JC-350

Version Update

from V. 1.09 to V. 1.10

Version Update

Jetter

Revision 1.01
August 2011 / Printed in Germany

Table of Contents

1 Introduction

Introduction

This chapter shows the history of OS versions for the controller JC-350.

Operating System Update - Why?

An OS update allows you to:

- add new functions to your controller
- fix software bugs
- make sure your controller is working with a definite OS version, for example, if a definite OS version has been released for a certain customer

Contents

Operating System Update

OS File for Updating the Operating System

For updating the OS the following file is needed:

OS File	Description
JC-350_1.10.0.00.os	OS file for JC-350 with version 1.10

Downloading the OS File

Jetter AG make operating system files available for download from their **homepage at http://www.jetter.de**. OS files can be found in the support area or on the page of the JC-350 controller via quicklink.

Operating System Update by means of JetSym

To update your OS proceed as follows:

Step	Action
1	Download the OS file from www.jetter.de
2	Establish a connection between PC and controller
3	In JetSym: Select menu item "Build -> Update OS" or Click on the button "OS Update" in the CPU window of the hardware manager
4	Select the OS File
5	Initiate the OS update by clicking OK
6	**Result:** Following Power OFF / Power ON the new OS is launched.

Minimum Requirements

For programming a JC-350 with version 1.10 JetSym 4.2 or higher is required.

JC-350 Version Update - Overview

V 1.04

The following table gives an overview of newly added features and fixed software bugs in OS version 1.04:

Function	New	Fixed
JX2 system bus:		
Register overlaying for digital inputs/outputs	✓	
Support of JX-SIO modules and third-party CANopen® devices	✓	
JX3 system bus:		
Register overlaying for digital inputs/outputs	✓	
System bus special registers for status and control	✓	
Operating system update:		
Via FTP: On completion notification the OS has actually been stored.		✓
Updating a JX2 slave module while registers are being accessed blocks communication		✓
Application program:		
Task switch could fail to happen		✓
Error signal in case of invalid file "/app/start.ini"		✓
Display commands:		
Redirection to JX2-SER1 works only if JX2-PRN1 has been configured, too		✓

V 1.05

The following table gives an overview of newly added features and fixed software bugs in OS version 1.05:

Function	New	Fixed
JX2 system bus: V1.05.0.00		
AS interface gateway BWU1821 is supported	✓	
Frequency inverter 8200 vector is supported	✓	
JetMove 1xx is not detected during boot process		✓
Automatic baud rate recognition does not work reliably for some of the baud rates and configurations of IP67 modules.		✓
Repetition counter does not work when polling I/O modules		✓
AutoCopy function:		
Automatic Copying of Controller Data		
Application program:	✓	
Pending cyclic tasks are started immediately after Taskunlock	✓	
For function pow(x,y) a floating point number can be entered as exponent	✓	

Function	New	Fixed
Cyclic tasks can be debugged	✓	
Length of project and program names > 39 characters		✓
Restart of an elapsed timer		✓
The value returned by DateTimeDecode() was always 1 day short of the actual day.		✓
DateTimeEncode and -IsValid might return the value TRUE irrespective of an invalid date		✓
User registers:		
The register type can be set up without having to start the application program	✓	
Displays and HMIs:		
A floating point value can be used as default for UserInput	✓	
The default value for UserInput is not displayed correctly		✓
It is not possible to enter LED register numbers		✓

V 1.08

The following table gives an overview of newly added features and fixed software bugs in OS version 1.08:

Function	New	Fixed
System configuration:		
System rights for configuration file	✓	
JX2 system bus: V1.11.0.00		
Timeout after CAN-PRIM message		✓
Registers of LJX7-CSL modules		✓
Write access to analog outputs of CANopen® modules		✓
State of digital inputs when the controller is powered on		✓
Digital outputs on JX-SIO or CANopen® modules		✓
Input/output 64 on JX-SIO or CANopen® modules		✓
User-programmable CAN Interface		✓
Application program:		
NetCopyList Functions	✓	
StrCopy()		✓
Crash in the case of "invalid" application program		✓
NetCopyVarFromReg()		✓
JX3 system bus:		
Module registers for digital I/Os	✓	
Displays and HMIs:		
UserInput()		✓

V 1.09

The following table gives an overview of newly added features and fixed software bugs in OS version 1.09:

Function	New	Fixed
System:		
System command register	✓	
JX2 system bus: V1.13.0.00		
Status change of inputs on JX2-ID8		✓
Status change of fast inputs		✓
Application program:		
FTP Client	✓	
Axis instructions		✓
Taskrestart in the case of Delay()		✓
Crash in the case of missing library		✓
Floating-point number registers in data files		✓
NetCopyVarToReg with floating-point number registers		✓
JX3 system bus:		
Dummy Modules	✓	
AutoCopy:		
FTP commands	✓	
Serial interface:		
Initialization after booting		✓

V 1.10

The following table gives an overview of newly added features and fixed software bugs in OS version 1.10:

Function	New	Fixed
System:		
LED registers		✓
SD memory card		✓
JX2 system bus: V1.17.0.00		
Additional modules	✓	
CAN-PRIM	✓	
Application program:		
Task Commands With Variable Parameters	✓	
UserInput()		✓
NetCopyListSend()		✓
Task status register		✓

Function	New	Fixed
Real-time clock:		
Additional register for milliseconds	✓	
User-programmable IP Interface:		
More connections	✓	

2 New Features

Introduction

This chapter describes the features which have been added or enhanced in the new software release.

