandby`	Stav Standby	`Bool`
`GroupStopping`	Stav Stopping	`Bool`
`GroupDisabled`	Stav Disabled	`Bool`
`ConstantVelocity`	Pohyb konstantní rychlostí	`Bool`
`Accelerating`	Zrychlování	`Bool`
`Decelerating`	Zpomolování	`Bool`
`InPosition`	Příznak dosažení zadané pozice	`Bool`
`Error`	Příznak chyby	`Bool`

`ErrorID` Výsledek poslední operace `Error`
`i` ..... obecná chyba systému REXYGEN

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

uAxesGroup	Odkaz na skupinu os	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

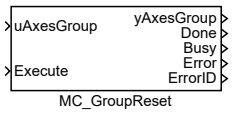
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error
GroupErrorID	Výsledek poslední operace i ..... obecná chyba systému REXYGEN	Error

`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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`

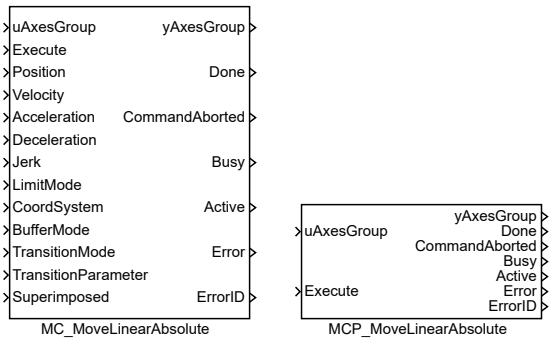
Výstupy

`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	`Error`
	`i` ..... obecná chyba systému REXYGEN	

MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice)

Symboly 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 trajektoriji). 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`

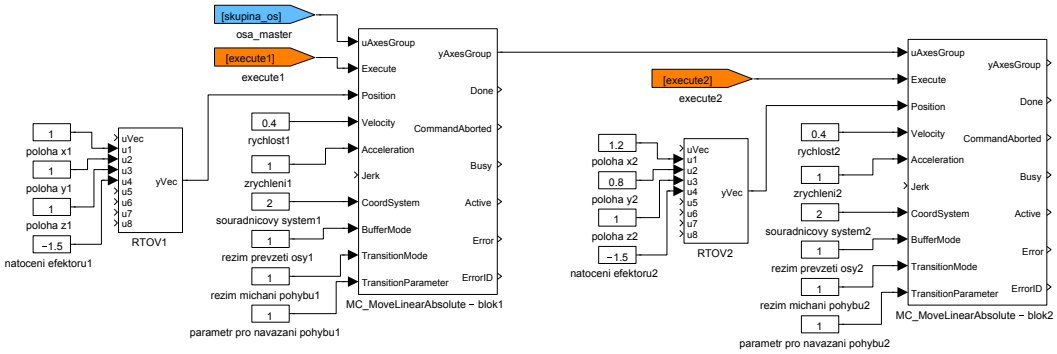
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Position`	Pole souřadnic (pozic a orientací)	`Reference`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Velocity, Acceleration, Jerk) ⊙1 `1` ..... Relative [part of group limit] `2` ..... Absolute [unit/s, unit/s^2, unit/s^3]	`Long (I32)`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`

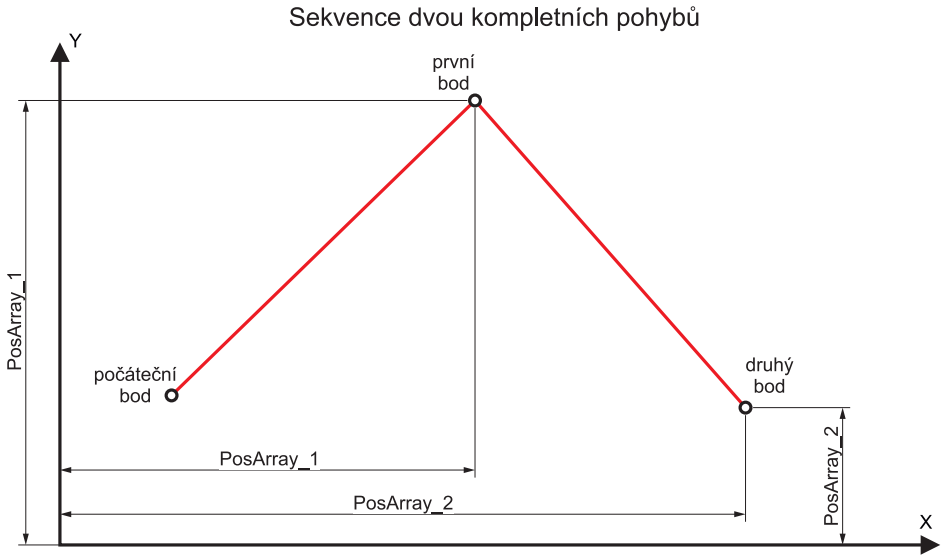
Výstupy

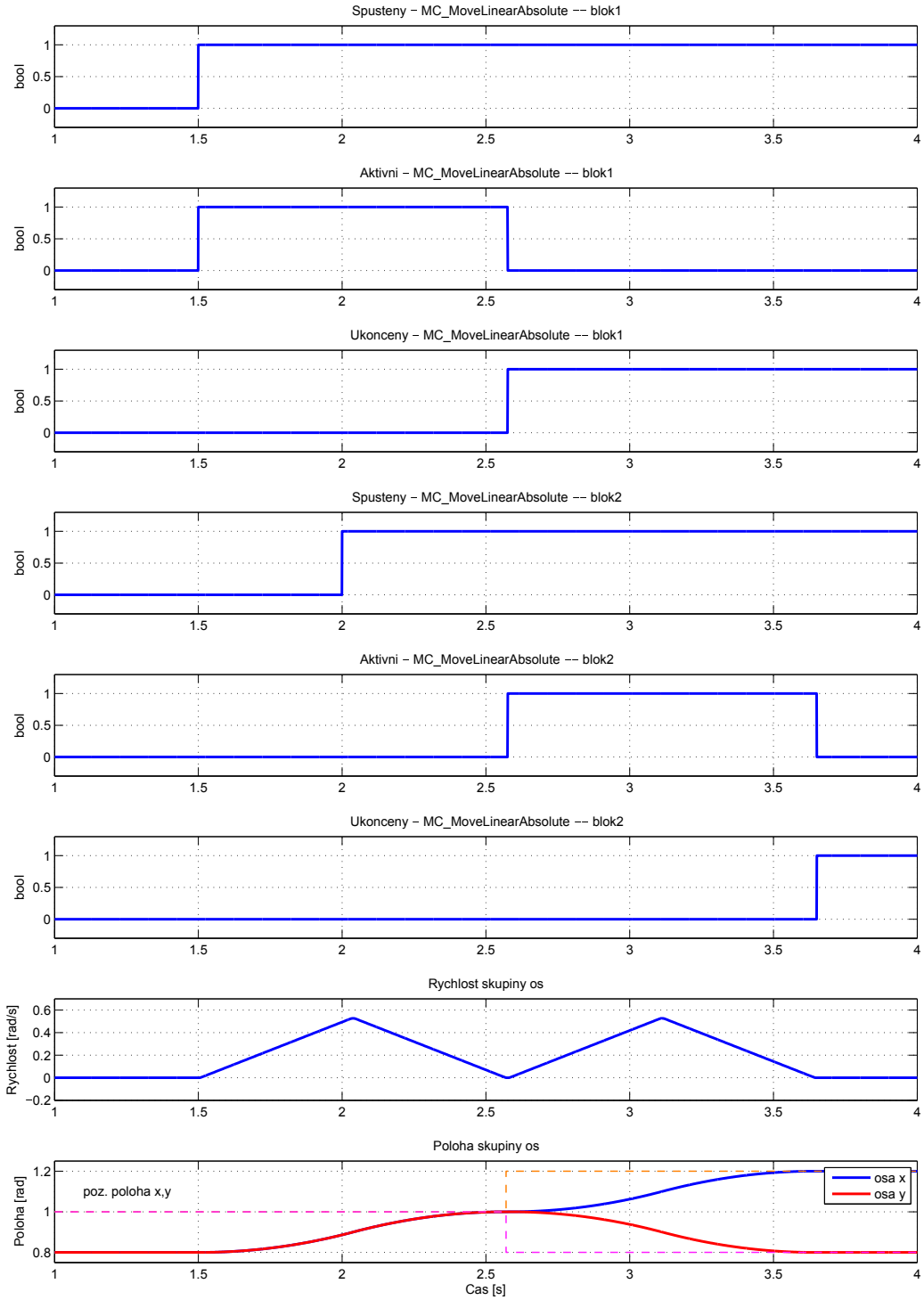
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`

`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	`Error`
	`i` ..... obecná chyba systému REXYGEN	

Příklad



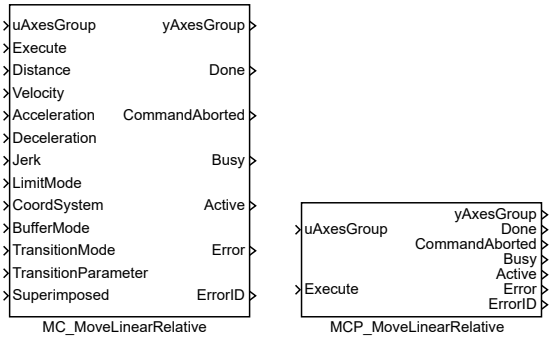



Spusteny – MC_MoveLinearAbsolute –– blok1
Aktivni – MC_MoveLinearAbsolute –– blok1
Ukonceny – MC_MoveLinearAbsolute –– blok1
Spusteny – MC_MoveLinearAbsolute –– blok2
Aktivni – MC_MoveLinearAbsolute –– blok2
Ukonceny – MC_MoveLinearAbsolute –– blok2
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha x,y
osa x
osa y
Cas [s]

