Fidelix Classic I/O-module specifications

Table of contents

Fidelix I/O-modules 3
Power supply 3
Communication 3
Firmware version 4
Modbus protocol 4
Format of modbus query frame 5
Format of modbus response frame 5
Format of modbus exception response frame 6
Calculating CRC 6
Implemented modbus functions 6
03 (0x03) Read Holding Registers 6
06 (0x06) Write Single Register 8
16 (0x10) Write Multiple registers 9
22 (0x16) Mask Write Register 11
43 (0x2B) Read Device Identification 13
Module Specifications 17
16-Channel Digital Input Module (DI-16), version V1.10 17
Description 17
Alarm mode 18
Pulse indication mode 18
Pulse counting mode 18
Register settings 18
Register 0, input states, read 18
Register 1, alarm jumper states, read 18
Register 2, indication jumper states, read 18
Register 3, red led control register, read/write 18
Register 4, green led control register, read/write 18
Register 5, led blinking control register, read/write 19
Registers 6-13, minimum pulse width control registers, read/write 19
Registers 14-29, pulse counters for input channels, read/write 19
8-Channel Relay Output Module (DO-8) 20
Description 20
Register settings 20
Register 0, relay control register, read/write 20
Register 1, Preset values, read/write (beginning from firmware version 1.10 and 2.20) 21
8-Channel Analog Input Module (AI-8) 22
Description 22
Voltage measurement 23
Current measurement 23
Resistance measurement 23
Register settings 23
Register 0, 16 most significant bytes of analog input 1, read 23
Registers 1-7, most significant bytes of analog inputs 2-8, read 23
Register 8, 4 least significant bytes of analog inputs 1 to 4, read 23
Register 9, 4 least significant bytes of analog inputs 5 to 8, read 24
Register 10, channel enable control register, read/write 24
8-Channel Analog Output Module (AO-8) 25
Description 25

Voltage Output 25
Driving relays 26
Current output 26
Register settings 27
Register 0, analog output control register for output 1, read/write 27
Registers 1-7, analog output control register for outputs 2-8, read/write 27
Register 8, channel enable control register, read/write 27
Register 9, preset value for channel 1, read/write (beginning from firmware version 1.10 and 2.20) 27
Registers 10-16, preset values for channels 2-8, read/write (beginning from firmware version 1.10 and 2.20) 27
8-Channel Security Interface Module (SI-8) 28
Description 28
Register settings 30
Register 0 30
Registers 1-7 30
Register 8, hysteresis, read/write 30
36-Channel Combi module (COMBI-36), version v1.10 31
Description 31
Appendix A, Modbus exception codes 33
Appendix B, CRC-calculation example 34
Appendix C, Numbering of terminals 35
16-Channel Digital Input Module 35
8-Channel Relay Output Module 35
8-Channel Analog Input Module 37
8-Channel Analog Output Module 38
8-Channel Security Interface Module 39
36-Channel Combi Module 40

Fidelix I/O-modules

Power supply

Modules need 24VDC power supply. Power connector terminals are numbered as follows: D = 24VDC, C = 0V. Terminal D has different purposes depending of module type. Digital input module (DI-16) uses it as opto-coupler power supply (24VDC). Analog output module (AO-8) uses it as ground of analog outputs, this terminal should be used if all outputs uses same wire as signal return path. Combi module uses D terminal as analog ground and F terminal as opto-coupler power supply (24VDC). Other modules doesn't use terminal D (52).

Communication

Communication between outstation and modules is implemented as serial communication. Serial communication uses RS-485 standard and modbus RTU protocol. Serial communication lines will be connected to terminals A and B (older modules 55 and 56) of serial communication connector. Terminal A is DATA+ and B is DATA-. Connections should be done using twisted pair cable with 120 ohm characteristic impedance. All DATA+ (A) terminals (in modules and outstation) are connected to one wire of cable and DATA- (B) terminals to other one. FX-2020 outstation uses COM2 as RS-485 port and DATA+ (A) is connected to pin 3 and DATA- (B) to pin 1.

RS-485 bus must be connected to daisy-chain configuration (figure 1), so there must be no more than two wires connected to each terminal. At the end of bus the cable should be terminated connecting a jumper near RS-485 connector (J36 for DI-16 module, J12 for DO-8 module, J20 for AI-8 module, J20 for AO-8 module, J11 for SI-8 module, J11 for COMBI-36 module). If these instructions are not followed communication errors may result.

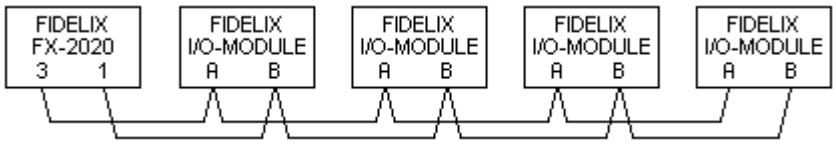

Figure 1, Right way to connect RS-485 bus

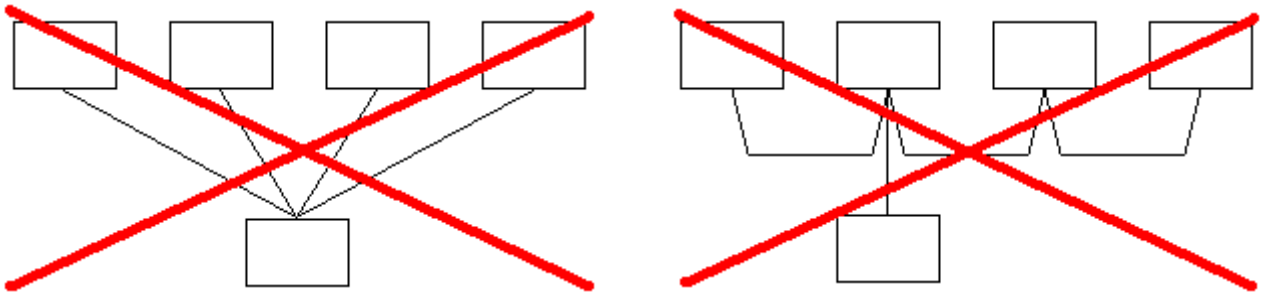

Figure 2, Wrong way to connect RS-485 bus

Communication speed is selected using dip-switch (S9 for COMBI-36 module, all others S1). Following table shows how is the communication speed selected. Communication speed has to be same in every module and outstation. Other communication parameters are: 8 data bits, no parity, 1 stop bit .

SPEED	BITRATE2 (BR2)	BITRATE1 (BR1)
9600bps	off	off
19200bps	off	on
38400bps	on	off
57600bps	on	on

Each module has its own address selected using same dip-switch (S9 for COMBI-36 module, all others S1) as in communication speed selection. Address is binary coded. Needed address can be calculated summing values of switches. For example if switches STATION1 (ST1) and STATION4 (ST4) are switched on, we get address 1 + 4 = 5. Address zero is not allowed. Each module should have different address. Following table shows couple of different address settings.

ADDRESS	ST32	ST16	ST8	ST4	ST2	ST1
1	off	off	off	off	off	on
2	off	off	off	off	on	off
3	off	off	off	off	on	on
…						
62	on	on	on	on	on	off
63	on	on	on	on	on	on

Each module has signal leds for received (RxD) and transmitted (TxD) data. RxD and TxD signs are printed on circuit board. Leds are near RS-485 connector. There is also another signal led, blinking once a second, which indicates that microcontrollers program is running (LED17 for DI-16 module, LED9 for DO-8 module, LED1 for AI-8 module, LED1 for AO-8 module, LED1 for SI-8 module, LED11 for COMBI-36 module).

Firmware version

Usually module firmaware version should be checked using serial communication and read device identification function. Version number can also be checked short-circuiting pins 2 and 3 of module programming connector (J34 for DI-16 module, J10 for DO-8 module, J18 for AI-8 module, J18 for AO-8 module, J10 for SI-8 module, J12 for COMBI-36 module) and resetting module by turning module power off and after a while back on. Programming connector is 6-pin long and it is located near RS-485 connector (Figure 3). Module starts to show firmware version number by blinking led used to indicate that program is running (LED17 for DI-16 module, LED9 for DO-8 module, LED1 for AI-8 module, LED1 for AO-8 module, LED1 for SI-8 module, LED11 for COMBI36-module). Led blinks first as many time as firmware major version, then it makes short pause and then led blinks as many time as minor version is. After that module waits three second and cycle starts again. For example if led blinks first once and after little pause three times, firmware version is 1.3 and after three second this cycle starts again. Removing short-circuit allows program to proceed normally. It is highly recommend using serial communication to check firmware version if possible.

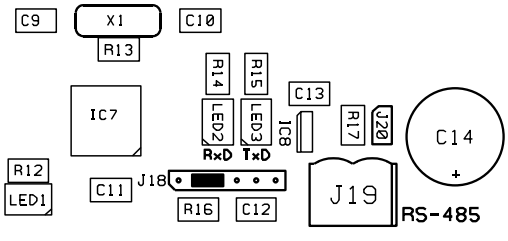

Figure 3, Short-circuiting pins 2 and 3 to enter firmware version check (AO-8 module)

Modbus protocol

Communication on a modbus network is initiated by outstation (master) with a query message to a module (slave). Each module are monitoring bus constantly and if query address matches the module address query is processed and module will perform needed action. After query is processed and needed actions are performed module will produce a

response message back to the outstation. It is important to notice that RS-485 bus is half-dublex, which means that only one device can send data at a time. It means, that bus has to be free when module starts its response, otherwise response cannot be send. More information about modbus protocol can be found from internet, http://www.modbus.org . Following section will introduce some modbus features more specific. Also modbus commands supported by Fidelix modules are explained.

Format of modbus query frame

Modbus query frame always consists of module address, function code, some data bytes and error check field as show in figure 4.

Module Address	Function Code	Data Bytes	Error Check (Low Byte)	Error Check (High Byte)

Figure 4, Modbus query frame

Module Address: Address of module that is queried. Field length is one byte. Fidelix modules support addresses from 1 to 63.

Function Code: Code of function that is performed in module. Field length is one byte. Fidelix modules support following function codes:

0x03, read multiple registers

0x06, write single register

0x10, write multiple registers

0x16, mask write register

0x2B, read device identification

Function codes are explained later in the document.

Data Bytes: Data to module. Field length is function dependent. See function descriptions for each function for more specific information.

Error Check: Error check field. Field length is two bytes. Error check is 16 bit CRC (Cyclic Redundancy Check). Least significant byte of CRC is send first (different order than used with data bytes). CRC calculation is explained later in this document.

