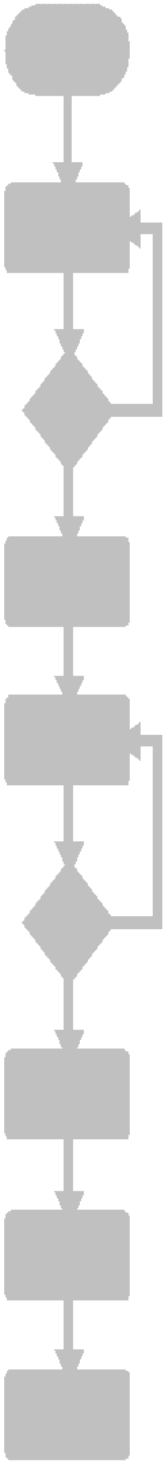


JetControl 24x

Version Update

from V. 3.18 to V. 3.20

Jetter
Automation. Made easy.

Edition 1.00 October 2005 / Printed in Germany

Jetter AG reserves the right to make alterations to its products in the interest of technical progress. These alterations need not to be documented in every single case.

This user information and the information contained herein have been compiled with due diligence. Jetter AG shall not be liable for errors contained herein or for incidental or consequential damage in connection with the furnishing, performance, or use of this material.

Table of Contents

1 Introduction

Version Updates - Survey			
Version	**Function**	**upgraded**	**corrected**
3.20	EtherNet/IP Adapter	✓	
	User programmable CAN-interface	✓	
	DNS	✓	
	System bus	✓	
	RTC		✓
V. 3.18	System bus		✓
	Networking	✓	✓
	Communication protocol	✓	
V. 3.17	Interpreter		✓

2 Expansions

2.1 EtherNet/IP

An EtherNet/IP adapter has been integrated into JetControl24x, which allows for 2 connections max. with EtherNet/IP controllers. The "Generic Device" profile is supported.

An EDS file (Electronic Data Sheet) for JetControl 24x is available on demand.

2.1.1 Licensing

To be able to use the EtherNet/IP communication protocol there has to be a valid license file when JetControl is booting. This license is an individual license, i.e. for each JetControl an individual license file is required.
If the EtherNet/IP communication protocol is ordered within the framework of a package, a valid license file is included in the delivery. This file must not be deleted. For reorders the serial number on the nameplate of the JetControl has to be indicated.

License File

The license file is named "license.dat" and is located in the subdirectory "/licenses". This file is a pure text file.
The file is split-up into certain sections. In the "ETHERNET_IP" section, there is the key for releasing the EtherNet/IP functions.

Note!

The license information for the Web functions and the Modbus/TCP protocol, if applicable, is also contained in this file. If various functions are purchased at different points in time, the contents of the various license files have to be copied together to create a common "license.dat" file.

Example:

```
[SMTPHTTP]
CONTROLLER=JetControl24x
SERIALNUMBER=20030825070353
LICENSE=bb40478bf99d5383cb8ad911879852330080f9296542bf55

[MODBUS_TCP]
CONTROLLER=JetControl24x
SERIALNUMBER=20030825070353
LICENSE=171a1dd0a3c7cfd99121834ba2c208d65adad1733f51e50f

[ETHERNET_IP]
CONTROLLER=JetControl24x
SERIALNUMBER=20030825070353
LICENSE=fc071dccbeda14c48ccd9e56c0df157447c7be6e2205f812
```

Note!

The subdirectory "/licenses" and the file "license.dat" are always available and cannot be cleared - not even by formatting the flash disk. If there is no license, the file will be empty, though.
The file has got a constant date and the permanent file length 0.

Availability of Web-Functions

The Web-functions are initialized after having switched on the controller. Bit 6 of register **2930** indicates the initialization status of the EtherNet/IP protocol, respectively of its availability.
If bit 6 has not been set, there is either no valid licence avialable, or the EtherNet/IP function has been disabled by means of register 2910 (See "Time Base" on page 8.).