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 trajektoriji). 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 `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`

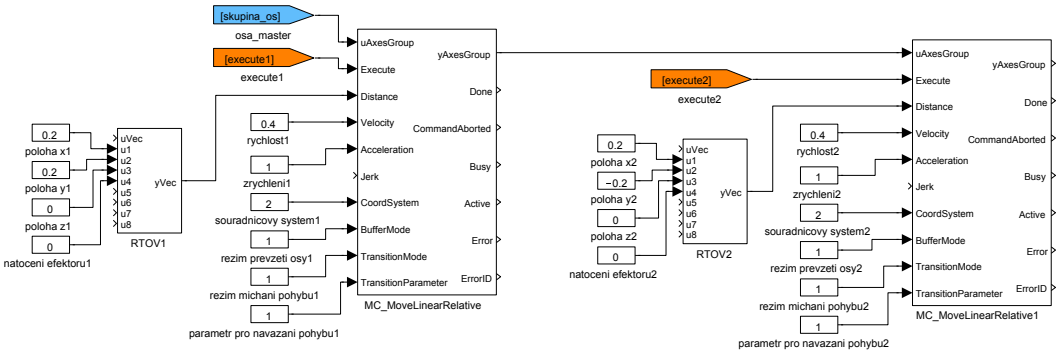
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Pole souřadnic (relativních pozic a orientací)	`Reference`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]	`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Velocity, Acceleration, Jerk) ⊙1 `1` ..... Relative [part of group limit] `2` ..... Absolute [unit/s, unit/s^2, unit/s^3]	`Long (I32)`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`

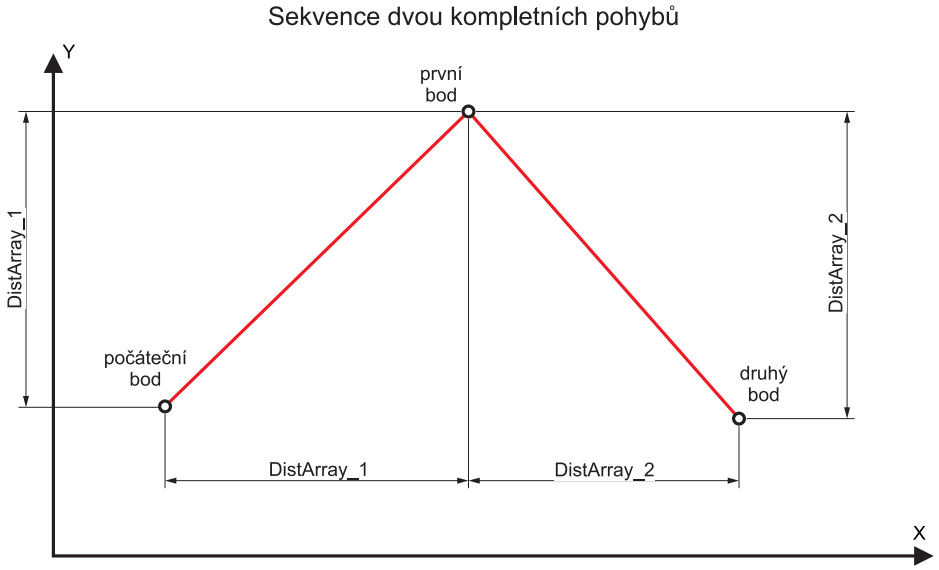
Výstupy

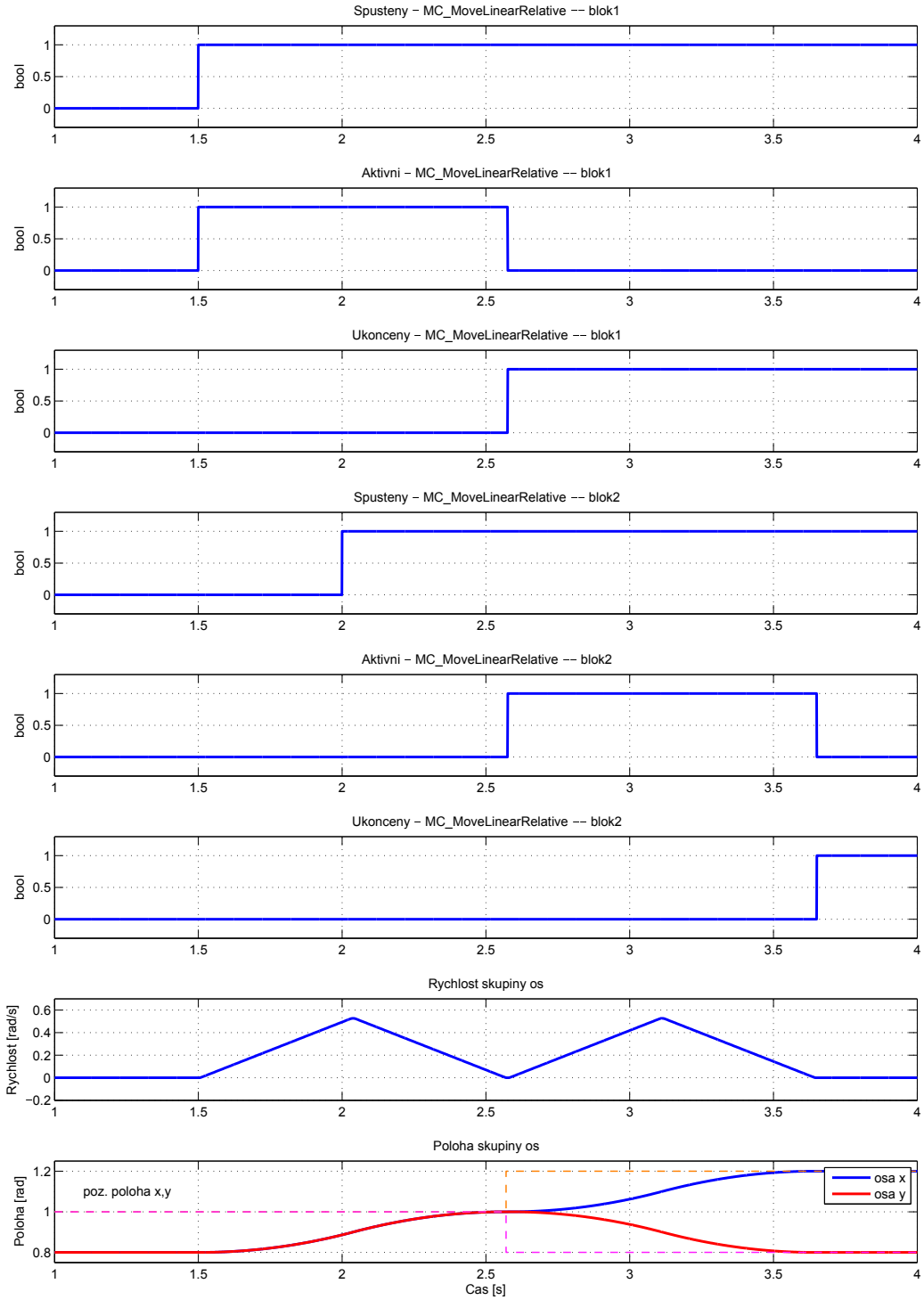
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`

`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	`Error`
	`i` ..... obecná chyba systému REXYGEN	

Příklad



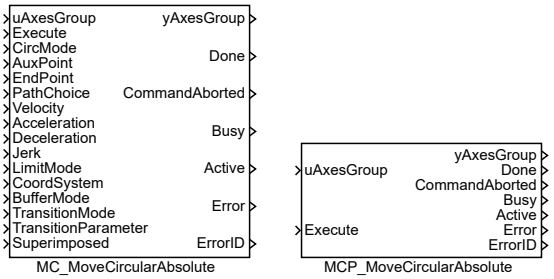



Spusteny – MC_MoveLinearRelative -- blok1
Aktivni – MC_MoveLinearRelative -- blok1
Ukonceny – MC_MoveLinearRelative -- blok1
Spusteny – MC_MoveLinearRelative -- blok2
Aktivni – MC_MoveLinearRelative -- blok2
Ukonceny – MC_MoveLinearRelative -- blok2
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha x,y
osa x
osa y
Cas [s]

MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)

Symboly bloků

Licence: COORDINATED MOTION



Popis funkce

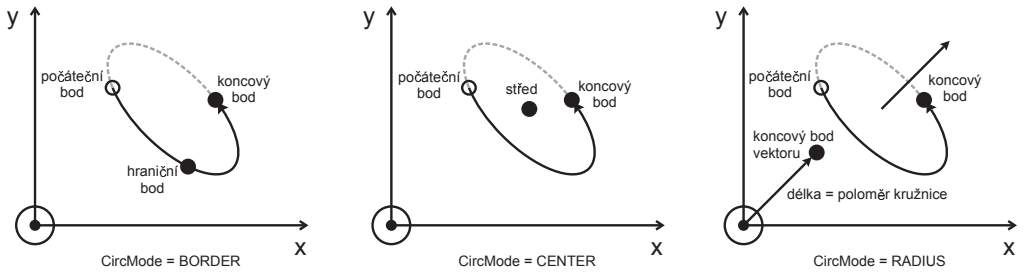
Bloky `MC_MoveCircularAbsolute` *a* `MCP_MoveCircularAbsolute` *mají naprosto shodnou funkci, jediným rozdílem je, že* `MCP_` *varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