Format of modbus response frame

After module has got a query it starts immediately to process it and generate response message back to outstation. Module sends the response message immediately after it is generated or after three milliseconds, depending which is longer time. Anyhow this should never take longer than 20 milliseconds. Format of modbus response frame is similar to query frame.

Module Address	Function Code	Data Bytes	Error Check (Low Byte)	Error Check (High Byte)

Figure 5, Modbus responce frame in normal situation

Module Address: Address of responding module. Field length is one byte.

Function Code: Function code of performed function (same as queried). Field length is one byte.

Data Bytes: Data back to outstation. Field length is function dependent. See function descriptions for each function for more specific information.

Error Check: Error check field. Field length is two bytes. Error check is 16 bit CRC (Cyclic Redundancy Check). Least significant byte of CRC is send first (different order than used with data bytes). CRC calculation is explained later in this document.

Format of modbus exception response frame

Modbus exception response is send if any error in the contents of query is detected (excluding parity error and error check mismatch). Exception codes that Fidelix modules may send are explained in appendix A.

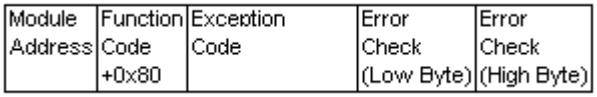

Figure 6, Modbus exception response frame

Module Address: Address of responding module. Field length is one byte.

Function Code: Function code of queried function added with hex value 0x80 (most significant bit is set to logical 1). Field length is one byte.

Exception Code: Exception code explaining the type of error. Exception codes are explained in appendix A. Field length is one byte.

Error Check: Error check field. Field length is two bytes. Error check is 16 bit CRC (Cyclic Redundancy Check). Least significant byte of CRC is send first (different order than used with data bytes). CRC calculation is explained later in this document.

Calculating CRC

Cyclic redundancy check is very reliable method to detect errors during communication. Transmitting device always calculates CRC of message and adds it to the end of message frame. Receiving device calculates CRC again (CRC bytes of message are not included to calculation) and compares it to received CRC value. These two CRCs should match. If mismatch is detected that message is discarded. CRC calculation example in C is shown in appendix B.

Implemented modbus functions

03 (0x03) Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a module. The request specifies the starting register address and the number of registers to read. In the module registers are addressed starting at zero.

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Request

Module Address	1 Byte	0x01 to 0x3F

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to last register
Quantity of Registers	2 Bytes	1 to number of registers
Error Check	2 Bytes	16bit CRC

Response

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	
Error Check	2 Bytes	16bit CRC

***N** = Quantity of Registers

Exception

Module Address	1 Byte	0x01 to 0x3F
Exception Function code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03
Error Check	2 Bytes	16bit CRC

Here is an example of a request to read registers 8 – 10 from module 2:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Module Address	**02**	Module Address	**02**
Function	**03**	Function	**03**
Starting Address Hi	**00**	Byte Count	**06**
Starting Address Lo	**07**	Register value Hi (reg 08)	**02**
No. of Registers Hi	**00**	Register value Lo (reg 08)	**2B**
No. of Registers Lo	**03**	Register value Hi (reg 09)	**00**
Error Check Lo	**B4**	Register value Lo (reg 09)	**00**
Error Check Hi	**39**	Register value Hi (reg 10)	**00**
		Register value Lo (reg 10)	**64**
		Error Check Lo	**11**
		Error Check Hi	**8A**

The contents of register 08 are shown as the two byte values of 02 2B hex, or 555 decimal. The contents of registers 9–10 are 00 00 and 00 64 hex, or 0 and 100 decimal, respectively.

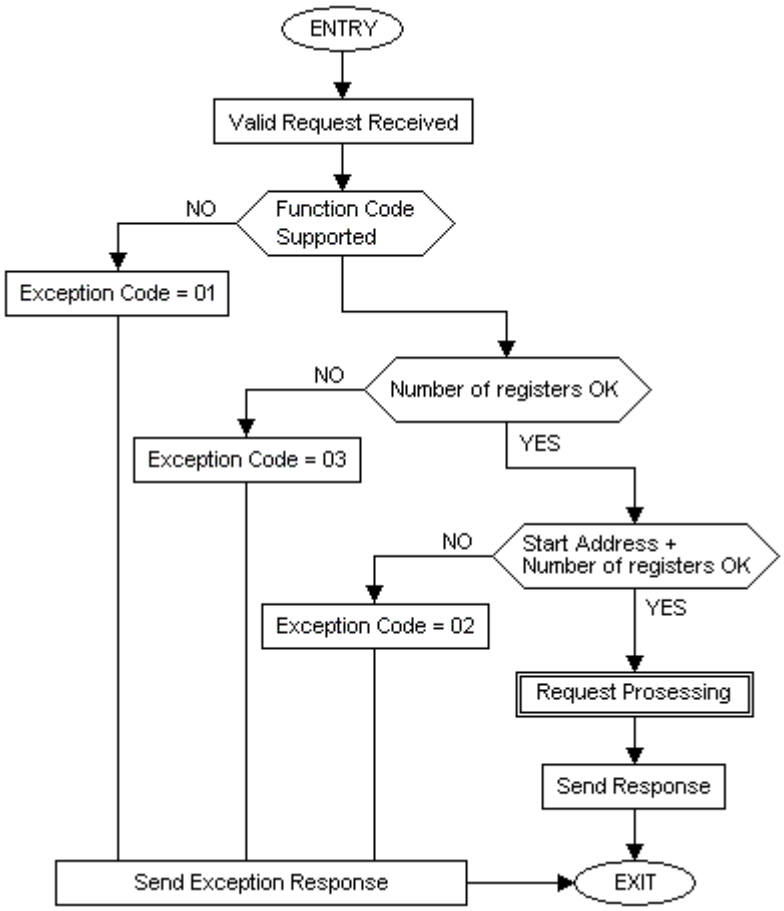

Figure 7, Read Holding Registers state diagram

06 (0x06) Write Single Register

This function code is used to write a single register in a module. The request specifies the address of the register to be written. Registers are addressed starting at zero. The normal response is an echo of the request, returned after the register contents have been written.

Request

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to last register
Register Value	2 Bytes	0x0000 to 0xFFFF
Error Check	2 Bytes	16bit CRC

Response

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to last register
Register Value	2 Bytes	0x0000 to 0xFFFF
Error Check	2 Bytes	16bit CRC

Exception

Module Address	1 Byte	0x01 to 0x3F
Exception Function code	1 Byte	**0x86**
Exception code	1 Byte	01 or 02
Error Check	2 Bytes	16bit CRC

Here is an example of a request to write register 2 in module 2 to 00 03 hex:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Module Address	**02**	Module Address	**02**
Function	**06**	Function	**06**
Register Address Hi	**00**	Register Address Hi	**00**
Register Address Lo	**01**	Register Address Lo	**01**
Register Value Hi	**00**	Register Value Hi	**00**
Register Value Lo	**03**	Register Value Lo	**03**
Error Check Lo	**98**	Error Check Lo	**98**
Error Check Hi	**38**	Error Check Hi	**38**

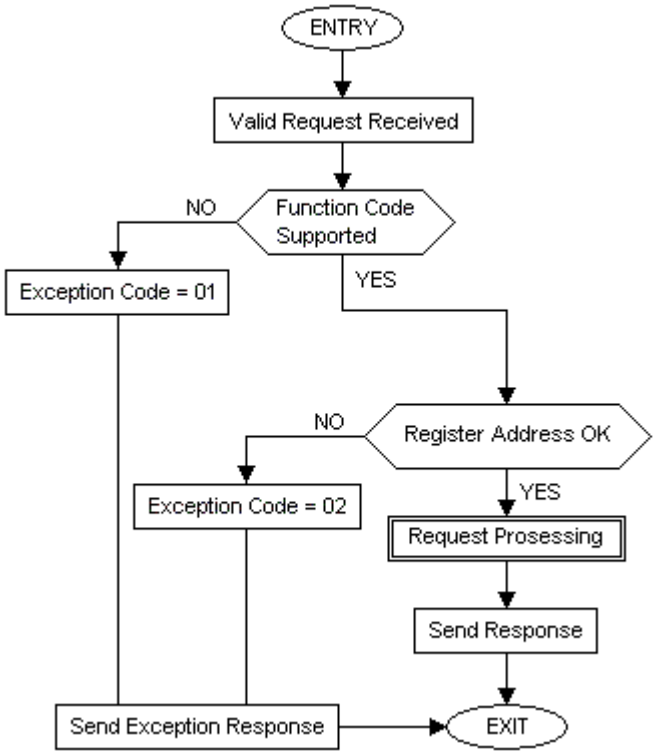

Figure 8, Write Single Register state diagram

16 (0x10) Write Multiple registers

This function code is used to write a block of contiguous registers in a module. The requested written values are specified in the request data field. Data is packed as two bytes per register. Registers are addressed starting at zero. The normal response returns the function code, starting address, and quantity of registers written.

Request

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to last register
Quantity of Registers	2 Bytes	1 to number of registers
Byte Count	1 Byte	2 x **N***
Registers Value	**N*** x 2 Bytes	value
Error Check	2 Bytes	16bit CRC

***N** = Quantity of Registers

Response

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to last register
Quantity of Registers	2 Bytes	1 to number of registers
Error Check	2 Bytes	16bit CRC

Exception

Module Address	1 Byte	0x01 to 0x3F
Exception Function code	1 Byte	**0x90**
Exception code	1 Byte	01 or 02 or 03
Error Check	2 Bytes	16bit CRC

Here is an example of a request to write two registers in module 2 starting at 2 to 00 0A and 01 02 hex:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Module Address	**02**	Module Address	**02**
Function	**10**	Function	**10**
Starting Address Hi	**00**	Starting Address Hi	**00**
Starting Address Lo	**01**	Starting Address Lo	**01**
Quantity of Registers Hi	**00**	Quantity of Registers Hi	**00**
Quantity of Registers Lo	**02**	Quantity of Registers Lo	**02**
Byte Count	**04**	Error Check Lo	**10**
Registers Value Hi	**00**	Error Check Hi	**3B**
Registers Value Lo	**0A**		
Registers Value Hi	**01**		
Registers Value Lo	**02**		
Error Check Lo	**9D**		
Error Check Hi	**74**		