Contents

2.1 Various New Features and Modifications

Introduction

This chapter covers the new features and modifications

Contents

RTC Registers: Milliseconds

Milliseconds

New registers containing the milliseconds of the current time have been added to the RTC register sets.

Overview of Registers

Register set # 1: Direct access

Registers	Description
R 102910	Milliseconds
R 102911	Seconds
R 102912	Minutes
R 102913	Hours
R 102914	Weekday (0 = Sunday)
R 102915	Day
R 102916	Month
R 102917	Year

Register set # 2: Buffer access

Registers	Description
R 102920	Milliseconds
R 102921	Seconds
R 102922	Minutes
R 102923	Hours
R 102924	Weekday (0 = Sunday)
R 102925	Day
R 102926	Month
R 102927	Year
R 102928	Read/write trigger

Multiple Connections of the User-Programmable IP Interface

Obsolete Technical Data

Function	Description
Number of Connections	10

New Technical Data

Function	Description
Number of Connections	20

Reason for this change Enabling multiple connections to be open at the same time.

JX3-MIX1 and JX3-MIX2 Support

Introduction

During the boot process of the controller JC-350 it detects the latest modules JX3-MIX1 and JX3-MIX2 connected to the system bus. Then, it initializes these modules and allows access to their registers and inputs/outputs.

Controlling the Serial Interface

Besides access to registers of the serial interface on the JX3-MIX2 module, the controller JC-350 supports controlling the serial interface by means of the instructions `DisplayText()`, `DisplayText2()`, and `DisplayValue()`.

Device Number

Device number which must be addressed by the *DISPLAY* instruction in order to output information via serial interface of the JX3-MIX2 module.

Module	Interface	Device Number
JX3-MIX2	Serial interface on the module	11

Module Number

The module number to be entered is calculated based on the number of the module on the system bus plus a constant value considering the system bus:

```
Module number = number of the module + system bus constant
```

System Bus	System Bus Constant
JX3	100

222838

Module number - serial interface module

This register contains the number of the module to which the display instruction is redirected (device # 11).

Module register properties	
Values (JX3 bus)	102 ... 117
Takes effect	when the next Display instruction is issued

Support of Additional Festo® Modules on the JX2 System Bus

Introduction

The controller JC-350 is able to automatically detect and commission additional modules by Festo AG & Co.:

- MTR-DCI
- SFC-DC
- SFC-LAC
- SFC-LACI
- CPX-CMAX
- CPX-CMPX

Application Note

For more information on how to operate these modules along with a controller JC-350 refer to the following application notes:

Modules	Document
MTR / SFC	Festo_apn042_xxx_Festo_Electrical_Motor_Controllers.pdf
CPX	Festo_apn043_xxx_CPX_Technologiemodule.pdf

Task Commands With Variable Parameters

Obsolete Function

The task control instructions covered by the STX language can be programmed using constant task names as parameters:

- `Taskbreak <Taskname>`
- `Taskcontinue <Taskname>`
- `Taskrestart <Taskname>`

New Function

New task control functions have been added allowing variable parameters to be used as Task ID. The functionality corresponds to that of the existing instructions. The existing instructions remain available:

- `Function TaskBreakById (TaskId: int): int;`
- `Function TaskContinueById (TaskId: int): int;`
- `Function TaskRestartById (TaskId: int): int;`

Function Parameters

The ID of the task to be controlled must be specified as function parameter (`TaskId`).

Returned Value

This function will return one of the following values:

Returned Value	
0	Function completed
-1	invalid task ID

Reason for this Change

The new function allows programmers to write programs where the task to be controlled becomes known only at runtime.

Prerequisites

The new task functions can be used starting from JetSym version 4.3.

2.2 User-Programmable CAN-PRIM Interface

The CAN-PRIM Interface

The user-programmable CAN-PRIM interface offers the possibility of transmitting and receiving any CAN messages. Processing the CAN messages is done in the application program exclusively.

Applications, e.g.

The following applications can be carried out with the help of the user-programmable CAN-PRIM interface:

- Connection of modules via CAN interface
- Connection of modules via CANopen® interface
- ...

Substantial Demands on the Programmer

The functions of the user-programmable CAN-PRIM interface require basic knowledge of the Controller Area Network CAN. Some of them are:

- Structure of a CAN message
- CANopen® services

Contents

User-Programmable CAN-PRIM Interface - Operating Principle

Operating Principle

The user-programmable CAN-PRIM interface uses message boxes for data exchange between CAN bus and application program. Each message box is able to accomodate a complete CAN message.

16 message boxes are available to the user. Each of these boxes can be configured either as inbox or as outbox with a specific CAN ID.

Technical Specifications

Function	Description
CAN ID	11-bit or 29-bit
Number of message boxes	16

Enabling the User-Programmable CAN-PRIM Interface

The CAN-PRIM interface is enabled using bit 2 or bit 3 in register 200002077 *JX2 System Bus Special Functions*.