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 trajektoriji). 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 způsobu 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 přeurčená 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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`CircMode`	Určuje význam vstupních signálů AuxPoint a CircDirection `1` ..... BORDER `2` ..... CENTER `3` ..... RADIUS	`Long (I32)`
`AuxPoint`	Absolutně zadaná pozice	`Reference`
`EndPoint`	Absolutně zadaná pozice koncového bodu	`Reference`
`PathChoice`	Volba směru `1` ..... Ve směru hodinových ručiček `2` ..... Protisměru hodinových ručiček	`Long (I32)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`

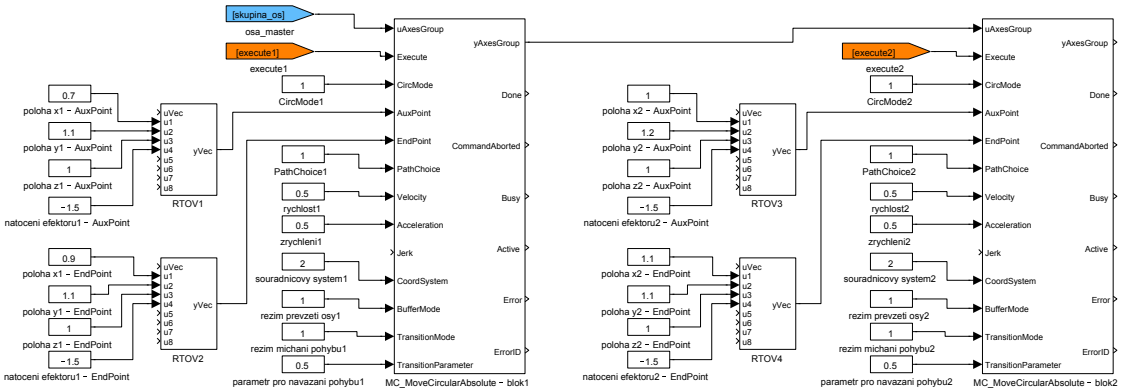
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]		`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Velocity, Acceleration, Jerk) `1` ..... Relative [part of group limit] `2` ..... Absolute [unit/s, unit/s^2, unit/s^3]	⊙`1`	`Long (I32)`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS		`Long (I32)`
`BufferMode`	Režim převzetí osy `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)		`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)		`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb		`Bool`

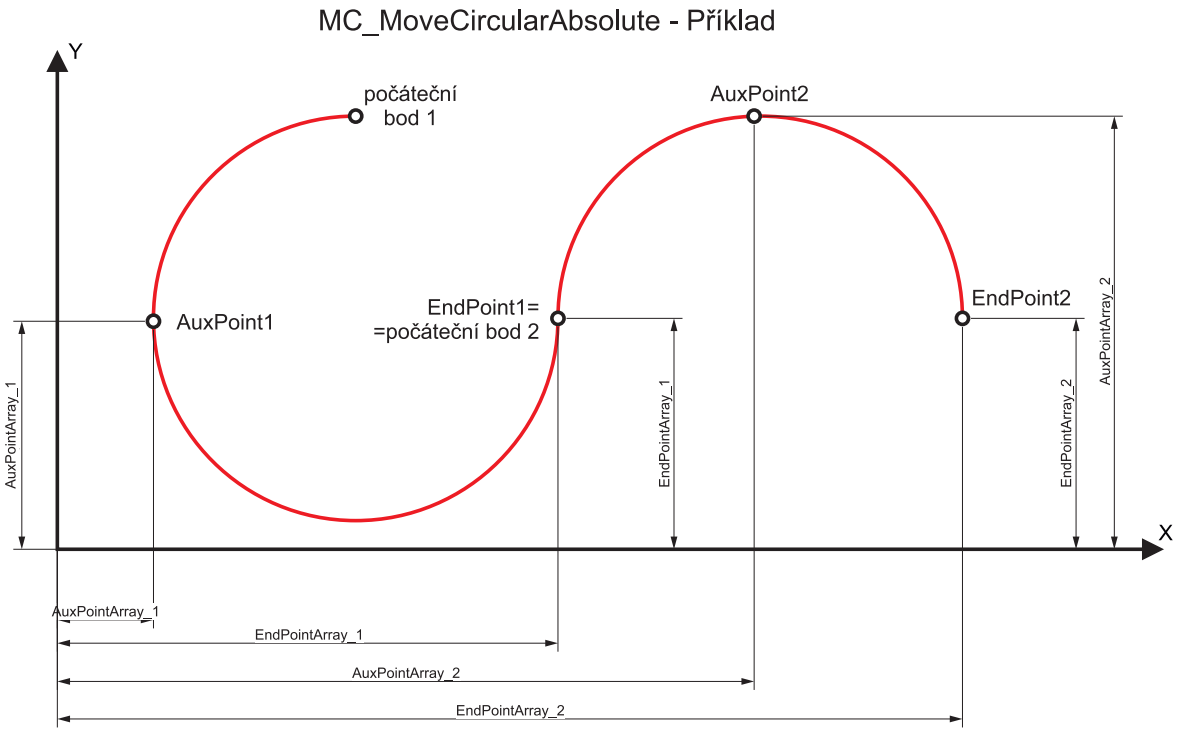
Výstupy

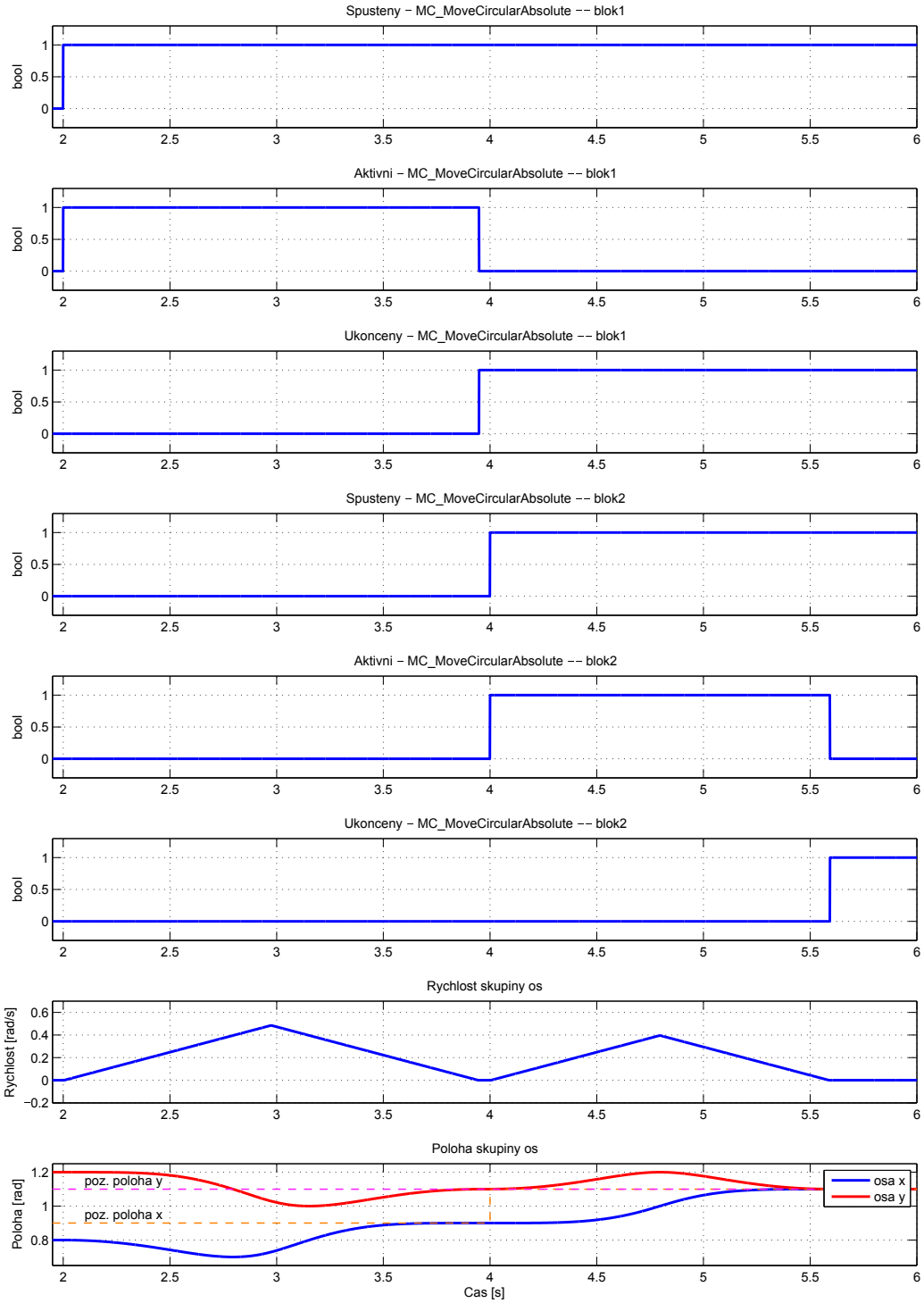
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`