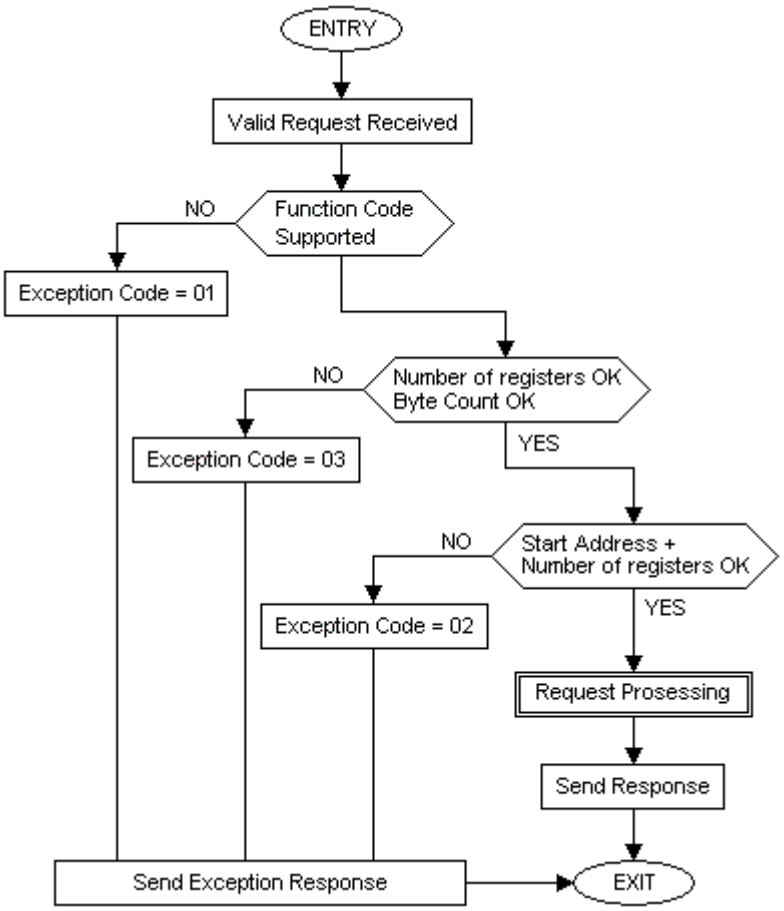

Figure 9, Write Multiple Registers state diagram

22 (0x16) Mask Write Register

This function code is used to modify the contents of a specified register using a combination of an AND mask, an OR mask, and the register's current contents. The function can be used to set or clear individual bits in the register. The request specifies the register to be written, the data to be used as the AND mask, and the data to be used as the OR mask. Registers are addressed starting at zero.

The function's algorithm is:

$$\text{Result} = (\text{Current Content AND And\_mask}) \text{ OR } (\text{Or\_mask AND } \overline{\text{And\_mask}})$$

For example:

	Hex	**Binary**
Current Content =	12	0001 0010
And_Mask =	F2	1111 0010
Or_Mask =	25	0010 0101
_____		
And_Mask =	0D	0000 1101
Result =	17	0001 0111

☞ **Note**:

- That if the Or_Mask value is zero, the result is simply the logical ANDing of the current contents and And_Mask. If the And_Mask value is zero, the result is equal to the Or_Mask value.

The normal response is an echo of the request. The response is returned after the register has been written.

Request

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to last register
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF
Error Check	2 Bytes	16bit CRC

Response

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to last register
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF
Error Check	2 Bytes	16bit CRC

Exception

Module Address	1 Byte	0x01 to 0x3F
Exception Function code	1 Byte	**0x96**
Exception code	1 Byte	01 or 02
Error Check	2 Bytes	16bit CRC

Here is an example of a Mask Write to register 5 in module 2, using the above mask values.

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Module Address	**02**	Module Address	**02**
Function	**16**	Function	**16**
Reference address Hi	**00**	Reference address Hi	**00**
Reference address Lo	**04**	Reference address Lo	**04**
And_Mask Hi	**00**	And_Mask Hi	**00**
And_Mask Lo	**F2**	And_Mask Lo	**F2**
Or_Mask Hi	**00**	Or_Mask Hi	**00**
Or_Mask Lo	**25**	Or_Mask Lo	**25**
Error Check Lo	**27**	Error Check Lo	**27**
Error ceck Hi	**FB**	Error Check Hi	**FB**

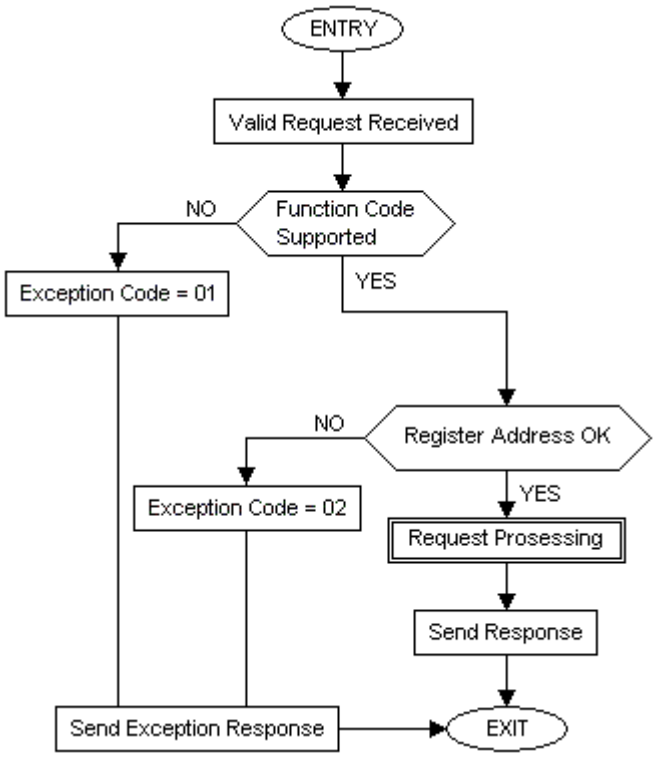

Figure 10, Mask Write Holding Register state diagram

43 (0x2B) Read Device Identification

This function code allows reading the identification and additional information relative to the physical and functional description of a module. The Read Device Identification interface is modelled as an address space composed of a set of addressable data elements. The data elements are called objects and an object Id identifies them.

The interface consists of 3 categories of objects (Fidelix modules have Basic and Regular categories):

- Basic Device Identification. All objects of this category are mandatory: Vendor Name, Product Code, and Revision Number.
- Regular Device Identification. In addition to Basic data objects, the device provides additional and optional identification and description data objects. All of the objects of this category are defined in the standard but their implementation is optional. Fidelix modules have VendorURL and ProductName objects.
- Extended Device Identification. In addition to regular data objects, the device provides additional and optional identification and description private data. All of these data are device dependent. Fidelix modules don't have extended device identifications.

Object Id	Object Name / Description	Type	Category
0x00	VendorName	ASCII String	**Basic**
0x01	ProductCode	ASCII String	
0x02	MajorMinorRevision	ASCII String	
0x03	VendorUrl	ASCII String	**Regular**
0x04	ProductName	ASCII String	
0x05	ModelName	ASCII String	

0x06	UserApplicationName	ASCII String	
0x07 … 0x7F	*Reserved*		
0x80 … 0xFF	*Private objects may be **optionally** defined* *The range [0x80 – 0xFF] is Product dependant.*	device dependant	**Extended**

Request

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x2B**
MEI Type*	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Object Id	1 Byte	0x00 to 0xFF
Error Check	2 Bytes	16bit CRC

* MEI = Modbus Encapsulated Interface

Response

Module Address	1 Byte	0x01 to 0x3F
Function code	1 Byte	**0x2B**
MEI Type	1 byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Conformity level	1 Byte	
More Folows	1 Byte	00
Next Object Id	1 Byte	00
Number of objects	1 Byte	
List Of		
Object ID	1 Byte	
Object length	1 Byte	
Object Value	1 Byte	
Error Check	2 Bytes	16bit CRC

Exception

Module Address	1 Byte	0x01 to 0x3F
Exception Function code	1 Byte	**0xAB :** **Fx 0x2B + 0x80**
MEI Type	1 Byte	0x0E
Exception code	1 Byte	01, 02, 03
Error Check	2 Bytes	16bit CRC

A Modbus Encapsulated Interface assigned number 14 (0xE0 as hex) identifies the Read identification request.

The paremeter "Read Device ID code" allows to define four access types :

01 : request to get the basic device identification (stream access)
02 : request to get the regular device identification (stream access)
03 : request to get the extended device identification (stream access)
04 : request to get one specific identification object (individual access)

An exception code 03 is sent back in the response if the Read device ID code is illegal.

In the case where the identification data does not fit into a single response, several request/response transactions are required. The Object Id byte gives the identification of the first object to obtain. For the first transaction, the client must set the Object Id to point first object that is going to be read. For the following transactions, the client must set the Object

Id to the value returned by the server in its previous response. Fidelix modules don't use multipart responses because all messages will fit into a single response.

In case of an individual access: ReadDevId code 04**,** the Object Id in the request gives the identification of the object to obtain.

If the Object Id doesn't match to any known object, the server returns an exception response with exception code = 02 (Illegal data address).

Response parameter description :

Function code:	Function code 43 (decimal) 0x2B (hex)
MEI Type:	14 (0x0E) MEI Type assigned number for Device Identification Interface
ReadDevId code:	Same as request ReadDevId code : 01, 02, 03 or 04
Conformity Level:	Identification conformity level of the device and type of supported access 01 : basic identification (stream access only) 02 : regular identification (stream access only) 03 : extended identification (stream access only) 81 : basic identification (stream access and individual access) 82 : regular identification (stream access and individual access) 83 : extended identification (stream access and individual access)
More Follows:	***In case of ReadDevId codes 01, 02 or 03 (stream access),*** If the identification data doesn't fit into a single response, several request/response transactions may be required. Fidelix modules don't use multipart responses and More Follows field is always zero. 00 : no more Object are available FF : other identification Object are available and further Modbus transactions are required ***In case of ReadDevId code 04 (individual access),*** this field must be set to 00.
Next Object Id:	If "MoreFollows = FF", identification of the next Object to be asked for. if "MoreFollows = 00", must be set to 00 (useless)
Number Of Objects:	Number of identification Object returned in the response (for an individual access, Number Of Objects = 1)
Object0.Id:	Identification of the first Object returned in the PDU (stream access) or the requested Object (individual access)
Object0.Length:	Length of the first Object in byte
Object0.Value:	Value of the first Object (Object0.Length bytes)
…	
ObjectN.Id:	Identification of the last Object (within the response)
ObjectN.Length:	Length of the last Object in byte
ObjectN.Value:	Value of the last Object (ObjectN.Length bytes)

Example of a Read Device Identification (from module 2) request for "Basic device identification".