Register 2930 - Bit #	Availability
0	0 = FTP server is not available 1 = FTP server is available
1	0 = HTTP server not available 1 = HTTP server available
2	0 = E-mail client not available 1 = E-mail client available
3	0 = Data file function is not available 1 = Data file function is available
4	0 = No Modbus/TCP 1 = Modbus/TCP has been licensed
5	0 = Modbus/TCP server is not available 1 = Modbus/TCP server is available
6	0 = no EtherNet/IP 1 = EtherNet/IP is available

2.1.2 Configuration

The "HostName" of the controller is needed for EtherNet/IP; it is set in the configuration memory.
The configuration memory can be accessed by means of the "/System/cfgvar.ini" file or registers 10131 through 10145 and 10200 through 10219. The configuration memory is read out once during boot-up of the controller. This means that for the changes to take effect, the JetControl must be re-started.

Configuration File

In order to gain access to the configuration file '/System/cfgvar.ini', the user having got the administrator rights must have been connected.
This file has the same structure as a Windows *.INI file:

```
[CFGVAR]
Version = 5
IP_Address    = 192.128. 10. 97
IP_SubNetMask = 255.255.255. 0
IP_DefGateway = 192.128. 10.  1
BasePort = 50000
IP_DNS        = 192.118.210.209
HostNameType  = 1
HostName      = JetControl24x
```

Important!

Please do by no means alter the version number!

Registers

Alternative access to the configuration memory is possible via registers 10131 through 10145 and 10200 through 10219.
In order to make a change via registers, first password register with password value 2002149714 (0x77566152) must be loaded. After this, registers 10132 through 10145 and 10200 through 10219 must be adjusted. Then, the changes must be saved to the configuration memory by writig any value into register 10100.

Registers	Meaning	Value used in the example
10100	Saving the configuration values	
10131	Version number	5
10132	IP address MSB	192
10133	IP address 3SB	128
10134	IP address 2SB	10
10135	IP-address LSB	97
10136	Subnet mask MSB	255
10137	Subnet mask 3SB	255
10138	Subnet mask 2SB	255
10139	Subnet mask LSB	0
10140	Default gateway MSB	192
10141	Default gateway 3SB	128
10142	Default gateway 2SB	10
10143	Default gateway LSB	1
10144	Port number of the JetIP server	50000
10145	IP address of the DNS server	0xC076D2D1 (192.118.210.209)
10159	Password	2002149714 (0x77566152)
10200	HostNameType	1
10201 through 10219	HostName (Format of the text variable)	JetControl 24x

Important!

Please do by no means change the version number in register 10131.

Time Base

The time base for processing the EtherNet/IP protocol can be set in the non-volatile register **2910**.

During boot-up, the set value is used for initializing the EtherNet/IP functions. This means that the controller has to be re-started for the value to be taken over after an alteration.

Register 2910: Time Base EtherNet/IP	
Function	**Description**
Read	Time base in milliseconds
Write	Set the new time base (will not be active before the next boot-up)
Value range	0, 2 .. 50 [ms]
Value after reset	As set last

At value '0', the Ethernet/IP adapter is not installed.

HostName / HostNameType

By means of the "HostNameType", the actually used "HostName" can be adjusted. At a "HostNameType" 1 or 2, the position identifiers of the 3 IP-address switches (S31 through S33) are attached to the "HostName" in the configuration memory. At a "HostNameType" of 3 or 4, the last three nibbles are attached to the IP-address of the controller. This way it its possible to generate unambiguous "HostName".

HostNameType	Appendices	Example
0	none	JetControl 24x
1 / 3	decimal	JetControl 24x-10-97
2 / 4	hexadecimal	JetControl 24x-a61

2.1.3 Registers

For communication with the EtherNet/IP controller, there are some 32 bit large registers available, into which the controller writes the output information, and out of which it reads the input information.
These communication registers have been subdivided in two sets each: There is one set, by which the values transmitted via EtherNet/IP are accessed directly. And there is the second set, which serves as an intermediate buffer thus safeguarding data consistency in the entire set. The values are transmitted from and to the buffer registers by means of a command issued by the user.

Registers	JetControl:	EtherNet/IP-Controller
10300 .. 10331	Outputs (direct)	Inputs

10332 .. 10363	Inputs (direct)	Outputs
10400 .. 10431	Outputs (buffer)	Inputs
10432 .. 10463	Outputs (buffer)	Outputs

Control Registers

Register 10390: Status	
Function	**Description**
Read	Status of a copying action
Write	-
Value range	32 bit (bit-coded)
Value after reset	0

Meaning of the individual status register bits:

Bit 0: **Copy EtherNet/IP inputs**

1 = Copying is in process

The contents of buffer registers 10400 through 10431 are transferred to registers 10300 through 10331 and made available to the EtherNet/IP.