`ErrorID`	Výsledek poslední operace `i` ..... obecná chyba systému REXYGEN	`Error`

Příklad



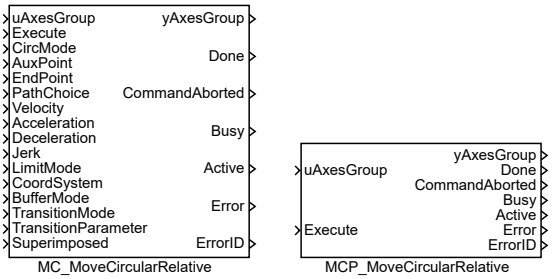



Spusteny – MC_MoveCircularAbsolute -- blok1
Aktivni – MC_MoveCircularAbsolute -- blok1
Ukonceny – MC_MoveCircularAbsolute -- blok1
Spusteny – MC_MoveCircularAbsolute -- blok2
Aktivni – MC_MoveCircularAbsolute -- blok2
Ukonceny – MC_MoveCircularAbsolute -- blok2
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha y
poz. poloha x
osa x
osa y
Cas [s]

MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)

Symboly bloků

Licence: COORDINATED MOTION



Popis funkce

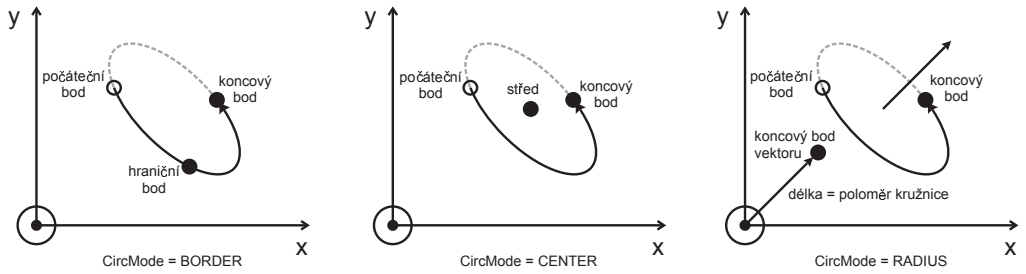
Bloky `MC_MoveCircularRelative` *a* `MCP_MoveCircularRelative` *mají naprosto shodnou funkci, jediným rozdílem je, že* `MCP_` *varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok `MC_MoveCircularRelatice` 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 trajektoriji). 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 způsobu 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 přeurčená 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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`CircMode`	Určuje význam vstupních signálů AuxPoint a CircDirection `1` ..... BORDER `2` ..... CENTER `3` ..... RADIUS	`Long (I32)`
`AuxPoint`	Absolutně zadaná pozice	`Reference`
`EndPoint`	Absolutně zadaná pozice koncového bodu	`Reference`
`PathChoice`	Volba směru `1` ..... Ve směru hodinových ručiček `2` ..... Protisměru hodinových ručiček	`Long (I32)`
`Velocity`	Maximální povolená rychlost [unit/s]	`Double (F64)`
`Acceleration`	Maximální povolené zrychlení [unit/s^2]	`Double (F64)`

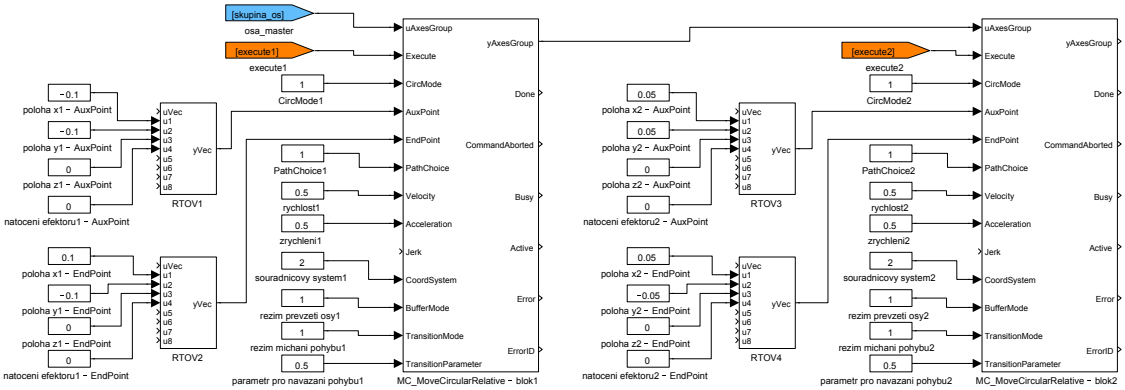
`Jerk`	Maximální povolená změna zrychlení [unit/s^3]		`Double (F64)`
`LimitMode`	Volba jednotek pro limity (Velocity, Acceleration, Jerk) `1` ..... Relative [part of group limit] `2` ..... Absolute [unit/s, unit/s^2, unit/s^3]	⊙`1`	`Long (I32)`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS		`Long (I32)`
`BufferMode`	Režim převzetí osy `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)		`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)		`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb		`Bool`

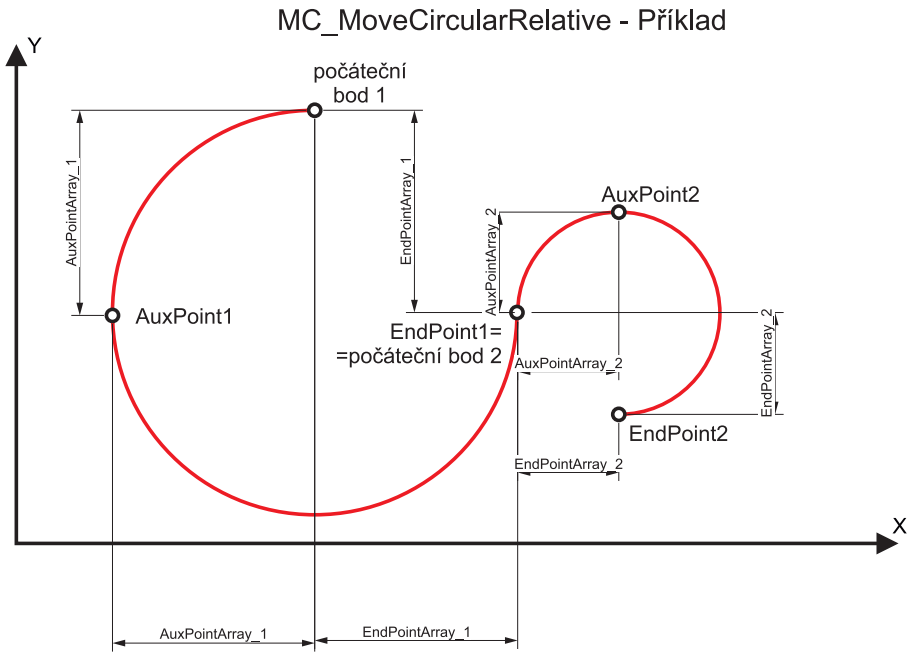
Výstupy

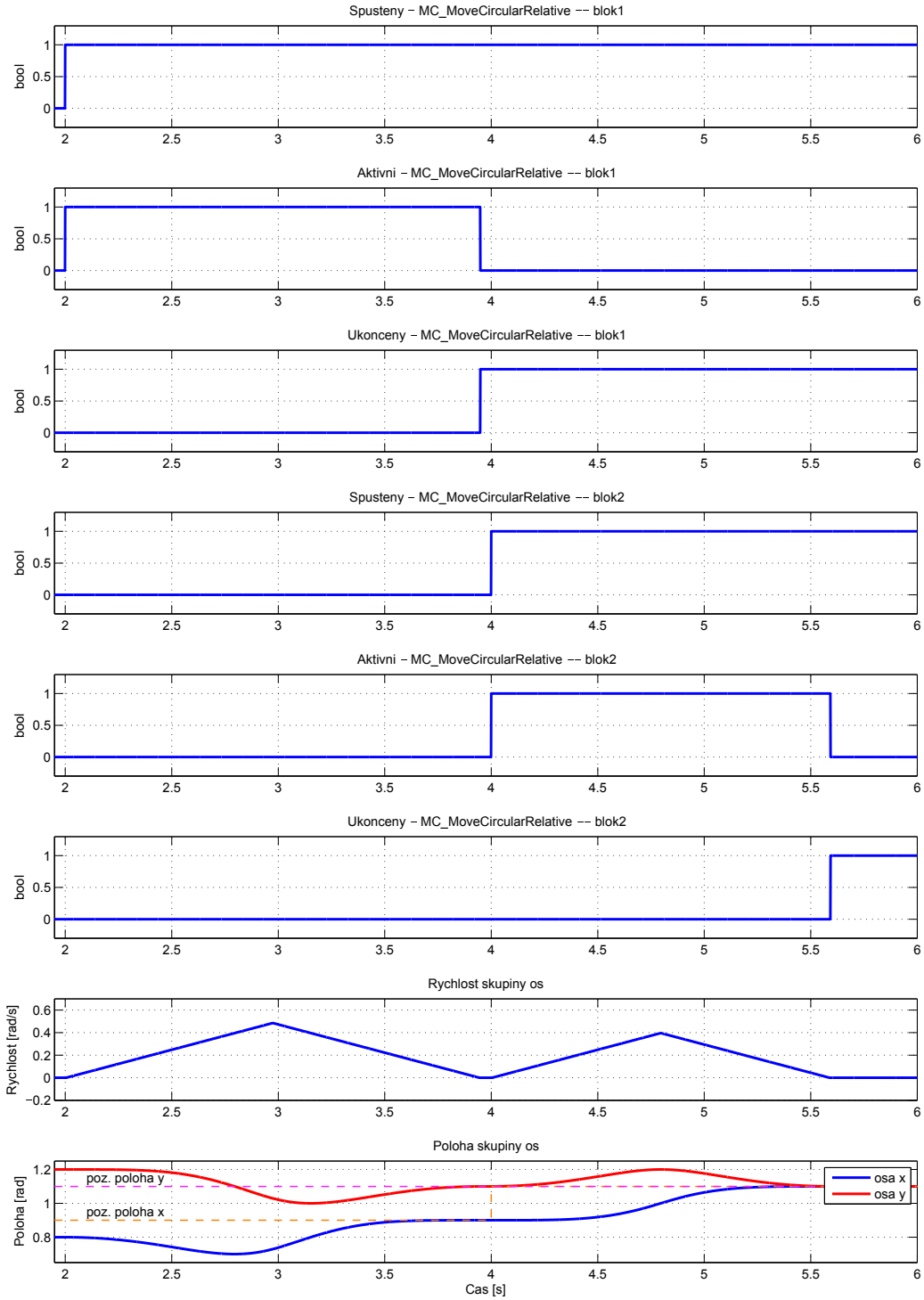
`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Done`	Příznak dokončení algoritmu	`Bool`
`CommandAborted`	Příznak přerušení funkce bloku	`Bool`
`Busy`	Příznak, že algoritmus ještě neskončil	`Bool`
`Active`	Příznak, že blok řídí osu	`Bool`
`Error`	Příznak chyby	`Bool`