Request	Response

Field Name	*Value*	*Field Name*	*Value*
Module Address	**02**	Module Address	**02**
Function	**2B**	Function	**2B**
MEI Type	**0E**	MEI Type	**0E**
Read Dev Id code	**01**	Read Dev Id Code	**01**
Object Id	**00**	Conformity Level	**01**
Error Check Lo	**34**	More Follows	**00**
Error Check Hi	**77**	NextObjectId	**00**
		Number Of Objects	**03**
		Object Id	**00**
		Object Length	**07**
		Object Value	**" Fidelix"**
		Object Id	**01**
		Object Length	**05**
		Object Value	**" DI-16 "**
		Object Id	**02**
		Object Length	**05**
		Object Value	**"V1.00"**
		Error Check Lo	**4F**
		Error Check Hi	**71**

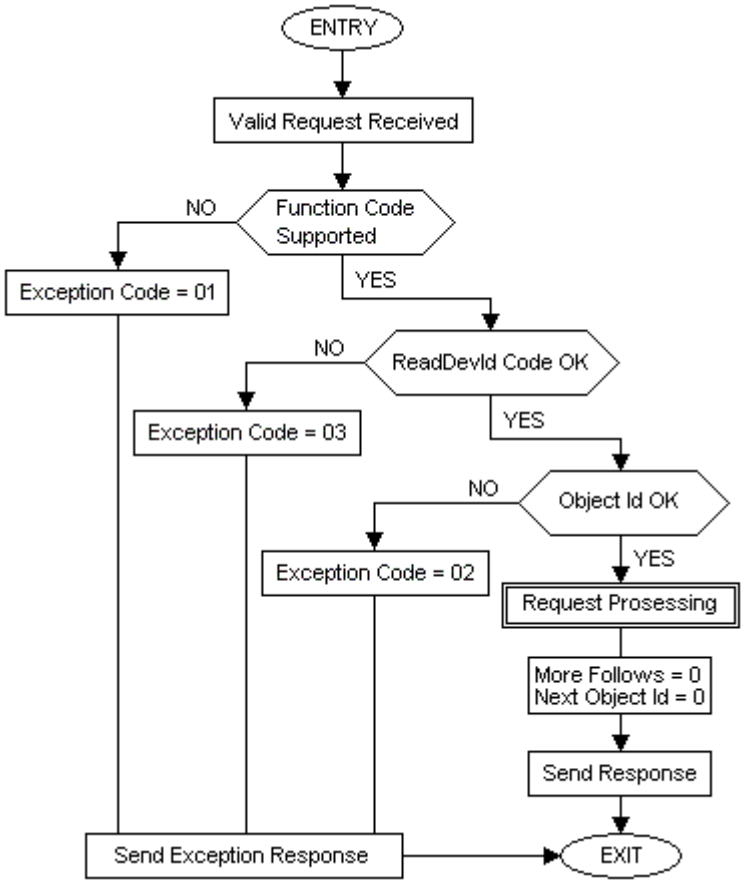

Figure 11, Read Device Identification state diagram

Module Specifications

16-Channel Digital Input Module (DI-16), version V1.10

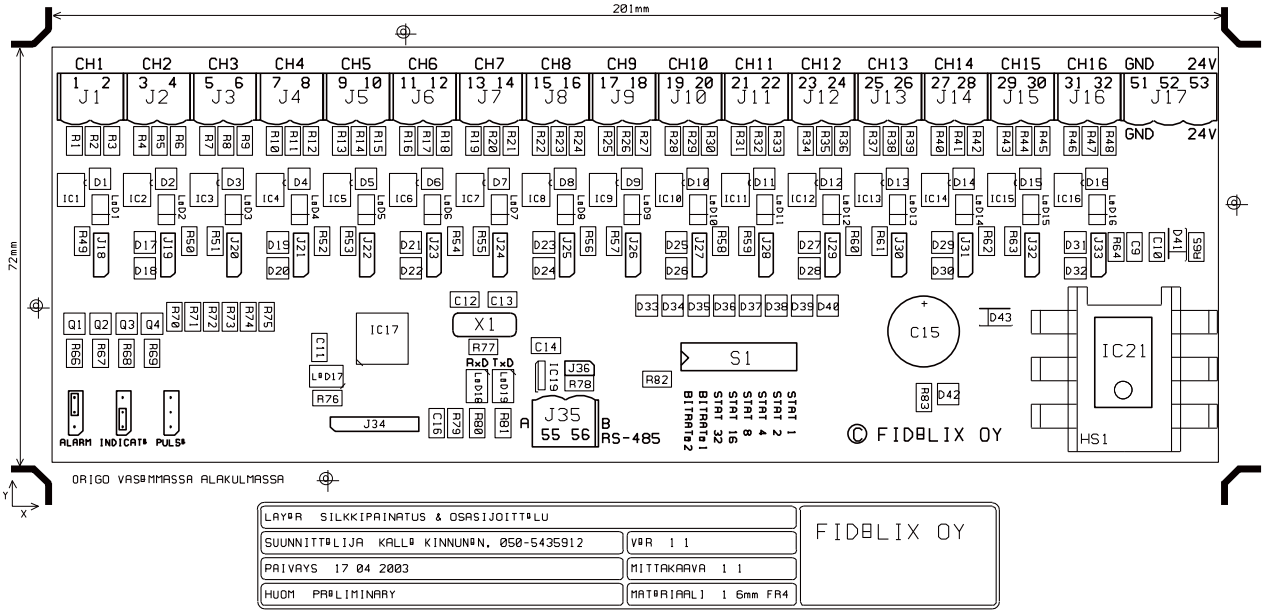

Figure 12, 16-Channel Digital Input Module, component placement

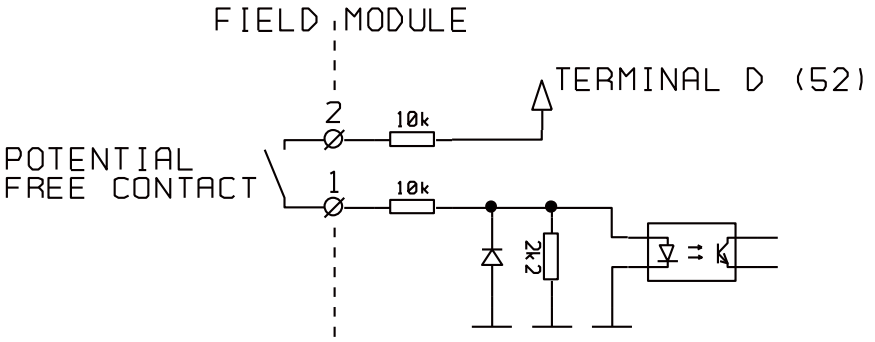

Figure 13, Digital input circuit (Channel 1)

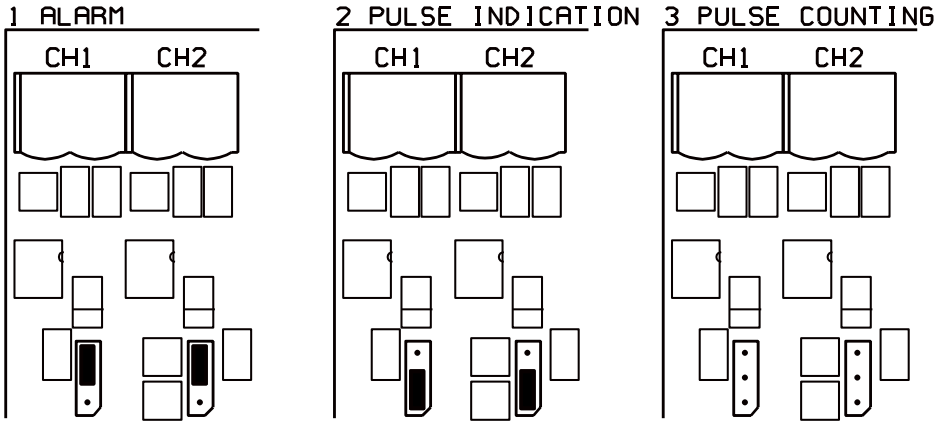

Figure 14, Operation mode selection jumpers

Description

DI-16 module has 16 optically isolated inputs. Only potential free contacts should be connected to inputs (as shown in figure 13). Inputs have three different operation modes, which can be selected using jumpers J18 – J33. Modes are alarm mode, pulse indication

mode and pulse counting mode. Opto-couplers need 24VDC power supply connected to terminal number 52 of power connector.

Alarm mode

This mode is selected if jumper is connected to upper position (figure 14, situation 1). In this mode pulses longer than minimum pulsewidth are recognized and all leds are controlled by outstation.

Pulse indication mode

This mode is selected if jumper is connected to lower position (figure 14, situation 2). All pulses longer than minimum pulsewidth are indicated and corresponding green led is set as long as pulse is present.

Pulse counting mode

This mode is selected if jumper is not connected (figure 14, situation 3). This mode is functionally same as pulse indication, but this setup is used to indicate that point is used as pulse counter. All pulses longer than minimum pulsewidth are counted.

Register settings

Register 0, input states, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	IN16	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1

Register 0 shows status of inputs. Bit 0 means status of input channel 1 and bit 15 channel 16. If input is connected longer than time specified in minimum pulse width regiser (register 6) corresponding bit is cleared, otherwise bit stays set. This register is read only register.

Register 1, alarm jumper states, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG1	JMP16	JMP15	JMP14	JMP13	JMP12	JMP11	JMP10	JMP9	JMP8	JMP7	JMP6	JMP5	JMP4	JMP3	JMP2	JMP1

Register 1 shows status of alarm jumpers. Bit 0 means jumper of input channel 1 and bit 15 channel 16. If jumper is set in indication position corresponding bit is cleared, otherwise bits stay set. This register is read only register.

Register 2, indication jumper states, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG2	JMP16	JMP15	JMP14	JMP13	JMP12	JMP11	JMP10	JMP9	JMP8	JMP7	JMP6	JMP5	JMP4	JMP3	JMP2	JMP1

Register 2 shows status of indication jumpers. Bit 0 means jumper of input channel 1 and bit 15 channel 16. If jumper is set in alarm position corresponding bit is cleared, otherwise bits stay set. This register is read only register.

Register 3, red led control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG3	RED16	RED15	RED14	RED13	RED12	RED11	RED10	RED9	RED8	RED7	RED6	RED5	RED4	RED3	RED2	RED1

Register 4 controls red leds. Bit 0 controls led of input channel 1 ans bit 15 controls led 16. If bit is set corresponding led is set and clear bit means that led is off. This register is read and write register.