Restrictions Regarding the CAN-PRIM Interface

Number of Connectable Modules

When using the user-programmable CAN-PRIM interface, the following restrictions apply:

- If 29-bit CAN identifiers are used, the serial number of non-intelligent JX2-I/O module must start with 2.

CAN Messages During Boot Phase

CAN modules connected to the system bus must not send CAN messages during the boot phase of the system bus.

Time Response

The interval between two CAN messages received via CAN-PRIM interface must be at least 10 ms. If the interval is shorter, the controller JC-350 is not able to receive all CAN messages.

Earmarked CAN IDs

When peripheral modules are simultaneously operated on the JX2 system bus and the CAN-PRIM interface, certain CAN IDs are earmarked.

Modules on the system bus	Earmarked CAN IDs
For all modules	0x100, 0x701 - 0x70A, 0x732 - 0x73B, 0x746 - 0x74F
JX2-I/O modules	0x180 - 0x19F, 0x1A0 - 0x1BF, 0x380 - 0x39F, 0x3A0 - 0x3BF
JX2-Slave modules	0x081 - 0x090, 0x09F - 0x0AF, 0x161 - 0x16F, 0x1D1 - 0x1DF
JX3 Modules	0x180 - 0x19F, 0x1A0 - 0x1BF, 0x320 - 0x33E, 0x380 - 0x39F, 0x3A0 - 0x3BF, 0x3E0 - 0x3FE
JX-SIO and CANopen® modules	0x1C6 - 0x1CF, 0x246 - 0x24F, 0x2C6 - 0x2CF, 0x346 - 0x34F, 0x3C6 - 0x3CF, 0x446 - 0x44F, 0x4C6 - 0x4CF, 0x581 - 0x58A, 0x5B2 - 0x5BB, 0x5C6 - 0x5CF, 0x601 - 0x60A, 0x632 - 0x63B, 0x646 - 0x64F, 0x732 - 0x73B, 0x746 - 0x74F
Festo CP-FB modules	0x010, 0x110, 0x120, 0x130, 0x140, 0x150, 0x1E0, 0x1F0, 0x250, 0x260, 0x270, 0x350, 0x360, 0x370, 0x3B0
LioN-S modules	0x2E0 - 0x2FE, 0x360 - 0x37E, 0x581 - 0x5A0, 0x601 - 0x620, 0x701 - 0x720
BWU1821	0x281 - 0x29F, 0x301 - 0x31F, 0x481 - 0x49F, 0x501 - 0x51F, 0x5C6 - 0x5CF, 0x646 - 0x647, 0x746 - 0x74F
LJX7-CSL	0x481 - 0x49F, 0x501 - 0x51F, 0x581 - 0x5A0, 0x601 - 0x620, 0x701 - 0x720

Programming the CAN-PRIM Interface

Registers for Configuring the JX2 System Bus

Activate the CAN-PRIM interface to be able to use it.

Registers	Description
R 200002029	JX2 System Bus - Baud Rate
R 200002077	JX2 system bus special functions

Registers for Configuring the CAN-PRIM Interface

Registers	Description
R 200010500	CAN-PRIM status
R 200010501	CAN-PRIM command register
R 200010503	FIFO buffer filling level
R 200010504	FIFO data
R 200010506	Global receiving mask
R 200010507	Global receive ID

Registers for Message Boxes of the CAN-PRIM Interface

20 registers with identical functions are assigned to each message box. The register number of individual message boxes is calculated from the base register number and the message box number.

Registers	Description
R 200010530 + Box*20	Box status
R 200010531 + Box*20	Box configuration
R 200010532 + Box*20	CAN ID
R 200010533 + Box*20	Number of data bytes
R 200010534 + Box*20	Data byte 0
R 200010535 + Box*20	Data byte 1
R 200010536 + Box*20	Data byte 2
R 200010537 + Box*20	Data byte 3
R 200010538 + Box*20	Data byte 4
R 200010539 + Box*20	Data byte 5
R 200010540 + Box*20	Data byte 6
R 200010541 + Box*20	Data byte 7
R 200010542 + Box*20	CAN ID mask
R 200010543 + Box*20	Box command
R 200010544 + Box*20	Received CAN ID

Initialization

To initialize the CAN-PRIM interface proceed as follows:

Step	Action
1	Set bit 2 = 1 or bit 3 = 1 in R 20002077 *JX2 system bus special functions*.
2	Launch the system bus.

3	Configure the CAN ID length for all message boxes
	
	If CAN ID length... / ... Then ...
	is 11 bits / R 200010501 := 8;
	is 29 bits / R 200010501 := 9;

Configuring a Message Box for Sending Messages

To configure a message box for sending messages proceed as follows:

Step	Action
1	Select a message box. In this manual message box 0 is used.
2	Configure message box 0 as outbox: R 200010531 := 1;
3	Configure the CAN ID for sending messages R 200010532 := CAN ID;
4	Activate message box 0: R 200010543 := 1; **Result if configuration was successful:** Bit 0 = 1 in R 200010530

Sending a CAN Message

To send a CAN message proceed as follows:

Step	Action
1	Select a message box. In this manual message box 0 is used.
2	Enter the number of bytes to be sent: R 200010533 := Number of bytes;
3	Enter the content into the data bytes to be sent: R 200010534 := Data byte 0; R 200010535 := Data byte 1; ... R 200010541 := Data byte 7;
4	Start transmission of the CAN message: R 200010543 := 3; **Result if sending was successful:** Bit 3 = 0.00 mm R 200010530

Configuring a Message Box for Receiving

To configure a message box for receiving messages proceed as follows:

Step	Action
1	Select a message box. In this manual message box 1 is used.
2	Configure message box 1 as inbox: R 200010551 := 0;
3	Configure the CAN ID for receiving messages R 200010552 := CAN ID;
4	Activate message box 1: R 200010563 := 1; **Result if configuration was successful:** Bit 0 = 1 in R 200010550

Receiving a CAN Message

To receive a CAN message proceed as follows:

Step	Action
1	Check bit 1 NEWDAT in R 200010500 **If ...** / **... Then ...** Bit 1 = 1 in R 200010500 / a CAN message has been received. Proceed with step 2
2	Read the number of the message box which has received a CAN message. Box := R 200010504;
3	Check the message box for overflow. **If ...** / **... Then ...** Bit 2 = 1 in R 200010530 + Box*20 / an overflow has occurred.
4	Read the number of received bytes Number of bytes = R 200010533 + Box*20;
5	Read the received bytes Data byte 0 = R 200010534 + Box*20; Data byte 1 = R 200010525 + Box*20; ... Data byte 7 = R 200010541 + Box*20;
6	Acknowledge that the message has been received R 200010543 + Box*20 := 4; **Result if message was successfully received:** Bit 1 = 0 in R 200010530 + Box*20

Internal Processes of the CAN-PRIM Interface

Introduction

The CAN-PRIM interface processes the following tasks independently:

- Reception of CAN messages
- Sending of CAN messages
- Filtering of CAN messages on reception

Internal Reception of CAN Messages

The CAN-PRIM interface receives new messages in the following way:

Level	Description
1	The CAN bus receives a valid CAN message.
2	The CAN ID matches the receiving mask.
3	The CAN ID matches the CAN ID of a message box which has been configured as inbox.
4	**If in R 200010510 + Box*20 of the message box ...** / **... Then ...** the NEW DAT bit = 0 / the NEW DAT bit switches to 1 proceed with step 5 the NEW DAT bit = 1 / the OVERRUN bit switches to 1; CAN message data are discarded.
5	R 200010503 *FIFO filling level* is incremented.
6	The message box number is entered into R 200010504 *FIFO data*.
7	In R 200010500 *CAN-PRIM Status* the NEW DAT bit is set to 1.

Register Description - CAN-PRIM Interface

R 200002077

Special functions - JX2 system bus

R 200002077 is used to enable or disable various special functions of the JX2 system bus.

Meaning of the individual bits		
Bit 2	**Enabling the user-programmable CAN-PRIM interface**	
	1 =	The CAN-PRIM interface is enabled following the next launch of the JX2 system bus. This function allows to connect expansion modules.
Bit 3	**Enabling the CAN-PRIM interface only**	
	1 =	The CAN-PRIM interface is enabled following the next launch of the JX2 system bus. Expansion modules cannot be connected.
Bit 4	**CAN IDs 0x081 ... 0x09F at the CAN-PRIM interface**	
	1 =	The CAN-PRIM interface allows communication with the CAN IDs 0x081 ... 0x09F.

R 200010500

CAN-PRIM status

R 200010500 allows to evaluate the status of the CAN-PRIM interface.

Meaning of the individual bits		
Bit 1	**NEW-DAT**	
	1 =	At least one message box has received a new CAN message.
Bit 2	**ID length**	
	0 =	The length of sent/received CAN IDs is 11 bits
	1 =	The length of sent/received CAN IDs is 29 bits

Module register properties	
Type of access	Read access
Takes effect	if the CAN-PRIM interface is enabled.

R 200010501

CAN-PRIM command register

R 200010501 is used to transfer certain commands to the CAN-PRIM interface.

CAN-PRIM Interface - Commands	
7	**Clearing the FIFO buffer** This command is for clearing all entries in the FIFO buffer. Result: R 200010503 = 0
8	**Setting the default ID length to 11 bits** The ID length for all CAN messages is set to 11 bits. Result: Bit 2 = 0 in R 200010500 R 200010506 = 0 R 200010507 = 0 R 200010542+Box*20 := 0x7FF (in all boxes)
9	**Setting the default ID length to 29 bits** The ID length for all CAN messages is set to 29 bits. Result: Bit 2 = 1 in R 200010500 R 200010506 = 0 R 200010507 = 0 R 200010542+Box*20 := 0x1FFFFFFF (in all boxes)
10	**Checking message boxes for new messages** The CAN-PRIM interface automatically checks the inbox for new messages. Command 10 is for extending the interval between checks.

Module register properties	
Takes effect	if the CAN-PRIM interface is enabled.

R 200010503

FIFO buffer filling level

R 200010503 shows whether new CAN messages have been received, as well as the number of messages.

Module register properties		
Values	Number of received messages:	0 ... 16
Type of access	Read access	
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010504

FIFO data

R 200010504 shows which of the messages boxes has received a new CAN message. Read access to R 200010504 removes the value which has been read last from the FIFO buffer. This access decrements the value of R 200010503 by one.