`ErrorID` Výsledek poslední operace `Error`
`i` ..... obecná chyba systému REXYGEN

Příklad



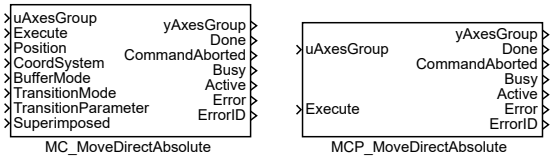



Spusteny – MC_MoveCircularRelative –– blok1
Aktivni – MC_MoveCircularRelative –– blok1
Ukonceny – MC_MoveCircularRelative –– blok1
Spusteny – MC_MoveCircularRelative –– blok2
Aktivni – MC_MoveCircularRelative –– blok2
Ukonceny – MC_MoveCircularRelative –– blok2
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha y
poz. poloha x
osa x
osa y
Cas [s]

`MC_MoveDirectAbsolute`, `MCP_MoveDirectAbsolute` – Nekoordinovaný pohyb do pozice (absolutní souřadnice)

Symboly 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

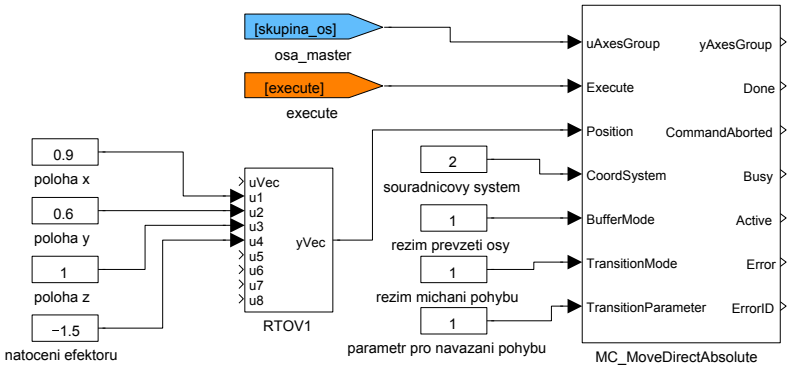
`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Position`	Pole souřadnic (pozic a orientací)	`Reference`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`

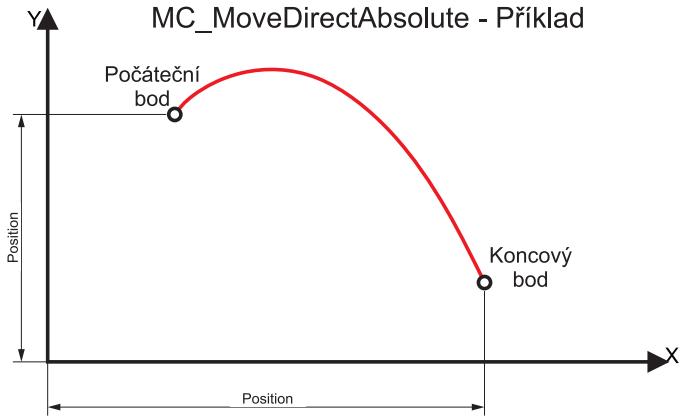
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`

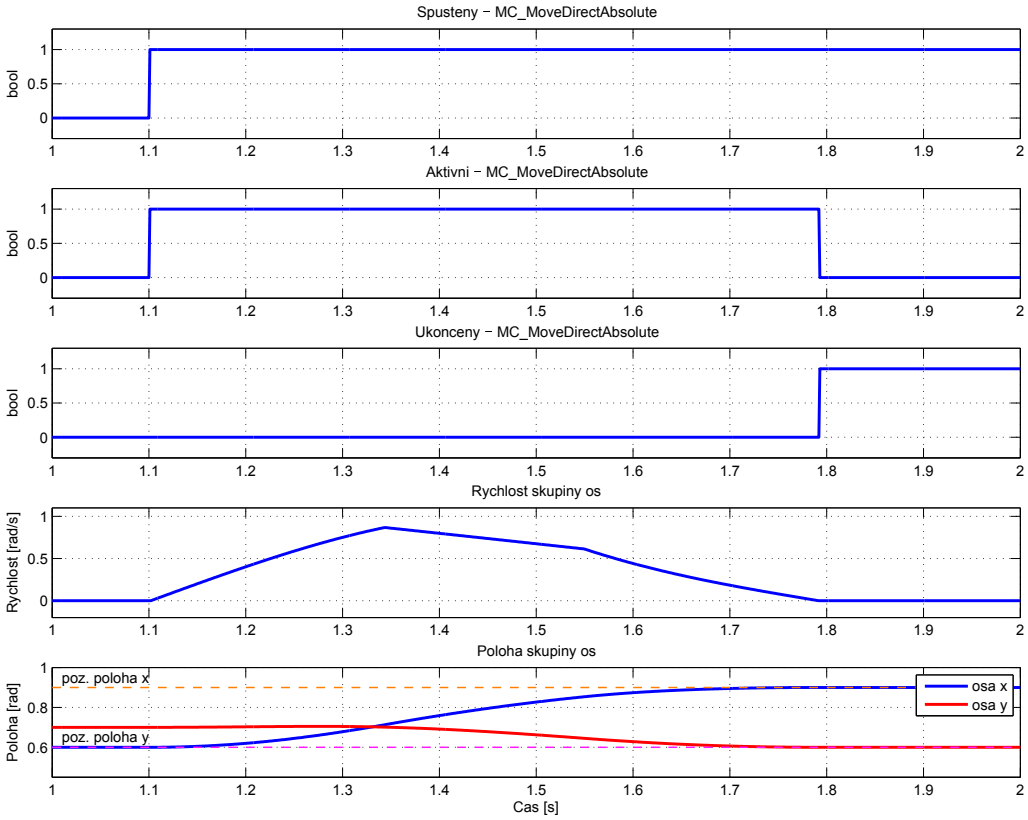
Výstupy

`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


[skupina_os]
osa_master
[execute]
execute
0.9
poloha x
0.6
poloha y
1
poloha z
−1.5
natoceni efektoru
uVec
u1
u2
u3
u4
u5
u6
u7
u8
yVec
RTOV1
2
souradnicovy system
1
rezim prevzeti osy
1
rezim michani pohybu
1
parametr pro navazani pohybu
uAxesGroup
Execute
Position
CoordSystem
BufferMode
TransitionMode
TransitionParameter
yAxesGroup
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveDirectAbsolute

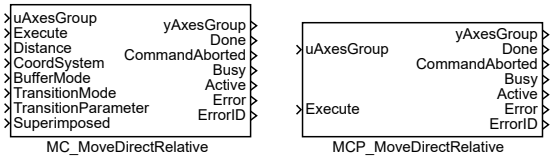

MC_MoveDirectAbsolute - Příklad
Y
Počáteční
bod
Koncový
bod
X
Position
Position


Spusteny – MC_MoveDirectAbsolute
bool
Aktivni – MC_MoveDirectAbsolute
bool
Ukonceny – MC_MoveDirectAbsolute
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
Poloha [rad]
poz. poloha x
poz. poloha y
osa x
osa y
Cas [s]

MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symboly 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