Register 4, green led control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG4	GRN16	GRN15	GRN14	GRN13	GRN12	GRN11	GRN10	GRN9	GRN8	GRN7	GRN6	GRN5	GRN4	GRN3	GRN2	GRN1

Register 4 controls green leds. Bit 0 controls led of input channel 1 ans bit 15 controls led 16. If bit is set corresponding led is set and clear bit means that led is off. This register is read and write register.

Register 5, led blinking control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG5	BLK16	BLK15	BLK14	BLK13	BLK12	BLK11	BLK10	BLK9	BLK8	BLK7	BLK6	BLK5	BLK4	BLK3	BLK2	BLK1

Register 5 controls blinking of leds. Bit 0 controls led of input channel 1 ans bit 15 controls led 16. If bit is set corresponding led is blinking (if led is set) and clearing bit means that led is not blinking. Blinking frequency is fixed. This register is read and write register.

Registers 6-13, minimum pulse width control registers, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG6	MSB2							LSB2	MSB1							LSB1

Registers 6 to 13 controls minimum allowed pulse width. All pulses shorter than set in these registers are discarded. Pulse width is counted in both rising and falling edges of pulse. It is very important to notice that each register sets pulse widths for two input channels. Lower eight bits are for odd channels (channels 1, 3, 5, 7, 9, 11, 13, 15) and higher eight bits for even channels (channels 2, 4, 6, 8, 10, 12, 14, 16). Register 6 is for inputs 1 and 2, register 7 for inputs 3 and 4, register 8 for inputs 5 and 6, etc. Pulse width value is multiplied by 5ms. For example if you set hexadesimal value 0x0A01 to register 6, minimum pulse width for channel one is set to 5ms (0x01 * 5ms) and for channel two it set to 50ms (0x0A * 5ms). Minimum pulse width can be set as big as 1275ms (0xFF * 5ms). This register is read and write register.

Registers 14-29, pulse counters for input channels, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG14	MSB															LSB

Registers 14 to 29 counts every pulse longer than set in corresponding pulse width register (registers 6 to 13). Register 14 is for input 1, register 15 for input 2, register 16 for input 3, etc. Counters are 16bit long counters and they will reset after 65536 pulses. These registers are read and write registers.

8-Channel Relay Output Module (DO-8)

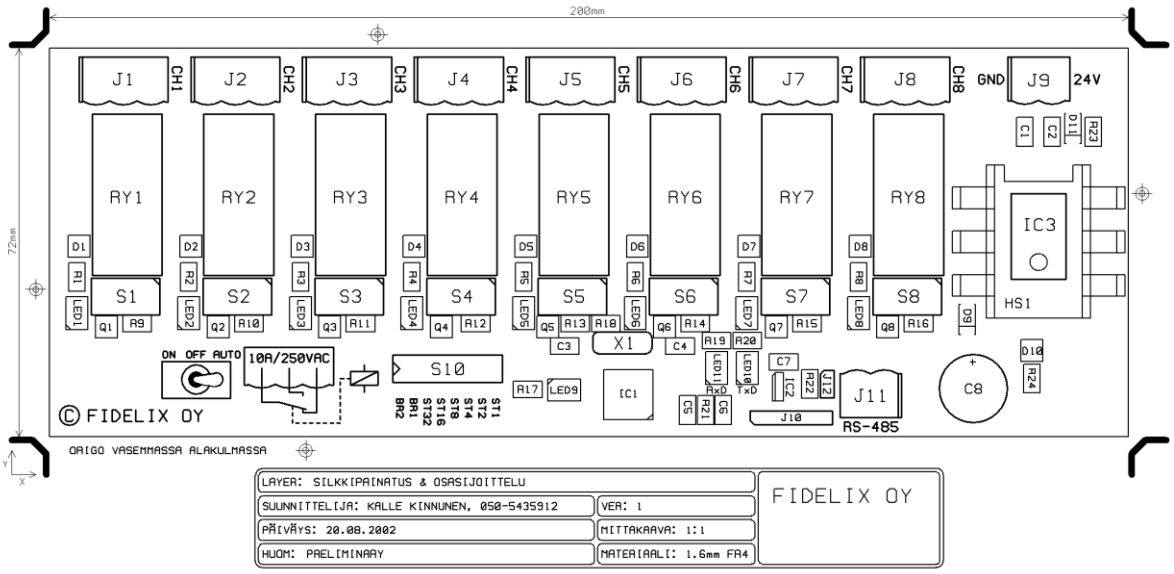

Figure 15, 8-Channel Relay Output Module, component placement

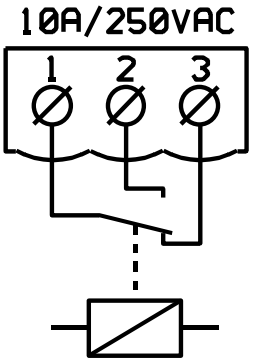

Figure 16, Output circuit (Channel 1)

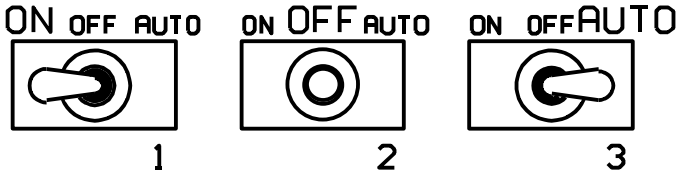

Figure 17, Different switch positions

Description

DO-8 module has 8 change-over relay outputs. All relays can be controlled either manually or by controller (see operation modes in figure 17). Each relay has also own indication led to show relay status. Relay connection diagram is shown in figure 15. Maximum load is 10A@250VAC or 10A@30VDC if resistive load and 7.5A@250VAC or 5A@30VDC if inductive (p.f. = 0.4, L/R = 7ms) load.

Register settings

Register 0, relay control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	X	X	X	X	X	X	X	X	RLY8	RLY7	RLY6	RLY5	RLY4	RLY3	RLY2	RLY1

X = NOT USED

Register 0 controls output relays. If bit is set corresponding relay operates (if mode switch is in auto position). Clearing bit causes relay to release. Only 8 least significant bits are used. This register is read and write register.

Register 1, Preset values, read/write (beginning from firmware version 1.10 and 2.20)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG1	MOD8	MOD7	MOD6	MOD5	MOD4	MOD3	MOD2	MOD1	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1

Register 1 has preset values for outputs. Preset values are used when power is turned on and substation has not been sending new data. Higher byte of register determines for each channel independently if preset values is in use. If bit is set, preset value is taken from lower byte of register. Otherwise current state of output is preserved. Same Preset values are also used if communication failure occurs. Communication failure means that mudule hasn’t got any modbus messages for 30 seconds. This register is read and write register.

8-Channel Analog Input Module (AI-8)

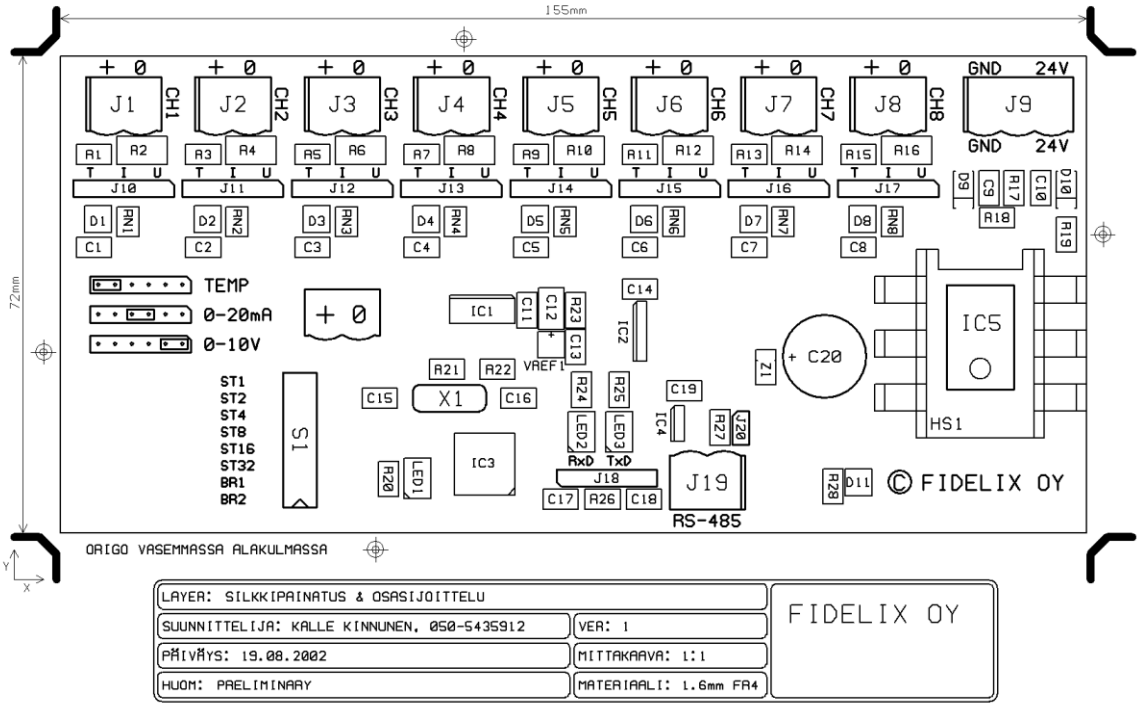

Figure 18, 8-Channel Analog Input Module, component placement

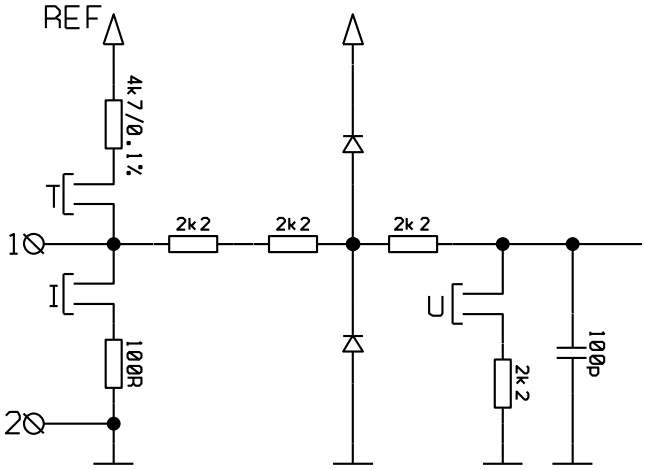

Figure 19, Analog input circuit (Channel 1)

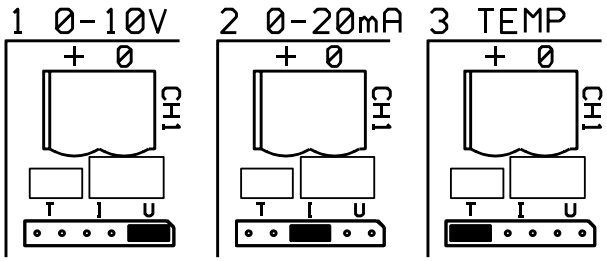

Figure 20, Sensor type selection jumper

Description

AI-8 module has 8 multiplexed analog input channels. Measurement resolution is 20 bits and used AD converter is delta sigma converter. Used sensor type can be selected for each channel using jumpers J10 – J17, selection guide is printed on PCB and it is also shown in

figure 20. Connector polarity is printed on PCB and is seen also in figure 18. Possible sensors are voltage sensor (range 0-10V), current sensor (range 0-25mA) and resistive sensor.