0 = there is no active copying process

The data of buffer registers 10400 through 10431 can be changed consistently.

Bit 1: **Copy EtherNet/IP outputs**

1 = Copying is in process

The contents of registers 10332 through 10363 are transmitted to buffer registers 10432 through 10463.

0 = there is no active copying process

The data of buffer registers 10432 through 10463 can be accessed consistently.

Copying from or to buffer registers is asynchronous with the application program; among others, it is dependent on the EtherNet/IP cycle. This means that if a copying process has been triggered by an instruction register, resetting the respective bit in the status register, must be waited for, before another instruction can be given, or before data registers can be accessed.

Register 10391: Instruction	
Function	**Description**
Read	Latest instruction
Write	New instruction
Value range	1 .. 2
Value after reset	0

The following instructions are available for the controller:

1	**Copy EtherNet/IP inputs:** The contents of buffer registers 10400 through 10431 are transferred to registers 10300 through 10331 and made available to the EtherNet/IP.
2	**Copy EtherNet/IP outputs:** The contents of registers 10332 through 10363 are transmitted to buffer registers 10432 through 10463.

Register 10392: Errors	
Function	**Description**
Read	Latest error
Write	Clearing the register
Value range	0 .. 3
Value after reset	0

Potential errors:

0	**No error**
1	**Copying the inputs is in process already:** If, after issuing command 1, and while bit 0 still remains set in the status register, command 1 is written again, the latest command will be ignored and the error register is set to 1.
2	**Copying the inputs is in process already:** If, after issuing command 2, and while bit 1 still remains set in the status register, command 2 is written again, the latest command is ignored and the error register is set to 2.
3	**Unknown Command**

2.2 User-Programmable CAN-Interface

Note!

Basic knowledge of the Controller Area Network (CAN) is required for using functions having been provided in JetControl 24x and for understanding their description.

If the user-programmable CAN-interface (CAN-PRIM) is active, 32 virtual message boxes are available to the user. Each of these boxes can be configured as a transmit or receive box with its individual CAN-ID. For this, the ID width (11 or 29 bit) can be set for all boxes alike. Varying settings are not possible.

Note!

When the user-programmable CAN interface is applied, only 9 non-intelligent I/O modules max. can further be connected to the controller. If additional CANopen devices are applied, the number of non-intelligent modules is reduced to 7.
If 29 bit wide CAN identifiers are applied, only non-intelligent modules having got '2' as the first figure of their serial number may be connected. If older I/O modules (the serial number starts with '1') are to be applied, only IDs of 11 bit width may be applied.

Important!

Please make sure that, during boot-up of the controller, the connected devices being controlled via CAN-PRIM are not transmitting any messages!

2.2.1 Enable

The function is enabled by the special functions register of the system bus driver.

Register 2077: Special Functions of the System Bus	
Function	**Description**

Read	Present value
Write	Enabling/disabling special functions of the system bus driver
Value range	0 to 65535 (bit-coded)
Value after reset	Value entered last

Meaning of the individual controller bits:

Bit 0:	**reserved**
Bit 1:	**reserved**
Bit 2:	**CAN-PRIM** 0 = the user-programmable CAN interface is not active 1 = the user-programmable CAN interface is available

Note!

The content of this register is remanent. This means its value is stored at switching off the controller.
The settings of this register are only evaluated during re-start of the controller.

2.2.2 Interface registers

Access to the user-programmable interface can be made by means of the following registers. The content of a message box will be displayed. The respective box is called up via an index register (10502).

Register 10500: Status Register	
Function	**Description**
Read	Present state
Write	-
Value range	32 bit (bit-coded)
Value after reset	0

Meaning of the individual status register bits:

Bit 0:	**reserved**
Bit 1:	**NEW-DAT** 1 = The global NEW-DAT bit is set, if in the NEW-DAT FIFO there is at least one value (reg. 10503 > 0)
Bit 2:	**ID-width** 0 = CAN-IDs of 11 bit width are transmitted/received 1 = CAN-IDs of 29 bit width are transmitted/received

Register 10501: Command Register	
Function	**Description**
Read	Latest command
Write	New command
Value range	32 bits
Value after reset	0

The application program delays on the assignment to the command register, until the command has been processed completely.