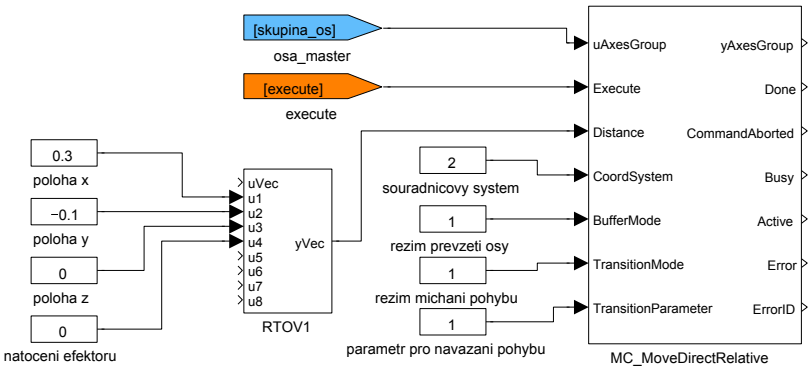
`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`Distance`	Pole souřadnic (relativních pozic a orientací)	`Reference`
`CoordSystem`	Volba souřadného systému	`Long (I32)`
	`1` ..... ACM	
	`2` ..... MCS	
	`3` ..... PCS	

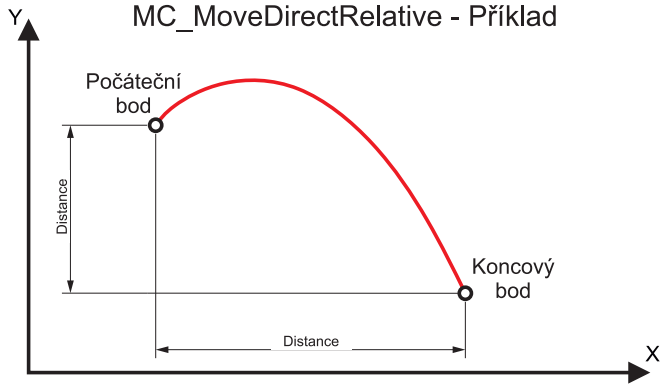
`BufferMode`	Režim převzetí osy `1` ..... Aborting (nový blok se spustí okamžitě) `2` ..... Buffered (nový blok se spustí po dokončení předchozího) `3` ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) `4` ..... Blending high (nový 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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	`Long (I32)`
`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`

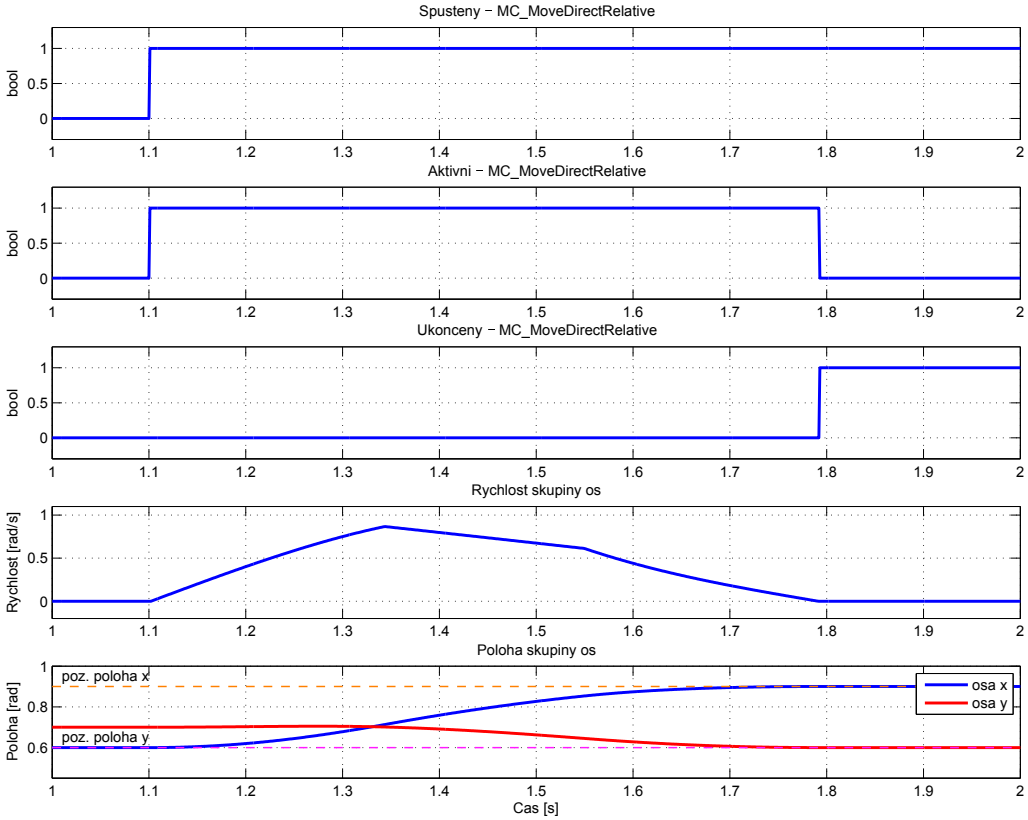
Výstupy

`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


[skupina_os]
osa_master
[execute]
execute
0.3
poloha x
−0.1
poloha y
0
poloha z
0
natoceni efektoru
uVec
u1
u2
u3
u4
u5
u6
u7
u8
yVec
RTOV1
2
souradnicovy system
1
rezim prevzeti osy
1
rezim michani pohybu
1
parametr pro navazani pohybu
uAxesGroup
Execute
Distance
CoordSystem
BufferMode
TransitionMode
TransitionParameter
yAxesGroup
Done
CommandAborted
Busy
Active
Error
ErrorID
MC_MoveDirectRelative

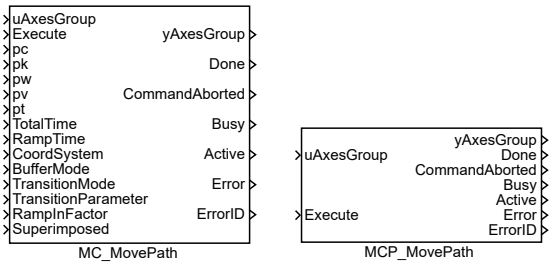

MC_MoveDirectRelative - Příklad
Y
Počáteční bod
Distance
Koncový bod
Distance
X


Spusteny – MC_MoveDirectRelative
bool
Aktivni – MC_MoveDirectRelative
bool
Ukonceny – MC_MoveDirectRelative
bool
Rychlost skupiny os
Rychlost [rad/s]
Poloha skupiny os
poz. poloha x
poz. poloha y
osa x
osa y
Poloha [rad]
Cas [s]

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 proměné (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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`
`TotalTime`	Čas celého pohybu	`Double (F64)`
`RampTime`	Čas [s] zrychlování/zpomalování	`Double (F64)`
`CoordSystem`	Volba souřadného systému `1` ..... ACM `2` ..... MCS `3` ..... PCS	`Long (I32)`
`BufferMode`	Režim převzetí osy `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)	`Long (I32)`
`TransitionMode`	Režim míchání pohybu `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)	`Long (I32)`

`TransitionParameter`	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	`Double (F64)`
`RampInFactor`	faktor pro najíždění na počáteční polohu (0 = RampIn režim se nepoužívá), odpovídá přibližně relativní rychlosti a zrychlení	`Double (F64)`
`Superimposed`	Příznak vykonání jako vedlejší (superimposed) pohyb	`Bool`

Parametry

`pc`	Matice řídicích bodů ⊙`[0.0 1.0 2.0; 0.0 1.0 1.0; 0.0 1.0 0.0]`		`Double (F64)`
`pk`	Uzlový vektor	⊙`[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`	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)`
`pt`	Uzlové body pro korekci posuvu	⊙`[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)`

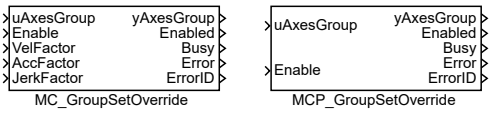
Výstupy

`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`

MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině

Symboly 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

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Enable`	Povolení funkce bloku (aktivace výstupů)	`Bool`
`VelFactor`	Faktor násobení pro rychlost	`Double (F64)`
`AccFactor`	Faktor násobení pro zrychlení	`Double (F64)`
`JerkFactor`	Faktor násobení pro změnu zrychlení	`Double (F64)`