Voltage measurement

Voltage measurement is selected if jumper is set to right position (Figure 20, situation 1). Measurement range is from 0 Volt to 10 Volt. Input impedance is about 8.8 kilo-ohm.

Current measurement

Current measurement is selected if jumper is set to middle position (Figure 20, situation 2). Measurement range is from 0mA to 25mA. Current will flow thru 100 ohm resistor and voltage over that resistor is measured. Current can be calculated usin following equation,

$$I = \frac{U}{100\Omega}.$$

Resistance measurement

Resistance measurement is selected if jumper is set to left position (Figure 20, situation 3). A reference voltage is feed thru 4.7 kilo-ohm precision resistor and voltage over measured resistor is measured. Corresponding resistor can be calculated using following equation,

$$R = \frac{4700\Omega \cdot U}{2.5V - U}.$$

If resistive sensor is used to measure for example temperature, resistance has to be converted to temperature value using some conversion table. Conversion table is usually provided with sensor.

Register settings

Register 0, 16 most significant bytes of analog input 1, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	MSB															BIT4

Register 0 cointains 16 most significant bytes of analog input channel 1 measurement data. If 16bit resulution is good enough measured voltage value can be calculated using following equation,

$$U = \frac{register\_value}{65535} \cdot 2.5V$$

If 20 bit resolution is needed, see register 8. This register is read only register.

Registers 1-7, most significant bytes of analog inputs 2-8, read

Same as register 0 but for analog channels 2 to 8. These register are read only registers ister.

Register 8, 4 least significant bytes of analog inputs 1 to 4, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG8	AD4_3	AD4_2	AD4_1	AD4_0	AD3_3	AD3_2	AD3_1	AD3_0	AD2_3	AD2_2	AD2_1	AD2_0	AD1_3	AD1_2	AD1_1	AD1_0

Register 8 contains 4 least significant bits of analog measurements from input channels 1 to 4. If these 4 bits are combined with corresponding 16 bit value (16 bit values in registers 0-7), 20 bit measurement data is achieved. If 20bit resolution is used measured voltage value can be calculated using following equation,

$$U = \frac{measured\_value}{1048575} \cdot 2.5V$$

This register is read only register.

Register 9, 4 least significant bytes of analog inputs 5 to 8, read

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG9	AD8_4	AD8_2	AD8_2	AD8_0	AD7_3	AD7_2	AD7_1	AD7_0	AD2_3	AD6_2	AD6_1	AD6_0	AD5_3	AD5_2	AD5_1	AD5_0

Same as register 8 but for analog channels 5 to 8. This register is read only register.

Register 10, channel enable control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG10	X	X	X	X	X	X	X	X	ENA8	ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1

X = NOT USED

Register 10 controls enabling of analog input channels. If bit is is set, corresponding input channel is enabled. One measurement takes about 170ms, if all 8 channels are enabled each measurement is repeated every 1,36 seconds. If faster response time is needed can amount of used channels be reduced so that needed response time is achieved. Only 8 least significant bits are used. This register is read and write register.

8-Channel Analog Output Module (AO-8)

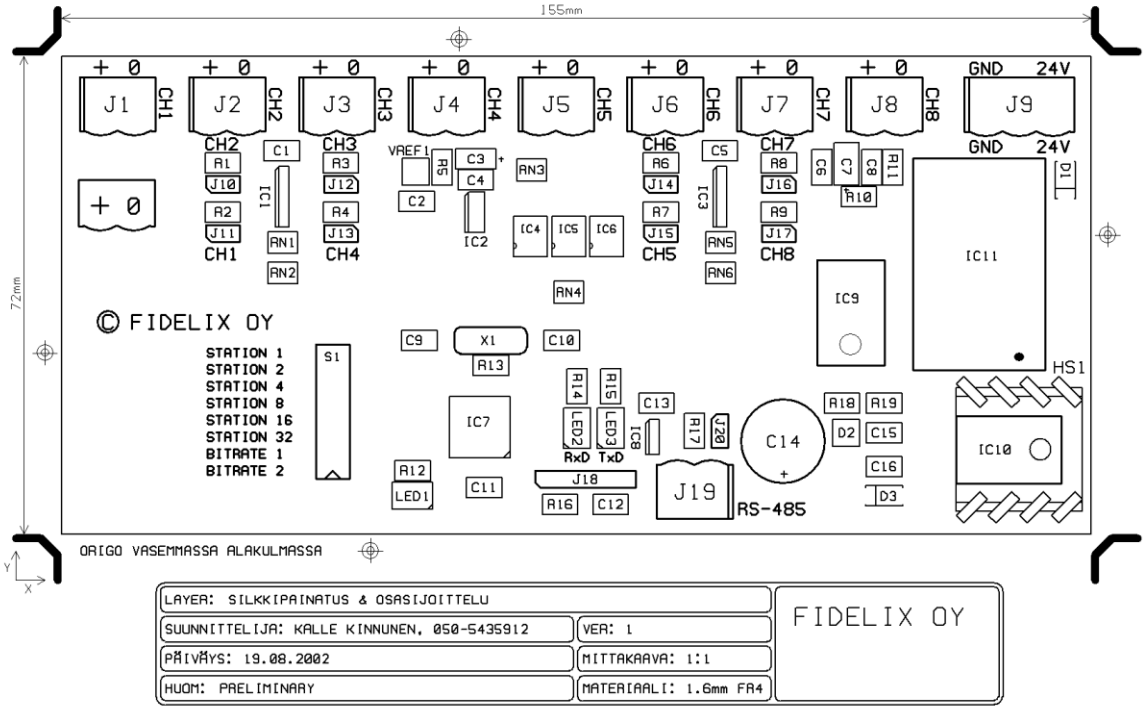

Figure 21, 8-Channel Analog Output Module, component placement

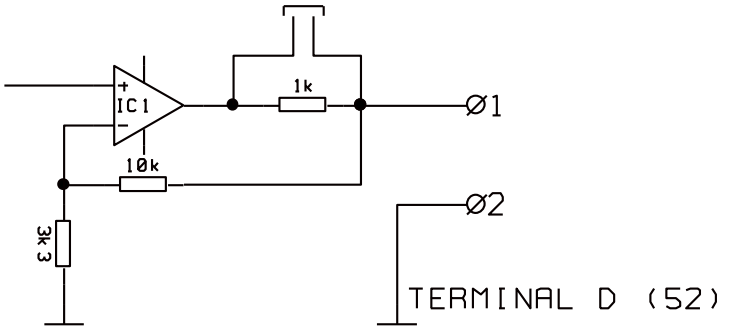

Figure 22, Analog output circuit (Channel 1)

Description

AO-8 module has 8 independently controlled analog output channels. Analog voltage can be programmed from 0V to 10V. Outputs are isolated from other circuit. Polarity of connectors is printed on PCB and also seen in figure 21. Jumpers J10 – J17 can be used to bypass 1 kilo-ohm (as seen in figure 22) output resistor. Bypassing 1 kilo-ohm resistor (using jumper) is needed if high current output is required, but it is not recommended, because 1 kilo-ohm resistor works as a protective circuit.

Voltage Output

Usually analog outputs are used as voltage outputs. Output voltage can be programmed from 0 volt to 10 volt. When used as voltage output, 1 kilo-ohm resistor connected to output circuit works as protective circuit and has no effect to functionality of output. If input impedance of controlled circuit is low, it may be necessary to bypass resistor using jumper to achieve lower driving impedance and thus more current. Output circuits are galvanic isolated from their control circuit and power supply, but there is galvanic connection between each output channel. If galvanic isolation is not needed power supply connectors middle terminal (marked D or 52) should be connected to power supply zero potential.

Driving relays

Analog outputs can be used to drive relays, but there are some restrictions. Usually it is recommended to use relay module. When driving relays directly from analog outputs, limited current driving capability of outputs has to be considered, relays should be low current models. Normally output current is limited to 10mA/output using 1 kilo-ohm resistor connected in output circuit. When driving relays directly this resistor has to be bypassed using resistor to achieve enough current. Afted bypassing resistor, inbuild current limiting is no longer in use and user has to make sure that too much current is not drawn. Outputs 1-4 and outputs 5-8 are driven by different IC, so if high current driving is necessary it should be taken evenly from these two groups. For example if it is needed to drive two relays directly from analog outputs it is recommended to connect one to output 1 and other to output 5. Recommended maximum current per output is 20mA, total current taken from all outputs together may never exceed 120mA. Reverse polarity diode (fly-wheel diode) has to be connected parallel with relay coil to avoid voltage spikes. Figure 23 shows how to connect relay directly to analog output.

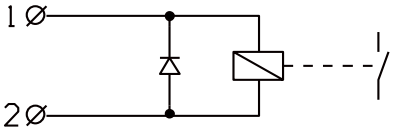

Figure 23, Driving relay directly

If more that one relay is needed to drive simultaneously or used relays needs too high current to be taken directly from analalog output, a transistor drive circuit can be used. Besides it is recommended to use transistors anyway, because it reduces current taken from output and protective circuit stays unaltered. Figure 24 shows how to drive relay using transistor. When driving high current relays, power supply has to be connected directly to coil and analog ground has to be connected to power supply zero potential. If galvanic isolation is needed a opto-coupler can be used as in figure 24. When using opto-coupler circuit, bypass jumper may **not** be connected.