The following commands are used by the CAN-PRIM:

1	**Enable the box:** In this case, the ID of the selected box is checked. If there is a valid ID (for CAN-PRIM, only CAN-IDs can be used, which have not been occupied by the system bus, yet), the box is enabled by setting the valid bit in the box status register (reg. 10510).
2	**Disable the box:** The box-status register of the selected box is set to '0', while the receive-FIFO is cleared of a possibly made entry.
3	**Transmit a message:** The message which is in the selected box is transmitted, if the box has been configured and enabled as a transmit box. The system will delay 5 milliseconds max., until the CAN controller has transmitted the message. If the message could not be transmitted, error bit 3 is set in the status register of the box.
4	**Clear the NEW-DAT bit:**

The following commands are used by the CAN-PRIM:	
	The NEW-DAT bit (bit 1) in the status register of the box is cleared. This way, the box is ready to receive again.
5	**Clear the reception overflow/overrun bit:** The overflow/overrun bit of the box-status register is cleared.
6	**Clear the transmit error bit:** The transmit error bit of the box-status register is cleared.
7	**Clear the NEW-DAT FIFO:** The FIFO, in which the box numbers of the received messages are displayed, is cleared.
8	**Set the standard ID width (11 bit):** The ID-width for all receive and transmit boxes is globally set to 11 bit. The receive FIFO is cleared, while the status registers of all boxes are set to '0'.
9	**Set the extended ID width (29 bit):** The ID-width for all receive and transmit boxes is globally set to 29 bit. The receive FIFO is cleared, while the status registers of all boxes are set to '0'.

Register 10502: Box Number	
Function	**Description**
Read	Number of the selected box
Write	Select new box
Value range	0 .. 31
Value after reset	0

The contents of the box selected here are displayed in registers 10510 ff. The commands of the command register also influence the box of the number set here.

Register 10503: FIFO occupancy	
Function	**Description**
Read	Number of received messages
Write	-
Value range	0 through 32
Value after reset	0

At receiving a message, this message is stored to the respective receive box; the number of this box is stored to a FIFO, in order to see easily, which box has just received a new message. The occupancy register shows how many messages have been received.

Register 10504: FIFO data	
Function	**Description**
Read	Number of the box, to which a message has been stored
Write	-
Value range	-1 through 31
Value after reset	0

At reading this register, the FIFO is cleared of the number that has just been read (while the occupancy is decremented by one). Value -1 is read, if the FIFO is empty (occupancy = 0).

The messages for the CAN-PRIM function are received in the CAN controller of the JetControl 24x by means of a reserved message box. When a message is received via the CAN bus, it is first checked for the ability to be assigned to the receive box of the system box driver.
If it cannot be assigned to the receive box, the CAN controller will check for a coincidence with the global receive-ID in register 10507. In register 10506, a mask can be defined to specify, which bits of the received ID and of the global receive-ID are to be drawn on for a comparison. If a coincidence has been found, the message is stored to one of the 32 virtual message boxes. For this, the configured ID must be exactly the same as the received ID.

Register 10506: Global receive mask	
Function	**Description**
Read	Set mask (bitcoded)
Write	Set new masks
Value range	0 through 0x7ff (11 bit ID) 0 through 0x1fffffff (29 bit ID)
Value after reset	0

At a set bit (1) in the receive mask, the corresponding bit of the received ID is compared with the global receive-ID.
A cleared bit (0) causes the corresponding bit of the IDs not to be admitted to the comparison.

Register 10507: Global receive-ID

Function	Description
Read	Set ID
Write	Define a new ID
Value range	0 through 0x7ff (11 bit ID) 0 through 0x1fffffff (29 bit ID)
Value after reset	0

By means of registers 10506 and 10507 it is possible to specify a range of IDs, which can be further transmitted to the CAN-PRIM.
By standard, all the messages, which have not been "intercepted" by the system bus driver, are further transmitted to the CAN-PRIM function.

Note!

At changing the ID-width by command 8 or 9, both registers (10506 and 10507) are reset to 0.

Via the subsequent registers, the contents of the box, the number of which has been written to register 10502 can be accessed.

