Monarco HAT driver for the REXYGEN system (the MonarcoDrv module)

User guide

REX Controls s.r.o.

Version 3.0.2 Plzeň (Pilsen), Czech Republic 2024-09-16

Contents

1	The MonarcoDrv driver and the REXYGEN system	2
	1.1 Introduction	. 2
	1.2 Installation of the driver on the host computer	. 2
	1.3 Installation of the driver on the target device (Raspberry Pi)	. 2
2	Including the driver in the project	3
	2.1 Adding the MonarcoDrv driver	. 3
	2.2 Configuration dialog of the MonarcoDrv driver	. 3
3	Connecting the inputs and outputs in the control algorithm	7
	3.1 Direct input and output signals	. 7
	3.2 Modbus communication via RS-485	. 8
	3.3 1-Wire sensors and expansion modules	. 9
4	Troubleshooting	10
	Bibliography	11

The MonarcoDrv driver and the REXYGEN system

1.1 Introduction

This manual describes the MonarcoDrv driver for direct access to inputs and outputs of the Monarco HAT [1] within the REXYGEN system. The driver was developed by the REX Controls company.

For a quick-start and step-by-step instructions see [2].

1.2 Installation of the driver on the host computer

The MonarcoDrv driver is included in the installation package of the Development tools of the REXYGEN system. It is necessary to select the corresponding package in the installer. The REXYGEN system typically installs to the

C:\Program Files\REX Controls\REX <version> folder.

The following files are copied to the installation folder:

Bin\MonarcoDrv_H.dll - Configuration part of the MonarcoDrv driver.

Doc\PDF\ENGLISH\MonarcoDrv_ENG.pdf - This user manual.

1.3 Installation of the driver on the target device (Raspberry Pi)

If there is no RexCore runtime module installed on your Pi, install it first using the Getting started guide of REXYGEN [2]. The installation includes all necessary drivers including MonarcoDrv.

If you want to install MonarcoDrv separately, it can be done from the command line of Raspberry Pi using the command

sudo apt-get install rex-monarcodrvt

Including the driver in the project

The driver is included in the project as soon as the driver is added to the project main file and the inputs and outputs are connected in the control algorithm(s).

2.1 Adding the MonarcoDrv driver

The project main file with the MonarcoDrv driver included is shown in Figure 2.1. There is one block which must be added to the project to include the driver. A block of type IODRV renamed to MNR and connected to the Drivers output of the main EXEC block. The name of this block (MNR, see Fig. 2.1), is the prefix of all input and output signals provided by this driver. The four most important parameters are:

- module name of the module linked to the driver, in this case MonarcoDrv the name
 is CASE SENSITIVE!
- classname class of the driver, in this case MonarcoHatDrv the name is CASE
 SENSITIVE!
- cfgname name of the driver configuration file, e.g. monarcohat.rio
- factor multiple of the EXEC block's tick parameter defining the driver's task execution period

The above mentioned parameters of the IODRV function block are configured in REXY-GEN Studioprogram. The configuration dialog is shown also in Fig. 2.1.

The Configure button opens the configuration dialog of the MonarcoDrv driver, which is described in chapter 2.2.

2.2 Configuration dialog of the MonarcoDrv driver

The configuration dialog can be activated from REXYGEN Studio by pressing the Configure button in the parameters dialog of the IODRV block (see chapter 2.1).

The RS-485 section configures the properties of the RS-485 bus for connecting to external devices.

IMPORTANT: Note that the internal communication between the CPU and the MCU of the Monarco HAT uses UART communication (/dev/ttyAMAO on the Raspberry Pi) at 115200 baudrate, 8 bits per byte, no parity, 1 stop bit. These are the values you have to use in the Modbus driver configuration if you are willing to use it. For details about the internal structure of the Monarco HAT visit http://www.monarco.io.

The SPI configuration section defines the communication between the CPU and the Monarco HAT MCU (/dev/spidev0.0 on the Raspberry Pi, maximum clock speed is 4 MHz).