Module register properties		
Values	No FIFO data available:	-1
	Number of the message box containing new data:	0 ... 15
Type of access	Read access removes characters	
Value after reset	-1	
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010506

Global receiving mask

The global receiving mask is for filtering the bits of the received CAN ID. If the bit of the global receiving mask is set, the received bit of the CAN ID is compared with the global receiving ID.

Module register properties		
Values	in the case of 11-bit CAN IDs	0 ... 0x7FF
	in the case of 29-bit CAN IDs	0 ... 0x1FFFFFFF
Bit = 0	Bit is not compared with R 200010507	
Bit = 1	Bit is compared with R 200010507	
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010507

Global receive ID

The global receiving ID and R 200010506 *Global receiving mask* are for setting a CAN ID range which is then forwarded to the CAN-PRIM interface.

Module register properties		
Values	in the case of 11-bit CAN IDs	0 ... 0x7FF
	in the case of 29-bit CAN IDs	0 ... 0x1FFFFFFF
Takes effect	if the CAN-PRIM interface is enabled.	

CAN Message Box - Description of Registers for Direct Access

Direct Access

For programming purposes, always use registers for direct access to message boxes. 20 registers with identical functions are assigned to each message box. The registers of individual message boxes start from a certain basic register number.

Message Box	Basic Register Number
0	R 200010530
1	R 200010550
2	R 200010570
3	R 200010590
4	R 200010610
5	R 200010630
6	R 200010650
7	R 200010670
8	R 200010690
9	R 200010710
10	R 200010730
11	R 200010750
12	R 200010770
13	R 200010790
14	R 200010810
15	R 200010830

R 200010530+Box*20

Box status

R 200010530+Box*20 allows to evaluate the status of the message box.

Meaning of the individual bits		
Bit 0	**Valid**	
	1 =	The message box is enabled
Bit 1	**NEW-DAT**	
	1 =	The message box has received a CAN message. Reception of additional CAN messages is blocked.
Bit 2	**OVERRUN**	
	1 =	The message box has received a new CAN message while NEW-DAT was 1.
Bit 3	**Sending error**	
	1 =	An error has occurred when sending a CAN message from this message box.

Module register properties	
Type of access	Read access
Takes effect	if the CAN-PRIM interface is enabled.

R 200010543+Box*20

Box Command Register

R 200010543+Box*20 is used to transfer certain commands to the message box.

CAN-PRIM Interface - Commands	
1	**Enabling the message box** The message box is enabled. When enabling the message box, the system checks whether the CAN ID of the box is reserved by the JX2 system bus or not. Result: Bit 0 = 1 in R 200010530+Box*20 (if the CAN ID is not reserved)
2	**Disabling the message box** The message box is disabled. Result: Bit 0 = 0 in R 200010530+Box*20
3	**Sending CAN messages** A CAN message is sent.
4	**Clearing the NEW DAT bit** This command is for clearing the NEW DAT bit in R 200010530+Box*20 which enables the selected message box to receive CAN messages again. Result: Bit 1 = 0 in R 200010530+Box*20 If for all message boxes the NEW DAT bit is 0, bit 1 in R 200010500 is set to 0.
5	**Clearing the OVERRUN bit** This command is for clearing the OVERRUN bit in R 200010530+Box*20 of the selected message box. Result: Bit 2 = 0 in R 200010530+Box*20
6	**Clearing the transmission error bit** This command is for clearing the transmission error bit in R 200010530+Box*20 of the selected message box. Result: Bit 3 = 0 in R 200010530+Box*20

Module register properties	
Takes effect	if the CAN-PRIM interface is enabled.

R 200010531+Box*20

Box Configuration

R 200010531+Box*20 is for configuring the message box.

Configuration Values	
0	**Inbox** For configuring the box as inbox
1	**Outbox** For configuring the box as outbox for standard frames
2	**Outbox RTR** For configuring the box as outbox for RTR frames

Module register properties	
Takes effect	if the CAN-PRIM interface is enabled.

R 200010532+Box*20

CAN ID

In the case of an outbox, a CAN message is sent using the CAN ID.
In the case of an inbox, CAN messages with this CAN ID - which is masked by the CAN ID mask - are received.

Module register properties		
Values	in the case of 11-bit CAN IDs	0 ... 0x7FF
	in the case of 29-bit CAN IDs	0 ... 0x1FFFFFFF
Takes effect	if the CAN-PRIM interface is enabled and the message box is disabled, i.e. if in R 200010530+Box*20 bit 0 = 0.	

R 200010542+Box*20

CAN ID mask

The CAN ID mask can be used to configure which bits of a received CAN ID are compared with the configured CAN ID of the message box.

Module register properties		
Values	Bit = 0	Bit is not compared with CAN ID
	Bit = 1	Bit is compared with CAN ID
Takes effect	if the CAN-PRIM interface is enabled	

R 200010544+Box*20

Received CAN ID

In the case of an inbox, the CAN IDs of received CAN messages are entered here.