Register 10510: Status register of the box	
Function	**Description**
Read	Status of the selected box
Write	-
Value range	32 bit (bit-coded)
Value after reset	0

Meaning of the individual status register bits:

Bit 0: Valid

0 = the box has been disabled; data can neither be transmitted nor received

1 = the box has been enabled; data can both be transmitted and received

Bit 1: NEW-DAT

0 = there haven't been any new messages; one receive box is ready to receive

Meaning of the individual status register bits:

1 = a new message has been received; receiving further messages has been blocked

Bit 2: Overrun

0 = no reception overrun

1 = one or more than one new messages have arrived for this box; yet, this box had not been ready to receive new messages (NEW-DAT has been set); the new message has got lost

Bit 3: Transmit error

0 = there has been no transmit error

1 = at transmitting a message at issuing command 3 an error has occurred; this is possible, for example, if there is no device connected to the system bus of the JC-24x or if the baud rates are not compatible

Register 10511: Register for box configuration	
Function	**Description**
Read	Present configuration
Write	Set new configuration; this can only be entered, if the box is not valid (reg. 10510: bit 0 = 0)
Value range	32 bit (bit-coded)
Value after reset	0

Meaning of the individual configuration register bits:

Bit 0: Transmit box

1 = transmit box

Via this box, CAN-frames can be transmitted.

0 = receive box

In this box, CAN-frames can be received.

Register 10512: CAN-ID	
Function	**Description**
Read	Actual CAN-ID

Write	Set new CAN-ID; this can only be entered, if the box is not valid (reg. 10510: bit 0 = 0)
Value range	11 or 29 bit (depending on the configuration)
Value after reset	0

In this register, the CAN-ID is set, which is "heard" by a receive box, or which is also sent when a message is transmitted via a transmit box.
When a box is activated, the ID is checked for possible overlapping with IDs that have already been allocated by the operating system. If an ID has been set that is not permitted, the respective box will not be activated.

Register 10513: Number of data bytes	
Function	**Description**
Read	Number of data bytes
Write	Number of data bytes to be transmitted (transmit box)
Value range	0 through 8
Value after reset	0

When a frame has been received, the number of data bytes that have arrived can be read out of the receive box.
If a transmit box is applied, though, the number of data bytes to be transmitted must be set here.

Registers 10514 through 10521: Data bytes 0 through 7	
Function	**Description**
Read	Data bytes 0 through 7
Write	Data bytes 0 through 7
Value range	0 through 255
Value after reset	0

There are only as many valid data bytes as there have been specified in register 10513.
At receiving a message of less than 8 bytes, the data bytes, up to data byte 7, are not overwritten after the last received byte, but keep their latest value.
At transmitting a message which is less than 8 bytes wide, the data bytes, up to data byte 7, need not to be occupied.

2.2.3 Flag

At receiving, the operating system of the controller checks for frames in cyclic mode; then it copies the received data to the corresponding receive box. In the standard setting, this check will be made once in a complete task cycle. Durch Setzen dieses Flags kann der Empfang priorisiert werden indem die Überprüfung bei jedem Anwendertaskwechsel durchgeführt wird.

Flag 2077: Priorities of CAN-PRIM Reception	
Function	**Description**
Read	Priority status of the CAN-PRIM reception
Clear	Receive check at changing from the last task to task 0
Set	Receive check at each change of user task
Value after reset	0

2.2.4 Sample program

The following segments of a JetSym ST program are to demonstrate, how the CAN-PRIM function can be initialized in the JC-24x, and how it can be used for transmitting and receiving of CAN-messages.

CAN-PRIM.stp

```
TYPE

    SYSREG_CANPRIM:
        STRUCT
            ns_State            : INT;
            ns_Command          : INT;
            ns_BoxNumber        : INT;
            ns_FifoNumData      : INT;
            ns_FifoData         : INT;
               zzDummy10        : INT;
            ns_GlobalMask       : INT;
            ns_GlobalID         : INT;
                zzDummy20       : ARRAY[2] OF INT;
            ns_BoxState         : INT;
            ns_BoxConfig           : INT;
            ns_BoxCanId         : INT;
            ns_BoxDLC           : INT;
            ans_BoxData         : ARRAY[8] OF INT;
        END_STRUCT;

END_TYPE;


VAR

    st_CanPrim : SYSREG_CANPRIM at %vl 10500;

END_VAR;
```