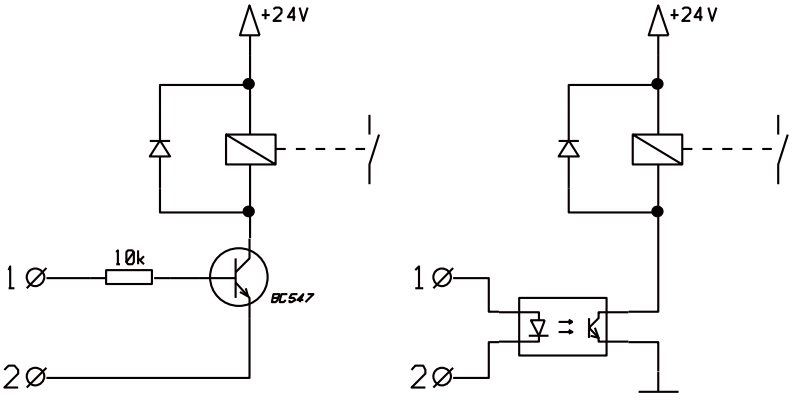

Figure 24, Driving relay using transistor or opto-coupler

Current output

Analog output can be used as current output, but there is some restrictions. Normally output current is limited to 10mA/output using 1 kilo-ohm resistor connected in output circuit. When driving current loops this resistor has to be bypassed using resistor to achieve 20mA current. Afted bypassing resistor, inbuild current limiting is no longer in use and user has to make sure that too much current is not drawn. It is also important to notice that maximum output voltage is 10 volts, which may not be enough to achieve 20mA current in all systems. Module

has no current feed back, so current depends of circuits input impedance. If feed back is needed it has to be made using some measurement circuit, for example using AI-8 module. Same load current regulations will apply as when driving relays.

Register settings

Register 0, analog output control register for output 1, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	X	X	X	X	X	X	MSB									LSB

X = NOT USED

This register controls output voltage of output channel 1. Only 10 least significant bits are used. Output voltage value can be calculated using following equation,

$$U = \frac{register\_value}{1023} \cdot 10V$$

This register is read and write register.

Registers 1-7, analog output control register for outputs 2-8, read/write

Same as register 0 but for analog channels 2 to 8. These registers are read and write registers.

Register 8, channel enable control register, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG8	X	X	X	X	X	X	X	X	ENA8	ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1

X = NOT USED

Register 8 controls enabling of analog output channels. If bit is is set, corresponding output channel is enabled. If bit is cleared, output channel is disabled and its is pulled down to GND using 100 kilo-ohm resistor. Only 8 least significant bits are used. This register is read and write register.

Register 9, preset value for channel 1, read/write (beginning from firmware version 1.10 and 2.20)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG9	MODE	X	X	X	X	X	MSB									LSB

X = NOT USED

Register 9 has preset value for channel 1. Preset value is used when power is turned on and substation has not been sending new data. Highest bit of register determines for each channel independently if preset value is in use. If bit is set, preset value is taken from lowest ten bits of register. Otherwise current state of output is preserved. Same Preset value is also used if communication failure occurs. Communication failure means that mudule hasn't got any modbus messages for 30 seconds. This register is read and write register.

Registers 10-16, preset values for channels 2-8, read/write (beginning from firmware version 1.10 and 2.20)

Same as register 9, but for analog channels 2 to 8. These registers are read and write register.

8-Channel Security Interface Module (SI-8)

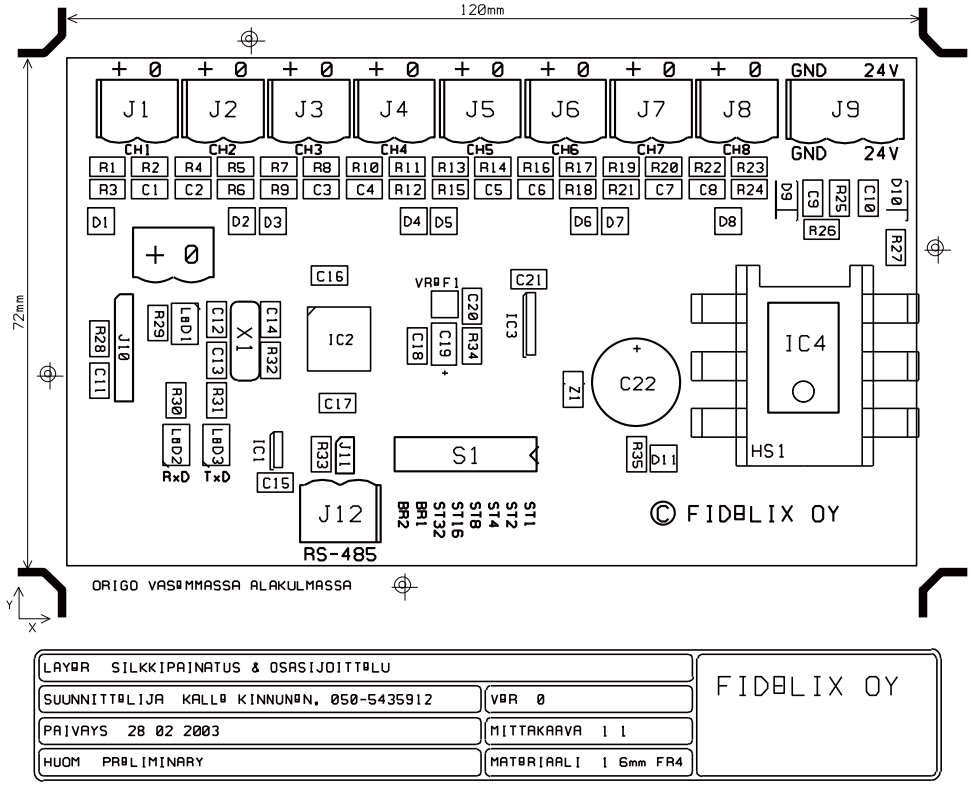

Figure 25, 8-Channel Security Interface Module, component placement

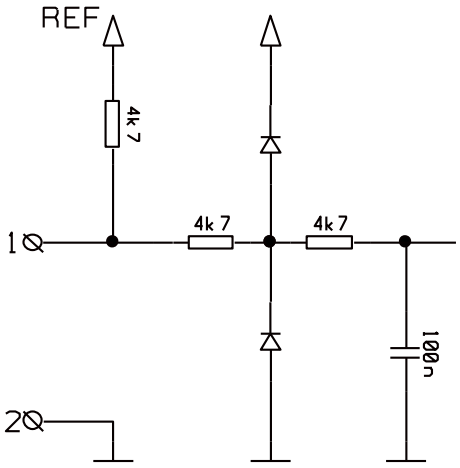

Figure 26, Input circuit (channel 1)

Description

Security module is used if analog measurements with short response time are needed. A typical situation where that is needed is resistance loops in security applications. In those applications it is important to notice even short pulses. Measurements are done using 8 channel multiplexed 10 bit AD-converter. Measured resistance value can vary between 470Ω - 47kΩ.

One measurement period takes 20 ms and each channel is measured every 160 ms as seen in picture 27. If two successive measurement results are within value set in hysteresis register, measurement is approved to valid. Valid measurement is compared to last saved measurement value and if difference between results is more than set in hysteresis register, new value is saved. Measurement values are saved in four level ring buffer. The oldest value

in ring buffer is moved to the corresponding register when register read thru modbus communication. This procedure ensures that all data is sended to substation even if communication speed is low. If substation anyhow misses to read all data from module and ring buffer gets full, the oldest value is replaced with new one. To indicate overflow of ring buffer, the highest bit of value is set. This way substation can detect if communication has failed.

If within five successive measurements is not possible to get valid signal, all signals are passed thru to ring buffer. This ensures that sabotage or indicator/loop malfunction will be detected immediately.

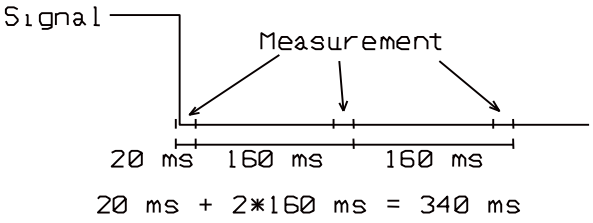

Figure 27, signal measurement in Security Module

Register settings

Register 0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	ERR	X	X	X	X	X	MSB									LSB

X = NOT USED

Register 0 cointains input channel 1 measurement data. Register value can be converted to resistance value using following equation,

$$R = \frac{4700\Omega \cdot register\_value}{1023 - register\_value}.$$

Conversion is usually not needed, because limits can be set using register values directly. Measurement value is prented using only 10 lowest bits of register. The highest bit of register is used as overflow indicator of ring buffer. Is measurement channels four level ring buffer overflows, this bit is set. This way substation can detect overflow condition and notice that communication or mearement loop is not functioning well. This register is a read only register.

Registers 1-7

Same as register 0 but for analog channels 2 to 8. These registers are read only registers.

Register 8, hysteresis, read/write

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG0	X	X	X	X	X	X	X	X	MSB							LSB

X = NOT USED

This register set hysteresis level. Smaller changes, than set in this register, are discarded. Two successive measurements has to be within hysteresis to be accepted as a valid measurement. This register is a read/write register.