Image: Income settings Window Help Image: Income setings Image: Income settings	₩ monarco_exec.mdl - REXYGEN Studio - [monarco_exec] — □ ×																		
Import to the second		dit 1	View	Projec	t large	t loc		Setting	s Windo	w Help) Sa (Ba la fa	l ni s	ħ. :≡ ₩4		n				
Modules Block: Block type: Drives MNR exedib\UODRV Block type description: REXYGEN input/output driver Open block documentation Toggle quick reference Parameters Options Scalar parameters Scalar parameters I would and the monarcoDruk string VO driver class name 3 cfgname MonarcoDruk string VO driver class name 4 factor 1 long Execution file name 4 factor 1 long Execution factor 5 stack 10240 long Stack size [bytes] 6 pri 3 long Driver thread logical priority 7 timer off bool Driver is a source of time	mo	narco	_exec	90 1			Blo	ck prop	erties	•	6	0 - 0 4			•			?	×
Level0 prev nextmonarco_task Level2 Scalar parameters Level3 Parameter Value Type Description 1 module MonarcoDrv string 2 classname 3 cfgname 4 factor 1 long 5 stack 10240 long 5 stack 10240 long 5 stack 10240 long 5 stack 10240 long 0 pri 3 long 0 priver thread logical priority 7 timer 0 off bool Driver is a source of time module: Module name	Mo	dules)rivers - chives	> 	► <u>prev</u> MN	nextÞ R	B B B C	Block name: Block type: MNR Exectib \tODRV Block type description: REXYGEN input/output driver Open block documentation												
Level2 1 module MonarcoDrv string Module name 2 classname MonarcoDrv string I/O driver class name 3 cfgname monarcohat.rio string Configuration file name 4 factor 1 long Execution factor 5 stack 10240 long Stack size [bytes] 6 pri 3 long Driver thread logical priority 7 timer off bool Driver is a source of time		evel0 -		prev monaro	next> o_task	P	Parameters Options Style Scalar parameters												
Level3 2 classname MonarcoHatDrv string I/O driver class name EXEC 3 cfgname monarcohat.rio string Configuration file name 4 factor 1 long Execution factor 5 stack 10240 long Stack size [bytes] 6 pri 3 long Driver thread logical priority 7 timer off bool Driver is a source of time	L	Level2>					1 module		e	Monar	rcoDrv	string	Туре	Module	e nam	le	lon		4
EXEC	. L	Level3>				2 classname				coHatDrv	string string		I/O driv	driver class name figuration file name					
5 stack 10240 long Stack size [bytes] 6 pri 3 long Driver thread logical priority 7 timer off bool Driver is a source of time	E	EXEC					4	factor		1		long		Executi	on fa	tor			
6 pri 3 long Driver thread logical priority 7 timer off bool Driver is a source of time							5 stack					long		Stack si	ize [b ₎	/tes]			
module: Module name							6 pri 7 timer					long		Driver t	hread	l logical			
						n	nodu	ule: Modu	ile name										
Configure OK Cancel) Co	onfigure]						[OK		Canc	el

Figure 2.1: An example of project main file with the MonarcoDrv driver included

🐸 Monarco Driver	— D X
MONARCO	RS-485 Configuration Baudrate 9600 Parity none Data bits 8 Stop bits one SPI Configuration SPI Device //dev/spidev0.0 SPI Clock 4000000 OK Cancel

Figure 2.2: Monarco HAT configuration dialog

Connecting the inputs and outputs in the control algorithm

The inputs and outputs of the driver must be interconnected with the individual tasks (.mdl files). The individual tasks (QTASK or TASK blocks) are connected to the QTask, Level0,..., Level3 outputs of the main EXEC block.

3.1 Direct input and output signals

The inputs and outputs of the MonarcoDrv driver can be accessed as shown in Fig. 3.1.

One block of the From type allowing the user to read one input has the Goto tag set to MNR__DI1, while the other has this tag set to MNR__DI2. The number in the flag corresponds with the terminal pinout. The block of Goto type allowing the user to set (write) one output has the Goto tag) set to MNR__DO3, the other output is accessed via the MNR__DO4 flag. The blocks always have the MNR prefix right at the beginning of the tag followed by two _ characters (underscore).