Fig. 1: Type and variable definition

CAN-PRIM.stp

```
    st_CanPrim.ns_Command := 8;                 // 11 bit ID

    st_CanPrim.ns_BoxNumber := 0;               // select box #0
    st_CanPrim.ns_BoxCanId  := 0x111;           // CAN-ID
    st_CanPrim.ns_BoxConfig := 0;               // receive box
    st_CanPrim.ns_Command   := 1;               // activate box
    IF
        BIT_CLEAR (st_CanPrim.ns_BoxState, 0)   // check for valid ID
    THEN
        // CAN-ID already used by system bus
    END_IF;

    st_CanPrim.ns_BoxNumber := 1;               // select box #1
    st_CanPrim.ns_BoxCanId  := 0x222;           // CAN-ID
    st_CanPrim.ns_BoxConfig := 0;               // receive box
    st_CanPrim.ns_Command   := 1;               // activate box
    IF
        BIT_CLEAR (st_CanPrim.ns_BoxState, 0)   // check for valid ID
    THEN
        // CAN-ID already used by system bus
    END_IF;

    st_CanPrim.ns_BoxNumber := 2;               // select box #0
    st_CanPrim.ns_BoxCanId  := 0x333;           // CAN-ID
    st_CanPrim.ns_BoxConfig := 1;               // transmit box
    st_CanPrim.ns_Command   := 1;               // activate box
    IF
        BIT_CLEAR (st_CanPrim.ns_BoxState, 0)   // check for valid ID
    THEN
        // CAN-ID already used by system bus
    END_IF;
```

Fig. 2: Initialization

CAN-PRIM.stp

```
// receive message
// 4 bytes --> 32 bit big endian (MSB first)
WHEN                                                    // wait for
    BIT_SET (st_CanPrim.ns_State, 1)                    // NEWDAT
CONTINUE;

st_CanPrim.ns_BoxNumber := st_CanPrim.ns_FifoData;  // select box

IF
    BIT_SET (st_CanPrim.ns_BoxState, 2)                 // check for overrun
THEN
    // process overrun error ...
    st_CanPrim.ns_Command := 5;                         // clear overrun bit
END_IF;


n_NewDat := st_CanPrim.ans_BoxData[0];

SHIFT_LEFT (n_NewDat, 8);
n_NewDat := n_NewDat + st_CanPrim.ans_BoxData[1];

SHIFT_LEFT (n_NewDat, 8);
n_NewDat := n_NewDat + st_CanPrim.ans_BoxData[2];

SHIFT_LEFT (n_NewDat, 8);
n_NewDat := n_NewDat + st_CanPrim.ans_BoxData[3];

st_CanPrim.ns_Command := 4;                             // clear NEWDAT bit
```

Fig. 3: Receiving a message

CAN-PRIM.stp

```
st_CanPrim.ns_BoxNumber     := 2;                   // select transmit box
st_CanPrim.ns_BoxDLC        := 4;                   // data length
st_CanPrim.ans_BoxData[0]   := 0x11;
st_CanPrim.ans_BoxData[1]   := 0x22;
st_CanPrim.ans_BoxData[2]   := 0x33;
st_CanPrim.ans_BoxData[3]   := 0x44;
st_CanPrim.ns_Command       := 3;                   // send message
IF
    BIT_SET (st_CanPrim.ns_BoxState, 3)             // check for transmit error
THEN
    // process transmit error ...
    st_CanPrim.ns_Command := 6;                     // clear transmit error bit
END_IF;
```

Fig. 4: Transmitting a message

2.3 DNS

As of V. 3.15, when the IP-addressed had been provided by the DNS-server, the assignment of host-names to an IP-address could also be carried out during runtime of the controller. As of this version, the DNS-resolution during runtime is also carried out, if host-names defined in the "/etc/hosts" file are used, in case there has been no connection to the IP-address specified in the file.

2.4 New Modules at the System Bus

As of this version, modules of the modular WAGO I/O-SYSTEM 750 can be connected by means of the CANopen bus head; at the initialization of the system bus, they will be put into operation automatically. Jetter brand JX2-I/O modules, JX2-Slave expansion modules and Smart I/O JX-SIO connected to the system bus can also be operated simultaneously.

2.4.1 Technical data

The following technical data have to be taken into account when connecting the WAGO-I/O-SYSTEM 750 to the system bus.

Technical Data	
Maximum number of fieldbus couplers	10
Supported fieldbus couplers	750-337 750-338
Module code	68

2.4.2 Module numbers

For operating the WAGO-I/O-SYSTEM 750 at the Jetter system bus, a module number out of 70 through 79 must be set at the DIP switch of the fieldbus coupler. Via the module number, the register and I/O numbers are set, which are needed for communication between the controller and the WAGO-I/O-SYSTEM 750.