Module register properties		
Type of access	Read access	
Values	in the case of 11-bit CAN IDs	0 ... 0x7FF
	in the case of 29-bit CAN IDs	0 ... 0x1FFFFFFF
Takes effect	if the CAN-PRIM interface is enabled	

R 200010533+Box*20

Number of data bytes

In the case of an outbox, a CAN message is sent with this number of data bytes.
In the case of an inbox, the number of received data bytes is entered.

Module register properties		
Values	Number of data bytes:	0 ... 8
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010534 ...
R 200010541+Box*20

Data bytes 0 through 7

In the case of an outbox, a CAN message is sent with these data bytes.
In the case of an inbox, the received data bytes are entered.

Module register properties		
Values	Data of data bytes:	0 ... 255
Takes effect	if the CAN-PRIM interface is enabled.	

CAN Message Box - Description of Registers for Indirect Access

Indirect Access

To get indirect access to message boxes of the CAN-PRIM interface always select the message box using R 200010502 "Message Box Number".

To allow compatibility with previous OS versions the registers for indirect access are still supported. Always use the registers for direct access when programming the CAN-PRIM interface.

R 200010501

CAN-PRIM command register

R 200010501 is used to transfer certain commands to the CAN-PRIM interface.

CAN-PRIM Interface - Commands	
1	**Enabling the message box** The selected message box in R 200010502 is enabled. When enabling the message box, the system checks whether the CAN ID of the box is reserved or not. Result: Bit 0 = 1 in R 200010510
2	**Disabling the message box** The selected message box in R 200010502 is disabled. Result: Bit 0 = 0 in R 200010510
3	**Sending CAN messages** A CAN message is sent containing the data of the selected message box.
4	**Clearing the NEW DAT bit** This command is for clearing the NEW DAT bit in R 200010500 which enables the selected message box to receive CAN messages again. Result: Bit 1 = 0 in R 200010510
5	**Clearing the OVERRUN bit** This command is for clearing the OVERRUN bit in R 200010510 of the selected message box. Result: Bit 2 = 0 in R 200010510
6	**Clearing the transmission error bit** This command is for clearing the transmission error bit in R 200010510 of the selected message box. Result: Bit 3 = 0 in R 200010510
7	**Clearing the FIFO buffer** This command is for clearing all entries in the FIFO buffer. Result: R 200010503 = 0
8	**Setting the default ID length to 11 bits** The ID length for all CAN messages is set to 11 bits. Result: Bit 2 = 0 in R 200010500 R 200010506 = 0 R 200010507 = 0

CAN-PRIM Interface - Commands	
9	**Setting the default ID length to 29 bits** The ID length for all CAN messages is set to 29 bits. Result: Bit 2 = 1 in R 200010500 R 200010506 = 0 R 200010507 = 0
10	**Checking message boxes for new messages** The CAN-PRIM interface automatically checks the inbox for new messages. Command 10 is for extending the interval between checks.

Module register properties	
Takes effect	if the CAN-PRIM interface is enabled.

R 200010502

Message box number

R 200010502 is for selecting a message box. The data contained in the message box can then be accessed via module registers R 200010510 through R 200010521.

Module register properties		
Values	Message box number:	0 ... 15
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010510

Box status

R 200010510 allows to evaluate the status of a message box.

Meaning of the individual bits	
Bit 0	**Valid** 1 = The message box is enabled
Bit 1	**NEW-DAT** 1 = The message box has received a CAN message. Reception of additional CAN messages is blocked.
Bit 2	**OVERRUN** 1 = The message box has received a new CAN message while NEW-DAT was 1.
Bit 3	**Sending error** 1 = An error has occurred when sending a CAN message from this message box.

Module register properties	
Type of access	Read access
Takes effect	if the CAN-PRIM interface is enabled.

R 200010511

Box Configuration

R 200010511 is for configuring the message box.

Configuration Values	
0	**Inbox** For configuring the box as inbox
1	**Outbox** For configuring the box as outbox for standard frames
2	**Outbox RTR** For configuring the box as outbox for RTR frames

Module register properties	
Takes effect	if the CAN-PRIM interface is enabled.

R 200010512

CAN ID

In the case of an outbox, a CAN message is sent using the CAN ID.
In the case of an inbox, only CAN messages with this CAN ID are received.

Module register properties		
Values	in the case of 11-bit CAN IDs	0 ... 0x7FF
	in the case of 29-bit CAN IDs	0 ... 0x1FFFFFFF
Takes effect	if the CAN-PRIM interface is enabled and the message box is disabled, i.e. if in R 200010510 bit 0 = 0.	

R 200010513

Number of data bytes

In the case of an outbox, a CAN message is sent with this number of data bytes.
In the case of an inbox, the number of received data bytes is entered.

Module register properties		
Values	Number of data bytes:	0 ... 8
Takes effect	if the CAN-PRIM interface is enabled.	

R 200010514 ...
R 200010521

Data bytes 0 through 7

In the case of an outbox, a CAN message is sent with these data bytes.
In the case of an inbox, the received data bytes are entered.

Module register properties		
Values	Data of data bytes:	0 ... 255
Takes effect	if the CAN-PRIM interface is enabled.	

CAN-PRIM Interface - Sample Program

Task

CAN messages with CAN IDs 0x200 are to be sent via CAN-PRIM interface. On receipt, a CAN message with CAN ID 0x277 is to be sent.