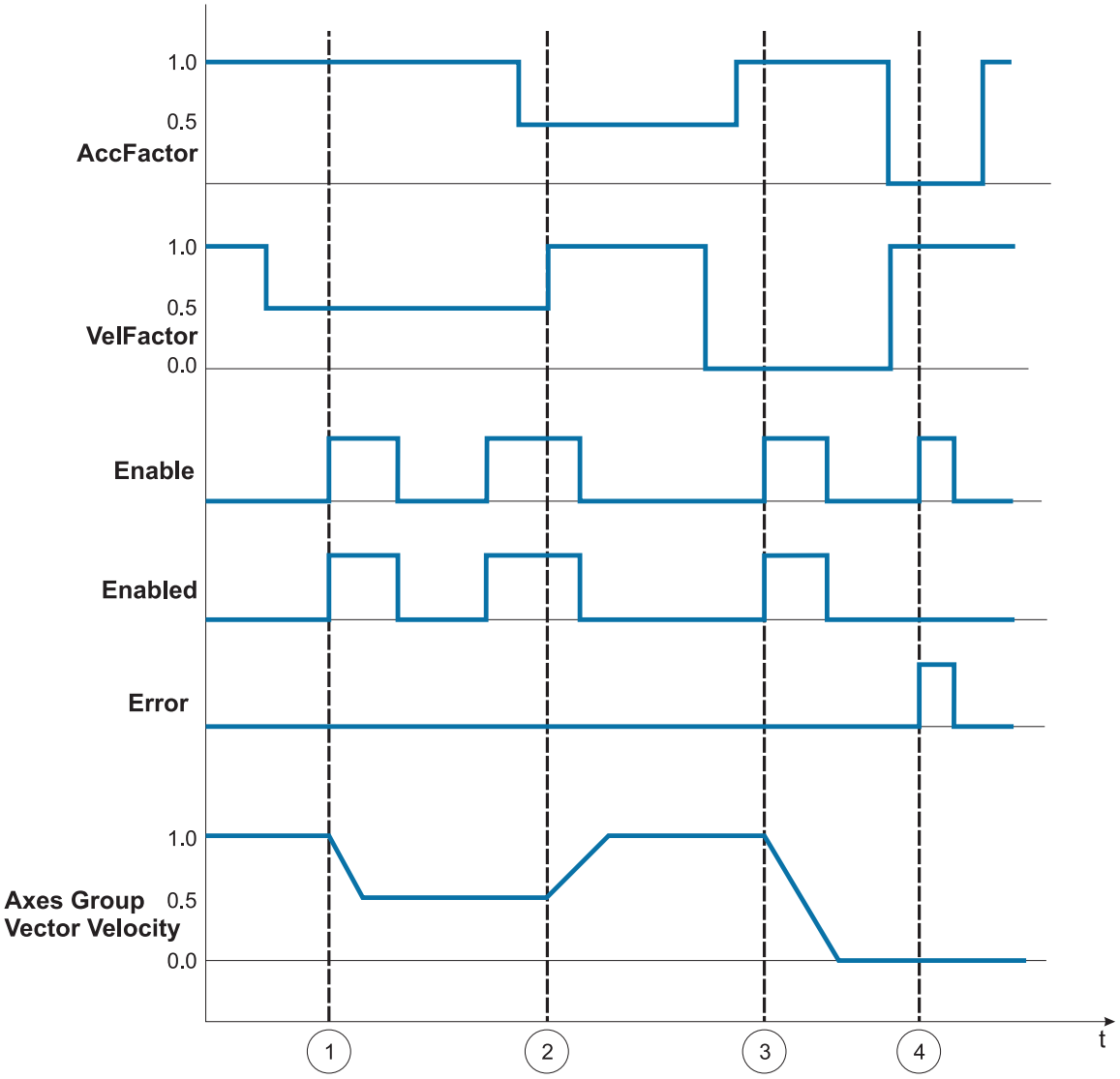
Parametr

`diff`	Pásmo necitlivosti (pro přepočet trajektorie)	⊙`0.05`	`Double (F64)`

Výstupy

`yAxesGroup`	Odkaz na skupinu os	`Reference`
`Enabled`	Signalizuje uspěšné 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

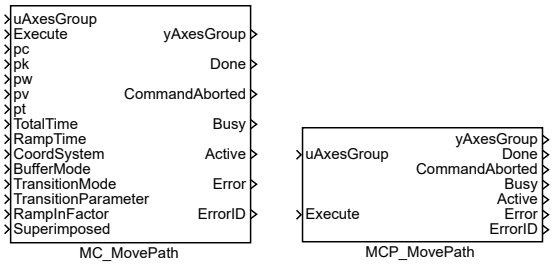


1. Rychlost skupiny os zmenšena na 50% se 100% zpomalením
2. Rychlost skupiny os vrácena na 100% s 50% akcelerací
3. Rychlost skupiny os zmenšena na 0% se 100% zpomalením
4. Žádná změna, protože není povolen AccFactor = 0.0, je sepnuta chyba

MC_SetKinTransform_Lin – Nastavení kinematické transformace

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_SetKinTransform_Lin` slouží pro nastavení kinematické transformace. To je transformace mezi polohou motorů a kartezkými souřadnicemi koncového efektoru stroje. Pro různé typy strojů má tato funkce různý charakter a parametry. Tento blok slouží pro případy, kdy je transformace prázdná, tj. polohy motorů přímo odpovídají kartézkým souřadnicím koncového efektoru (to je typické pro obráběcí stroje nebo portálové manipulátory) nebo lineární (např. struktura T-bot).

Systém motion control obsahuje 3 souřadnice polohy, kvaternion pro určení orientace a několik dalších (tzv. aux) souřadnic. Blok `MC_SetKinTransform_Lin` umožňuje nadefinovat, kolik používáme polohových souřadnic, (např. některý stroj se pohybuje jen v ploše a pak používá jen 2 polohové souřadnice nebo stroj se může jen natáčet a pak nemá žádnou polohovou souřadnici), zda stroj používá natočení v 3D prostoru, kolik a jakých (posuvné nebo rotační) dalších os je použito. Počet polohových os definuje parametr `npos`. Orientace je interně vždy reprezentována kvaternionem. Jak se bude zadávat do parametrů bloků určuje parametr `RotMode`. Lze ji zadávat buď kvaternionem nebo pomocí Eulerovo úhlů (ty si lze představit jako souřadnice na zeměkouli: zeměpisná délka, zeměpisná šířka, azimut). Protože tato kinematika je lineární, stejná čísla, která se zadávají do parametrů odpovídající požadované orientaci, jsou pak i polohy odpovídajících os/motorů.

Parametry `A` a `b`. slouží pro lineární transformaci. Pokud polohy os/motorů přímo odpovídají kartézkým souřadnicím, transformace se neuplatňuje a je potřeba ponechat původní nastavení, tj. `A=[1], b=[0]`. Pro manipulátor struktury T-bot platí X=M1+M2, Y=M1-M2. Pak je potřeba nastavit `A=[1 1; 1 -1], b=[0 0]`. Platí, že `MCS=A*ACS+b`,

kde ACS jsou polohy motorů a MCS jsou kartézké souřadnice. Matice `A` musí být čtvercová a regulární, aby bylo možné určovat polohy motorů z kartézkých souřadnic.

Poznámka 1: pokud stroj používá natáčení jen v jedné (někdy i dvou) ose, bývá výhodnější nepoužít orientaci, ale přidat natočení jako dodatečnou (aux) osu. Poznámka 2: čtyři souřadnice kvaternionu přímo mapovat na osy/motory reálně asi nejde. Pro různé simulace a testování a také proto, že u jiných kinematik se to vyskytuje, tak je to podpořeno i zde.

Vstupy

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`

Parametry

`npos`	počet použitých polohových os ↓0 ↑3 ⊙3	`Long (I32)`
`RotMode`	Volba reprezentace orientace ⊙1 `1` ..... no orientation (aux rad); stroj orientaci nepoužívá, doplňkové rotační osy se zadávají v radiánech `2` ..... quaternion; orientace se zadává kvaternionem (tj. čtveřice čísel) `3` ..... ZYX angles [rad]; otočení je zadáno pomocí 3 úhlů (otočení postupně podle osy Z, Y, X), přičemž úhly jsou v radiánech `4` ..... ZYX angles [deg]; otočení je zadáno pomocí 3 úhlů (otočení postupně podle osy Z, Y, X), přičemž úhly jsou v úhlových stupních `5` ..... no orientation (aux deg); stroj orientaci nepoužívá, doplňkové rotační osy se zadávají v úhlových stupních	`Long (I32)`
`naux`	Počet doplňkových os ↓0 ↑3	`Long (I32)`
`fcyc`	čísla/pořadí doplňkových os, které jsou rotační. Pro rotační doplňkové osy se poloha zadává v radiánech nebo ve stupních a normalizuje se na interval -1805^o až +1805^o (resp. ekvivalent v radiánech) ↓0	`Long (I32)`
`A`	transformační matice. Platí, že `MCS=A*ACS+b`. Matice musí být čtvercová, regulární a správné dimenze. ⊙`[1.0]`	`Double (F64)`
`b`	transformační vektor. Platí, že `MCS=A*ACS+b`. Vektor musí být správné dimenze. ⊙`[0.0]`	`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`

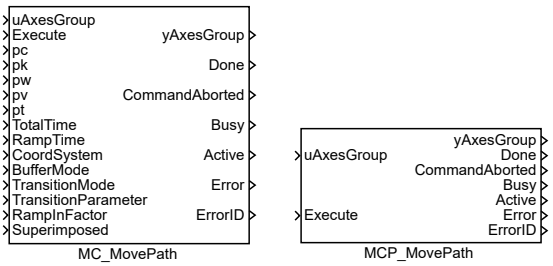
`ErrorID` Výsledek poslední operace `Error`

`i` ..... obecná chyba systému REXYGEN

MC_SetKinTransform_Arm – Nastavení kinematické transformace

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_SetKinTransform_Arm` slouží pro nastavení kinematické transformace. To je transformace mezi polohou motorů a kartezkými souřadnicemi koncového efektoru stroje. Pro různé typy strojů má tato funkce různý charakter a parametry. Tento blok slouží pro obvyklé průmyslové 6-ti osé roboty, tj. manipulátor s rotačními osami, kdy každá další osa je namontována na rameni předchozí osy a osy jsou navzájem kolmé, kromě 2. a 3. které jsou navzájem rovnoběžné. Osy 4., 5. a 6. musí společný průsečík, ostatní osymohou být mimoběžné, přičemž jejich vzdálenost udávají jednotlivé parametry. Situace je zřejmá z obrázku, kde jsou vidět i požadované polohy motorů (tj. kde je nulový úhel), aby transformace fungovala správně. Hodnoty parametrů`a1, .. d6` se zadávají ve stejných jednotkách v jakých pak budeme zadávat polohu robota (kartézké souřadnice X, Y, Z), tj. obvykle metry nebo milimetry.