36-Channel Combi module (COMBI-36), version v1.10

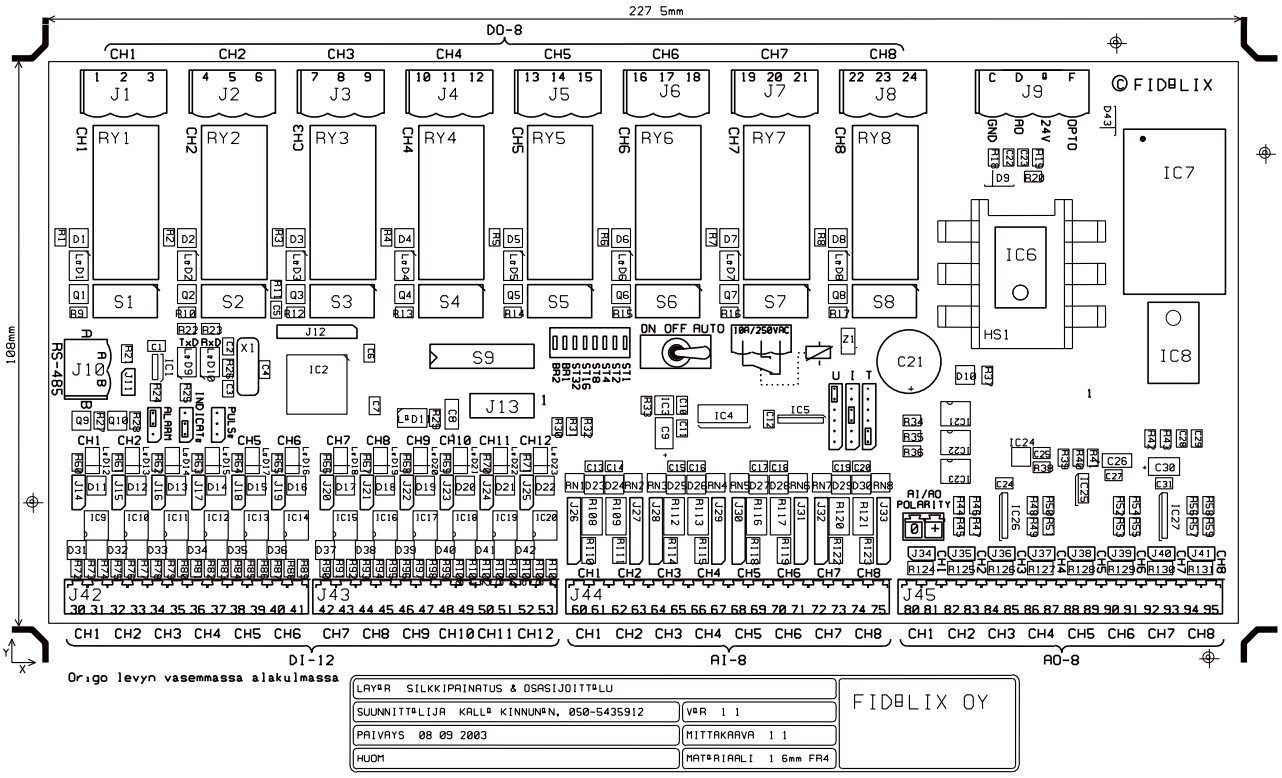

Figure 28, 36-Channel Combi Module, component placement

Description

Combi module combines 12 digital inputs, 8 relay outputs, 8 analog inputs and 8 analog outputs. Combi module is designed to be compatible with each corresponding module and thus it is seen as four different modules. Address of digital inputs is selected using dip-switch and digital inputs will be presented as DI-16 module. Relay outputs will get next address and then it will be presented as DO-8 module. Analog inputs get third address and they are presented as AI-8 module, analog outputs get fourth address and they are presented as AO-8 module. For example if address 10 is selected, modules will have following addresses:

Signal	Corresponding module	Address
Digital inputs	DI-16	10
Relay outputs	DO-8	11
Analog inputs	AI-8	12
Analog outputs	AO-8	13

All modules, excluding DI-16, are identical to their corresponding modules, both physical circuit and register structure. Connector numbering is different (see appendix C) and physical size of connectors. Digital input count is different compared to DI-16 module, combi module has 12 inputs compared to DI-16 modules 16. Anyway register structure is kept similar to maintain compatibility between module types. Inputs 13-16 have own registers (registers 12-13 and 26-29 and bits 12-15 in registers 0-5), but those are not used because those inputs don't exist. Registers allocated to inputs 13-16 have no effect to functionality of module. More specific instructions and register structure are explained in part dealing with each different module type.

Combi module can be identified from single modules by reading device identification data. This data is read using “Read Device Identification” modbus function. ProductName field for combi module is “36-Channel Combi Module”.

Appendix A, Modbus exception codes

MODBUS Exception Codes		
Cod	**Name**	**Meaning**
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the module. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the module. More specifically, the combination of reference number and transfer length is invalid. For a controller with 10 registers, a request with offset 6 and length 4 would succeed, a request with offset 6 and length 5 will generate exception 02.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the module. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.

Appendix B, CRC-calculation example

CRC-calculation example using C.

```
unsigned int crc16 (unsigned char *data, unsigned char amount)
{
unsigned int CRC = 0xFFFF;
unsigned char i = 0, j = 0;

for (j=0; j< amount; j++)
    {
    CRC = CRC ^ data[j];
    for (i=0; i < 8; i++)
        {
        if ((CRC & 0x0001))
            {
            CRC = CRC >> 1;
            CRC = CRC ^ 0xA001;
            }
        else
            CRC = CRC >> 1;
        }
    }
return CRC;
}
```

Appendix C, Numbering of terminals

16-Channel Digital Input Module

terminal	function
1	Input 1, from field
2	Input 1, to field
3	Input 2, from field
4	Input 2, to field
5	Input 3, from field
6	Input 3, to field
7	Input 4, from field
8	Input 4, to field
9	Input 5, from field
10	Input 5, to field
11	Input 6, from field
12	Input 6, to field
13	Input 7, from field
14	Input 7, to field
15	Input 8, from field
16	Input 8, to field
17	Input 9, from field
18	Input 9, to field
19	Input 10, from field
20	Input 10, to field
21	Input 11, from field
22	Input 11, to field
23	Input 12, from field
24	Input 12, to field
25	Input 13, from field
26	Input 13, to field
27	Input 14, from field
28	Input 14, to field
29	Input 15, from field
30	Input 15, to field
31	Input 16, from field
32	Input 16, to field
C (51)	0V
D (52)	Opto-coupler power supply (24VDC)
E (53)	Power supply 24VDC
A (55)	RS-485 Data+
B (56)	RS-485 Data-

8-Channel Relay Output Module

terminal	function
1	Output 1, Common
2	Output 1, N.O.
3	Output 1, N.C.
4	Output 2, Common

5	Output 2, N.O.
6	Output 2, N.C.
7	Output 3, Common
8	Output 3, N.O.
9	Output 3, N.C.
10	Output 4, Common
11	Output 4, N.O.
12	Output 4, N.C.
13	Output 5, Common
14	Output 5, N.O.
15	Output 5, N.C.
16	Output 6, Common
17	Output 6, N.O.
18	Output 6, N.C.
19	Output 7, Common
20	Output 7, N.O.
21	Output 7, N.C.
22	Output 8, Common
23	Output 8, N.O.
24	Output 8, N.C.

C (51)	0V
E (53)	Power supply 24VDC

A (55)	RS-485 Data+
B (56)	RS-485 Data-

N.O. = Normally Open
N.C. = Normally Closed

8-Channel Analog Input Module

terminal	function
1	Input 1, Vin
2	Input 1, 0V
3	Input 2, Vin
4	Input 2, 0V
5	Input 3, Vin
6	Input 3, 0V
7	Input 4, Vin
8	Input 4, 0V
9	Input 5, Vin
10	Input 5, 0V
11	Input 6, Vin
12	Input 6, 0V
13	Input 7, Vin
14	Input 7, 0V
15	Input 8, Vin
16	Input 8, 0V
C (51)	0V
D (52)	Not Connected
E (53)	Power supply 24VDC
A (55)	RS-485 Data+
B (56)	RS-485 Data-

8-Channel Analog Output Module

terminal	function
1	Output 1, Vout
2	Output 1, 0V
3	Output 2, Vout
4	Output 2, 0V
5	Output 3, Vout
6	Output 3, 0V
7	Output 4, Vout
8	Output 4, 0V
9	Output 5, Vout
10	Output 5, 0V
11	Output 6, Vout
12	Output 6, 0V
13	Output 7, Vout
14	Output 7, 0V
15	Output 8, Vout
16	Output 8, 0V
C (51)	0V
D (52)	Analog ground
E (53)	Power supply 24VDC
A (55)	RS-485 Data+
B (56)	RS-485 Data-

8-Channel Security Interface Module

terminal	function
1	Input 1, Vin
2	Input 1, 0V
3	Input 2, Vin
4	Input 2, 0V
5	Input 3, Vin
6	Input 3, 0V
7	Input 4, Vin
8	Input 4, 0V
9	Input 5, Vin
10	Input 5, 0V
11	Input 6, Vin
12	Input 6, 0V
13	Input 7, Vin
14	Input 7, 0V
15	Input 8, Vin
16	Input 8, 0V
C (51)	Power supply 24VDC
D (52)	Not Connected
E (53)	0V
A (55)	RS-485 Data+
B (56)	RS-485 Data-

36-Channel Combi Module

terminal	function
1	Relay Output 1, Common
2	Relay Output 1, N.O.
3	Relay Output 1, N.C.
4	Relay Output 2, Common
5	Relay Output 2, N.O.
6	Relay Output 2, N.C.
7	Relay Output 3, Common
8	Relay Output 3, N.O.
9	Relay Output 3, N.C.
10	Relay Output 4, Common
11	Relay Output 4, N.O.
12	Relay Output 4, N.C.
13	Relay Output 5, Common
14	Relay Output 5, N.O.
15	Relay Output 5, N.C.
16	Relay Output 6, Common
17	Relay Output 6, N.O.
18	Relay Output 6, N.C.
19	Relay Output 7, Common
20	Relay Output 7, N.O.
21	Relay Output 7, N.C.
22	Relay Output 8, Common
23	Relay Output 8, N.O.
24	Relay Output 8, N.C.
30	Digital Input 1, to field
31	Digital Input 1, from field
32	Digital Input 2, to field
33	Digital Input 2, from field
34	Digital Input 3, to field
35	Digital Input 3, from field
36	Digital Input 4, to field
37	Digital Input 4, from field
38	Digital Input 5, to field
39	Digital Input 5, from field
40	Digital Input 6, to field
41	Digital Input 6, from field
42	Digital Input 7, to field
43	Digital Input 7, from field
44	Digital Input 8, to field
45	Digital Input 8, from field
46	Digital Input 9, to field
47	Digital Input 9, from field
48	Digital Input 10, to field
49	Digital Input 10, from field
50	Digital Input 11, to field
51	Digital Input 11, from field
52	Digital Input 12, to field
53	Digital Input 12, from field

60	Analog Input 1, 0V
61	Analog Input 1, Vin
62	Analog Input 2, 0V
63	Analog Input 2, Vin
64	Analog Input 3, 0V
65	Analog Input 3, Vin
66	Analog Input 4, 0V
67	Analog Input 4, Vin
68	Analog Input 5, 0V
69	Analog Input 5, Vin
70	Analog Input 6, 0V
71	Analog Input 6, Vin
72	Analog Input 7, 0V
73	Analog Input 7, Vin
74	Analog Input 8, 0V
75	Analog Input 8, Vin

80	Analog Output 1, 0V
81	Analog Output 1, Vout
82	Analog Output 2, 0V
83	Analog Output 2, Vout
84	Analog Output 3, 0V
85	Analog Output 3, Vout
86	Analog Output 4, 0V
87	Analog Output 4, Vout
88	Analog Output 5, 0V
89	Analog Output 5, Vout
90	Analog Output 6, 0V
91	Analog Output 6, Vout
92	Analog Output 7, 0V
93	Analog Output 7, Vout
94	Analog Output 8, 0V
95	Analog Output 8, Vout

C	0V
D	Analog output ground
E	Power supply 24VDC
F	Opto-coupler power supply (24VDC)

A (55)	RS-485 Data+
B (56)	RS-485 Data-

N.O. = Normally Open
N.C. = Normally Closed