Solution

The data are sent and received via CAN-PRIM interface. To this end, a message box is configured as inbox for CAN ID 0x200. A second message box is configured as outbox with CAN ID 0x277.

Configuration

In this example, the CAN-PRIM interface of a JC-350 controller is used.

Configuring the JetSym STX Program

```
Type
    TYPE_JC_CAN_PRIM:
    Struct
        State       : Int;
        Command     : Int;
        FifoNumData : Int;
        FifoData    : Int;
    End_Struct;

    TYPE_JC_CAN_PRIM_BOX:
    Struct
        State       : Int;
        Config      : Int;
        CanId       : Int;
        DLC         : Int;
        Data        : Array[8] Of Int;
        Mask        : Int;
        Command     : Int;
    End_Struct;
End_Type;

Var
    SysBusSpecial   : Int At %VL 200002077;
    CanPrim         : TYPE_JC_CAN_PRIM At %VL 200010500;
    CanPrimBox      : Array[16] Of TYPE_JC_CAN_PRIM_BOX
                                   At %VL 200010530;
    RxData          : Array[8] Of Int;
    BoxNum          : Int;
End_Var;
```

```
Task main Autorun
    // Enabling CAN-PRIM
    // Takes effect once the controller is re-booted
    BitSet(SysBusSpecial, 2);

    // 11-bit CAN ID
    CanPrim.Command := 8;

    // Configuring box 0 to receive ID 0x200
    CanPrimBox[0].CanId := 0x200;
    // Configuring box as inbox
    CanPrimBox[0].Config := 0;
    // Enabling the box
    CanPrimBox[0].Command := 1;
    If
        BitClear(CanPrimBox[0].State, 0)
    Then
        // CAN ID is already used by CAN system bus
    End_If;

    // Configuring box 1 to send to ID 0x2FF
    CanPrimBox[1].CanId := 0x2FF;
    // Configuring box as outbox
    CanPrimBox[1].Config := 1;
    // Enabling the box
    CanPrimBox[1].Command := 1;
    If
        BitClear(CanPrimBox[1].State, 0)
    Then
        // CAN ID is already used by CAN system bus
    End_If;
End_Task;
```

JetSym STX Program - Receiving Data

```
// Waiting for new CAN messages
When
    BitSet(CanPrim.State, 1)
Continue;

// Reading box number out of FIFO buffer
BoxNum := CanPrim.FifoData;

// Checking for overrun
If
    BitSet(CanPrimBox[BoxNum].State, 2)
Then
    // Acknowledging overrun
    CanPrimBox[BoxNum].Command := 5;
End_If;

// Copying received data
RxData[0] := CanPrimBox[BoxNum].Data[0];
RxData[1] := CanPrimBox[BoxNum].Data[1];

// Resetting the NEW-DATA bit to be able to receive
// new messages in this box
CanPrimBox[BoxNum].Command := 4;
```

JetSym STX Program - Sending Data

```
// Number of data bytes = 2
CanPrimBox[1].DLC := 2;
// Entering the data to be sent
CanPrimBox[1].Data[0] := 12;
CanPrimBox[1].Data[1] := 25;

// Starting to send the CAN message
CanPrimBox[1].Command := 3;

// Checking for errors
If
    BitSet(CanPrimBox[1].State, 3)
Then
    // Acknowledging errors
    CanPrimBox[1].Command := 6;
End_If;
```

Using CAN ID Masks

Introduction

Usually the CAN-PRIM interface receives only CAN messages with a CAN ID which matches the configured CAN ID of the message box.

You can use a mask to add CAN IDs to be received by a message box. Each message box has got a CAN ID and a CAN ID mask of its own.

Operating Principle

If ...	... Then ...
Bit = 0 in R 200010542+Box*20	the bit of the CAN ID received is not evaluated.
Bit = 1 in R 200010542+Box*20	the bit of the CAN ID received must match the configured CAN ID.

JetSym STX example:

In the following example message box 5 is configured to receive CAN IDs ranging from 0x200 through 0x20F. The CAN ID mask must mask out the four least significant bits.

```
Type
    TYPE_JC_CAN_PRIM_BOX:
    Struct
        State       : Int;
        Config      : Int;
        CanId       : Int;
        DLC         : Int;
        Data        : Array[8] Of Int;
        CanIdMask   : Int;
        Command     : Int;
    End_Struct;
End_Type;

Var
    CanPrimBox      : Array[16] Of TYPE_JC_CAN_PRIM_BOX
                                At %VL 200010530;
End_Var;

Task main Autorun
    // Configuring box 5 to receive CAN IDs
    // from 0x200 through 0x20F
    CanPrimBox[5].CanId := 0x200;
    CanPrimBox[5].CanIdMask := 0x7F0;
    // Configuring box as inbox
    CanPrimBox[5].Config := 0;
    // Enabling the box
    CanPrimBox[5].Command := 1;
    // ...
```

RTR Frames Via CAN-PRIM Interface

RTR Frames

RTR (Remote Transmission Request) frames are a type of message specific to CAN. Using an RTR frame a CAN node can prompt another CAN node to send a message.

Both CAN nodes have got the same CAN ID.