Similarly for other pins we can use e.g. the flags:

- Goto, MNR__DO4 digital output 4
- Goto, MNR__AO1 analog output 1
- From, MNR__AI1 analog input (voltage mode 0..10V)
- From, MNR__AI2C analog input (current mode 0..20mA)
- Goto, MNR__LED5 Onboard LED 5
- Goto, MNR__LED5_Mask Only when TRUE, onboard LED5 can be controlled from algorithm

In general, the link to a particular signal consists of the driver name MNR, two underscore characters __, signal reference, signal number and an optional symbol defining the mode of the input. The terminal numbering of the Monarco HAT is shown in Fig. 3.2.



Figure 3.1: Example of input and output flags of the MonarcoDrv driver



Figure 3.2: Terminal pinout for Monarco HAT.

The installation of the REXYGEN system includes a library of examples, where among other things, the section 0121_Monarco_HAT is dedicated to the use of MonarcoDrv. The example 0121-00_I0_Flags contains a library of usable inputs and outputs.

3.2 Modbus communication via RS-485

The RS-485 bus provides a standard interface to communicate with external devices (servo drives, energy meters, etc.) or to expand the I/O capabilities of the Monarco HAT

itself. Modbus communication is typically used. There is a separate driver for Modbus communication in the **REXYGEN** system, see [3].

IMPORTANT: Note that the internal communication between the CPU and the MCU of the Monarco HAT uses UART communication (/dev/ttyAMAO on the Raspberry Pi) at 115200 baudrate, 8 bits per byte, no parity, 1 stop bit. These are the values you have to use in the Modbus driver configuration if you are willing to use it. For details about the internal structure of the Monarco HAT visit http://www.monarco.io.

Figure 3.3 shows an example project main file with multiple I/O drivers.

3.3 1-Wire sensors and expansion modules

The 1-Wire bus provides additional interface to expand the I/O capabilities of the Monarco HAT itself (e.g. using 1-Wire temperature sensors, relative humidity sensors, relay modules etc.). There is a separate driver for 1-Wire communication in the REXYGEN system, see [4].

Figure 3.3 shows an example project main file with multiple I/O drivers.



Figure 3.3: A project with MonarcoDrv, 1-Wire, Modbus RTU and Modbus TCP drivers

Troubleshooting

First and foremost, it's advisable to explore the library of examples, especially the section 0121_Monarco_HAT, which pertains to the usage of MonarcoDrv.

In the case that the diagnostic tools of the REXYGEN system (e.g. REXYGEN Studio@) report unexpected or incorrect values of inputs or outputs, it is desirable to test the functionality outside the REXYGEN system (command line tools, simple scripts, etc.). Also double check the configuration – the most common problems include:

Hardware problem – incorrect wiring.

Internal communication problem – SPI bus, I2C bus or UART is occupied by another service or program.

In the case that the given input or output works with other software tools and does not work in the REXYGEN system, report the problem to us, please. E-mail is preferred, reach us at support@rexygen.com. Please include the following information in your description to help us process your request as soon as possible:

- Identification of the REXYGEN system you are using. Simply export it to a file using the REXYGEN Studio program (Target → Licensing → Export).
- Short and accurate description of your problem.
- The configuration files of the REXYGEN system (.mdl and .rio files) reduced to the simplest case which still demonstrates the problematic behavior.

Bibliography

- [1] REX Controls s.r.o.. Internet webpage www.monarco.io, 2020.
- [2] REX Controls s.r.o.. Getting started with REXYGEN and Monarco HAT, 2020. \rightarrow .
- [3] REX Controls s.r.o.. Modbus driver of REXYGEN User guide, 2020. \rightarrow .
- [4] REX Controls s.r.o.. OwsDrv driver of REXYGEN for 1-Wire devices user guide, 2020. →.

Documentation reference number: 16749