2.4.3 Register arrays

The register arrays of the WAGO-I/O-SYSTEM 750 correspond to the register arrays of the Jetter JX-SIO modules. In this case, the 'x' needs to be replaced by the respective figures 1 through 9 of the set module number.

Register Arrays for the WAGO-I/O-SYSTEM 750		
Register Array	**Description**	**Remanent Register**
5x00 - 5x27	Register overlay for digital inputs	-
5x60 - 5x71	Analog inputs 16 Bit	-
6x00 - 6x27	Register overlay for digital outputs	-
6x60 - 6x71	Analog outputs 16 bit	-
7x00 - 7x64	Permanently assigned registers	-
7x65 - 7x89	User registers to be configured at will	-
7x90 - 7x99	Diagnostics and adminstration registers	

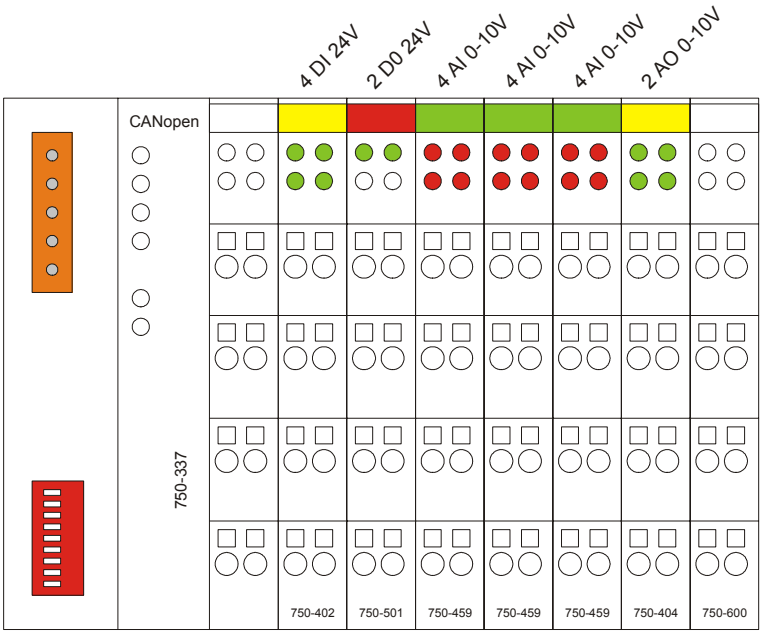
2.4.4 I/O area

The numbering of the inputs and outputs at a WAGO-I/O-SYSTEM 750 corresponds to the addresses known from JX-SIO and is carried out as follows:

- The digital inputs have got input numbers IN 7x01 bis IN 7x64. They are numbered from left to right. Terminals without digital inputs are not considered.
- The digital outputs have got output numbers OUT 7x01 bis OUT 7x64. They are numbered from left to right. Terminals without digital outputs are not considered.

Here, the 'x' results from the set module number as well.

Example 1: Numbering Inputs and Outputs



In this example, a WAGO-I/O SYSTEM-750 has been connected to a JetControl JC-24x. The WAGO-I/O-SYSTEM 750 has got module number 70.

I/O-Numbers			
Module	**Type**	**Input #**	**Output #**
750-402	4 DI 24 V	IN 7001 ... IN 7004	-
750-501	2 DO 24 V	-	OUT 7001 ... OUT 7002
750-459	4 AI 0 - 10 V	REG 5060 ... REG 5063	-
750-459	4 AI 0 - 10 V	REG 5064 ... REG 5067	-
750-459	4 AI 0 - 10 V	REG 5068 ... REG 5071	-
750-550	2 AO 0 - 10 V	-	REG 6060 ... REG 6061
750-600	End plate	-	-

For further detailed information on connection and operation of the WAGO-I/O-System 750, please refer to the WAGO user information, which is available at Jetter AG (in German; English translation to be supplied medium-term).

3 Eliminated Software Bugs

3.1 Real-Time Clock (RTC)

At a high amount of communication and/or frequent file access, the RTC component might not have been read out any more. For this reason, the registers concerning the real-time clock would possibly receive the same value over several seconds.
As of this version, reading out the RTC component is given priority before e-mail, HTTP- and FTP-server and file access.