Configuration for Sending and Receiving RTR Frames

Step	Action
1	Select any message box for sending RTR frames and another message box for receiving them. In this manual message box 0 is used for sending and message box 1 for receiving RTR frames.
2	Configure message box 0 as outbox for RTR frames: R 200010531 := 2;
3	Configure the CAN ID of the RTR frame: R 200010532 := CAN ID;
4	Activate message box 0: R 200010543 := 1; **Result:** Bit 0 = 1 in R 200010530
5	Configure message box 1 as inbox for replies to an RTR frame: R 200010551 := 0;
6	Configure the CAN ID of the RTR frame: R 200010552 := CAN ID;
7	Activate message box 1: R 200010563 := 1; **Result:** Bit 0 = 1 in R 200010550

Sending and Receiving RTR Frames

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Sending an RTR frame from message box 0:
R 200010543 := 3;</td></tr>
<tr><td>2</td><td>Waiting for a reply to the RTR frame in message box 1:
<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>Bit 1 = 1 NEWDAT in R 200010550</td><td>the reply to the RTR frame has been received.
Proceed with step 3</td></tr></table></td></tr>
<tr><td>3</td><td>Read the number of received bytes
Number of bytes = R 200010553;</td></tr>
<tr><td>4</td><td>Read the received bytes
Data byte 0 = R 200010554;
Data byte 1 = R 200010555;
...
Data byte 7 = R 200010561;</td></tr>
<tr><td>5</td><td>Acknowledge that the message has been received
R 200010563 := 4;</td></tr>
<tr><td>6</td><td>The message box is again ready to receive.</td></tr>
</table>

3 Fixed Software Bugs

Introduction

This chapter describes the software bugs which have been fixed in the new operating system release.

Contents

No Xcom Communication with JetSym

Effects of this Bug

Access to variables on the controller is no longer possible from within JetSym (e.g. from Setup, Monitor, Debugger). JetSym informs that only an OS update can be carried out. This problem can only be fixed by restarting the controller.

In most cases, this error occurs if the connection has been interrupted several times for a short time while communication was running (e.g. due to cable problems or a poor connection during remote maintenance).

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	< 1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Check Results in SD Card Formatting

Effects of this Bug

When the value for checking the SD card ((0x2c9b3c94) has been entered into the control register of the file system (register 202936) and the controller has been restarted, the SD card is not checked but formatted and all data on the SD card are deleted.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	< 1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Negative Default Value for UserInput() is Displayed Incorrectly

Effects of this Bug

A negative floating-point default value for `UserInput()` is incorrectly displayed on an HMI.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	< 1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Remedy / Workaround

There is no remedy/workaround for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Crash When Using NetCopyList

Effects of this Bug

When the function `NetCopyListSend()` is invoked twice in succession with an invalid handle, the controller crashes.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	< 1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Remedy / Workaround

Invoke the functions `NetCopyListSend()` and `NetCopyListDelete()` only after the function `NetCopyListConfig()` has returned a positive value.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

LED Registers Always Return 0

Effects of this Bug

Read access to LED registers 108002 through 108002 always returns the value "0".

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	< 1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	1.10.0.0
Hardware revision	not applicable
Configuration or operating mode	not applicable

Task State Registers Return Wrong Value

Effects of this Bug

Registers indicating the task state (register 210100 through 210199) return incorrect states.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340 JC-350 JC-360 JC-940MC	< 1.10.0.00 < 1.10.0.00 < 1.10.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340 JC-350 JC-360 JC-940MC	1.10.0.00 1.10.0.00 1.10.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Register Numbers

The numbers of registers indicating the state of a user task are composed of a constant value and the task ID.

```
Register number := 210100 + task ID
```

This results in register numbers ranging from 210100 through 210199.

Registers

Task state

These registers indicate the state of a task in bit-coded form.

Meaning of the individual bits		
Bit 0	**Wait for event**	
	1 =	The task is waiting for an event. For example, at a Delay() until the time has elapsed or until the access is completed in case of an access to an I/O module or via network.
Bit 1	**Active**	
	0 =	The task is not yet started, has been interrupted by the TaskBreak instruction, or terminated by the TaskExit instruction.
	1 =	The task exists and has not been interrupted.
Bit 2	**Stopped**	
	1 =	The task has been stopped at a breakpoint or by the debugger.

Bit 3	**Starting**
	1 = The program is being started
Bit 4	**Canceled**
	1 = The task has caused an unhandled exception (e.g. division by 0)
Bit 5	**Exception**
	Used by debugger
Bit 6	**Indirection**
	Used by debugger
Bit 8	**Motion Semaphore**
	Used by Motion API
Bit 9	**Break Pending**
	Used by Motion API
Bit 10	**Restart Pending**
	Used by Motion API

Module register properties	
Type of access	Read access

Crash When Accessing the Status Registers of a Task

Effects of this Bug

Read access to status registers of tasks which do not exist in the program may cause the controller JC-350 to crash. The following register areas are affected:

- 210100 ... 210199
- 210400 ... 210499
- 210500 ... 210599

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360 JC-940MC	< 1.10.0.00 < 1.10.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Make sure to read out only tasks which do exist in the executed application program.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350 JC-360 JC-940MC	1.10.0.00 1.10.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	