Tato kinematická transformace tedy používá 3 souřadnice polohy a kvaternion nebo eulerovy úhly pro určení orientace (a nepoužívá dodatečné souřadnice). Počátek kartézkého souřadného systému je v místě, kde 1. osa protíná montážní přírubu, ale lze jej posunout do libovolného místa pomocí `base offset` transformace blokem `MC_SetCartesianTransform`. Referenční bod, pro který zadáváme MCS-souřadnice je v místě, kde 6. osa protíná přírubu pro montáž nástroje, ale lze jej posunout do libovolného místa pomocí `tool offset` transformace blokem `MC_SetCartesianTransform`.

Vstupy

`uAxesGroup`	Odkaz na skupinu os	`Reference`
`Execute`	Náběžná hrana aktivuje blok	`Bool`

Parametry

`arot`	počet jednotek polohy motorů na celou otáčku. Pokud používáme radiány zadáme 6.283 (tj. 2π), pokud používáme úhlové stupně, zadáme 360, atd. ⊙`6.28318531`	`Double (F64)`
`mrot`	počet jednotek úhlu v kartézkých souřadnicích na celou otáčku. Pokud používáme radiány zadáme 6.283 (tj. 2π), pokud používáme úhlové stupně, zadáme 360, atd. ⊙`6.28318531`	`Double (F64)`
`irt`	Volba reprezentace orientace ⊙`1` `1` ..... ZYX angles; otočení je zadáno pomocí 3 úhlů (otočení postupně podle osy Z, Y, X), přičemž jednotky úhlů definuje parametr `mrot` `2` ..... quaternion; orientace se zadává kvaternionem (tj. čtveřice čísel)	`Long (I32)`
`a1`	vzdálenost 1. osy od 2. osy ⊙`400.0`	`Double (F64)`
`a2`	vzdálenost 2. osy od 3. osy ⊙`300.0`	`Double (F64)`
`a3`	vzdálenost 3. osy od 4. osy	`Double (F64)`
`d1`	vzdálenost 2. osy od referenční roviny kolmé k 1. ose (obvykle montážní příruba). Lze také zadat 0 a počátek souřadného systému nastavit pomocí `base offset` transformace blokem MC_SetCartesianTransform.	`Double (F64)`
`d23`	vzdálenost 1. osy od 4. osy	`Double (F64)`
`d4`	vzdálenost 3. osy od 5. osy ⊙`200.0`	`Double (F64)`
`d6`	vzdálenost 5. osy od referenční roviny nástroje (obvykle montážní příruba). Lze také zadat 0 a počátek souřadného systému nastavit pomocí `tool offset` transformace blokem MC_SetCartesianTransform. ⊙`100.0`	`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`

Kapitola 23

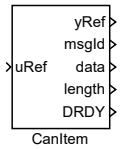
CanDrv – Komunikace po sběrnici CAN

Obsah

`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

`uRef`	Odkaz na další přijaté packety	`Reference`

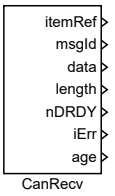
Výstupy

`yRef`	Odkaz na další přijaté packety		`Reference`
`msgId`	Číslo zprávy (COB-ID)		`Long (I32)`
`data`	Data zprávy (max. 8 bajtů, nejnižší bajt první)	↓`-9.22337E+18` ↑`9.22337E+18`	`Large (I64)`
`length`	Počet datových bajtů zprávy	↓0 ↑8	`Long (I32)`
`DRDY`	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

`filterId`	Číslo zprávy, které jsou přijaty tímto blokem	↓0 ↑536870911	`Long (I32)`
`filterIdMask`	Označuje platné bity v parametru filterId	↓0 ↑536870911	`Long (I32)`
`filterLength`	Kolik bajtů dat musí mít zpráva, aby byla akceptována tímto blokem (-1 nefitruje se podle délky dat)	↓-1 ↑8	`Long (I32)`
`RTR`	Příznak žádost o zprávu (RequestToSend)	⊙on	`Bool`
`EXT`	Rozšířený formát čísla zprávy (29bitů)	⊙on	`Bool`
`timeout`	Pokud během této doby nepřijde packet, je indikována chyba [s]	↓0.0	`Double (F64)`
`nmax`	Maximální počet zpráv přijatých blokem během jedné periody	↓1 ↑255	`Long (I32)`

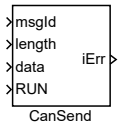
Výstupy

`itemRef`	Odkaz na další přijaté packety		`Reference`
`msgId`	Číslo zprávy (COB-ID)		`Long (I32)`
`data`	Data zprávy (max. 8 bajtů, nejnižší bajt první)	↓`-9.22337E+18` ↑`9.22337E+18`	`Large (I64)`
`length`	Počet datových bajtů zprávy	↓`0` ↑`8`	`Long (I32)`
`nDRDY`	Počet přijatých zpráv v aktuální periodě tasku	↑`255`	`Word (U16)`
`iErr`	Kód chyby		`Error`
`age`	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 podtržítky 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

`msgId`	Číslo zprávy (COB-ID)	↓0 ↑536870911	`Long (I32)`
`length`	Počet datových bajtů zprávy	↓0 ↑8	`Long (I32)`
`data`	Data zprávy (max. 8 bajtů, nejnižší bajt první)	↓-9.22337E+18 ↑9.22337E+18	`Large (I64)`
`RUN`	Povolení odeslání zprávy		`Bool`

Parametry

`RTR`	Příznak žádost o zprávu (RequestToSend)	⊙`on`	`Bool`
`EXT`	Rozšířený formát čísla zprávy (29bitů)	⊙`on`	`Bool`

Výstup

`iErr`	Kód chyby	`Error`

Kapitola 24

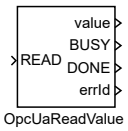
OpcUaDrv – Komunikace pomocí OPC UA

Obsah

`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 [9].

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` ..... 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	⊙`1`	`Long (I32)`

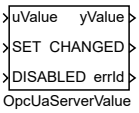
Výstupy

`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 [9].

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

`uValue`	Vstupní signál	`Any`
`SET`	Překopírování hodnoty vstupu do hodnoty OPC UA uzlu	`Bool`
`DISABLE`	Deaktivace OPC UA uzlu	`Bool`

Parametry

`NodeId`	OPC UA Node Id		`String`
`NodeId_type`	Typ id OPC UA uzlu `1` ..... string `2` ..... numeric `3` ..... guid	⊙`1`	`String`

`type`	Datový typ hodnoty `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	⊙`1`	`Long (I32)`
`BrowseName`	Atribut 'browse name' OPC UA uzlu		`String`
`Description`	Popiska OPC UA uzlu		`String`
`DisplayName`	Zobrazované jméno uzlu OPC UA uzlu		`String`
`READONLY`	Nastavení hodnoty OPC uzlu jako pouze pro čtení	⊙`on`	`Bool`

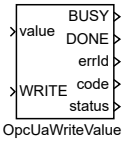
Výstupy

`yValue`	Výstupní signál	`Any`
`CHANGED`	Indikace zápisu hodnoty přes OPC UA protocol	`Bool`
`errId`	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 [9].

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

`value`	Vstupní signál	`Any`
`WRITE`	Povolení běhu algoritmu	`Bool`

Parametry

`NodeId`	OPC UA Node Id	`String`

`NodeId_type`	Typ Node Id `1` ..... string `2` ..... numeric `3` ..... guid `4` ..... cesta	⊙`1`	`Long (I32)`
`type`	Datový typ posílané hodnoty `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	⊙`1`	`Long (I32)`

Výstupy

`BUSY`	Příznak probíhající operace	`Bool`
`DONE`	Příznak dokončení transakce	`Bool`
`errId`	Kód chyby	`Error`
`code`	OPC UA výsledný stavový kód operace	`DWord (U32)`
`status`	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
Funkční bloky					
STANDARD	●	●	●	●	●
ADVANCED	●	–	●	●	●
REXLANG	●	–	●	●	●
MOTION CONTROL	●	–	○	○	●
COORDINATED MOTION	●	–	○	○	●
AUTOTUNING	–	–	○	○	●
MATRIX	●	–	○	○	●
I/O ovladače					
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_	•	
ABSROT		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		
S1OF2		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
SMTP		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 REXYGEN
-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` .... Vyrovnávací pamět pro příjem je příliš malá
`-440` .... Vyrovnávací pamět pro vysílání je příliš malá
`-441` .... Špatná hlavička
`-442` .... Server HTTP vrátil chybu
`-443` .... Server HTTP vrátil přesměrování
`-444` .... Nepřípustná blokující 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 generovaní 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

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] Modelica Association Project FMI. *Functional Mock-up Interface for Model Exchange and Co-Simulation, version 2.0*, July 2014.

[6] Python Software Foundation. Python documentation, 2019.

[7] REX Controls s.r.o.. *Ovladač MQTTDrv systému REXYGEN – Uživatelská příručka*, 2020. →.

[8] OASIS. MQTT Version 3.1.1, 2014.

[9] *Ovladač OpcUaDrv systému REXYGEN – Uživatelská příručka*.

Rejstřík

Referenční číslo dokumentace: 15831