DigiMux - DigIO Modbus Interfacing

1.0 General

Both the DigiMux and the DigIO support interfacing to a Modbus protocol. This is switch selectable to be either an ASCII or an RTU Modbus protocol. The Modbus address or the H2NS address is controlled by switch settings in both the DigiMux and the DigIO. In the DigiMux, this is switch #1 located on the LCD display board. All switches Off yields an H2NS address of 00_H.

All switches Off, or only switch position #1 On, yields a Modbus address of 01_H. A Modbus address of 17 (decimal) would have switch positions #5 and #1 On and all others Off. In the DigIO the switch number is 05 located on the main mother PC board. The comm port Baud rates are also switch selectable in both units, and are installed on the same PC boards as the address switches. The settings are defined below.

DigiMux – Switch #2 Position	Function	DigIO – Switch #05 Position
1	Comm Port #4 Baud	1
2	Comm Port #4 Baud	2
3	Off = DigiMux, On = DigIO	3
4	Off = H2NS, On = Modbus	4
5	Off = RTU, On = ASCII	5
6	Off = Regulation CS, On = Wlidcard ?? CS	6
7	Comm Port #1 Baud	7
8	Comm Port #1 Baud	8

Baud Rates – Comm #4

1	2	
Off	Off	2400 Baud
On	Off	4800 Baud
Off	On	9600 Baud
On	On	19.6K Baud

Baud Rates – Comm #1

7	8	
Off	Off	2400 Baud
On	Off	4800 Baud
Off	On	9600 Baud
On	On	19.6K Baud

Factory defaults are 9600 Baud for both comm ports, no parity, 1 stop bit, H2NS operation, and an ID of $00_{H.}$ If the Modbus switch is On and the RTU option selected, the factory defaults are, 9600 Baud, 8 data bits, 1 stop bit, even parity, ID of 01_H and all registers starting at the base addresses. If the ASCII option is selected, the factory defaults are 8 data bits, no parity, 1 stop bit 9600 Baud, ID of 01_H and all registers starting at the base address.

Positions #7 and 8 on switch #6 select the parity option. Presented below are the parity options

Parity Bit Options
Switch #6, Positions #7, 8

8	7	Parity
0	x	No parity
1	1	Odd parity
1	0	Even parity

Switch #5, position #6 allows the use of a wildcard CRC or LRC character. With switch position #6 on, the units will accept two question marks (??) as a proper check character and will respond normally.

Switch #6, positions 1 – 6, determines the ID of the DigIO unit. With all positions off, the ID is 00_H for H2NS protocol, and 01_H for Modbus protocol. Either a 00_H or a 01_H will be interpreted as a 01_H ID for Modbus operation. An ID 0f 17_H would have positions #5 and #1 On, and all others Off.

2.0 Modbus Function Formats

All addresses in Modbus are referenced to zero. The first occurrence of a data item is addresses as item number zero. For example:

V coil 1 in a programmable controller is addressed as coil 0000 in the address field of a Modbus message.

V coil 127_{10} is addressed as coil $007E_H$ (126 decimal).

V holding register 40001 is addresses as register 0000 in the data address field of the message. The function code field already specifies a holding register operation. therefore, the 4*x* reference is implicit.

V holding register 40108 is addressed as register $006B_H$ (107 decimal).

The slots in a DigiMux or DigIO device are numbered from 1 …16. This corresponds to 0000 to $000F_H$ in Modbus register nomenclature.

The following tables show examples of a Modbus query and normal response. Both examples show the field contents in hexadecimal (subscript H), and also show how a message could be framed in ASCII or in RTU mode.

Query

Field Name	Example	ASCII Characters	RTU 8 bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function Code	03	0 3	0000 0011

Starting Add Hi	00	0 0	0000 0000
Starting Add Lo	6B	6 B	0110 1011
No. of registers Hi	00	0 0	0000 0000
No. of registers Lo	03	0 3	0000 0011
Error Check		LRC (2 char)	CRC (16 bits)
Trailer		Cr,Lf	None
Total Bytes		17	8

Response

Field Name	Example	ASCII Characters	RTU 8 bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function Code	03	0 3	0000 0011
Byte Count	06	0 6	0000 0110
Data Hi	02	0 2	0000 0010
Data Lo	2B	2 B	0010 1011
Data Hi	00	0 0	0000 0000
Data Lo	00	0 0	0000 0000
Data Hi	00	0 0	0000 0000
Data Lo	63	6 3	0110 0011
Error Check		LRC (2 Char)	CRC (16 bits)
Trailer		Cr,Lf	None

The master query is a Read Holding Register request to slave device address 06. The message requests data from three holding registers, 40108 … 40110. (Note: the message specifies the starting register as 0107 ($006B_H$).

The slave response echoes the function code, indicating this is a normal response. The Byte Count field specifies how many eight-bit data items are being returned. It shows the count of eight-bit bytes to follow in the data, for either ASCII or RTU. With ASCII, this value is half the actual of ASCII characters in the data. In ASCII, each four-bit hexadecimal value requires on ASCII character, therefore two ASCII characters must follow in the message to contain each eight-bit data item.

For example, the value 63 hex is sent as on eight-bit byte in RTU mode (01100011). The same value sent in ASCII mode requires two bytes, for ASCII 6(00110110) and 3 (00110011). The Byte Count field counts this as one eight-bit data item, regardless of the character framing method, either ASCII or RTU.

To construct a Byte Count value, use only the count of eight-bit bytes in the message data. The value is exclusive of all other field contents, including the Byte Count field.

3.0 Commands supported

The commands that the units will respond to are presented in Table 1.0 below.

Table 1.0
Modbus commands

Command	Function code

1.	Read coil status	01	
2.	Read input status	02	
3.	Read input register	04	DigiMux only
4.	Force multiple coils	15	
5.	Preset multiple registers	16	DigiMux only
6.	Report slave ID	17	
7.	Force Single Relay	05	

4.0 Command definition

The command definitions are presented below.

4.1 Read Relay Status (0*x* references, coils)

Reads the On/Off status of the relay outputs. Broadcast is not supported. This command is uses to read the present setting of the relay outputs of the unit. Addresses 00000-00015 correspond to slot positions 1 … 16 in the DigiMux and DigIO.

Query

The query message specifies the starting relay and quantity of relays to be read. Relays are addressed starting at zero-relays. Sixteen relays are addressed as 0 … 15. Here is an example of a query to read relays 20 … 56 from slave device 17:

Field Name	Example
Slave Address	11
Function Code	01
Starting Address Hi	00
Starting Address Lo	13
Number of Points Hi	00
Number of Points Lo	25
CRC Characters	LRC or CRC

Response

The relay status in the response message is packed as one relay per bit of the data field. Status is indicated as:1 = On; 0 = Off. The LSB of the first byte contains the relay addressed in the query. The other relays follow toward the high order end of the byte, and from low order to high order in subsequent bytes.

If the returned relay quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeroes (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data. Here is an example of a response to the query.

Field name	Example
Slave Address	11
Function Code	01
Byte Count	05
Data (relays 27…20)	CD
Data (relays 35…28)	6B
Data (relays 43…36)	B2

Data (relays 51…44)	0E
Data (relays 56…52)	1B
Error Check	LRC or CRC

The status of relays 27 … 20 is shown as the byte value CD_H, or binary 1100 1101. Relay 27 is the MSB of this byte, and coil 20 is the LSB. Left to right the status of relays 27 … 20 is On-On-Off-Off-On-On-Off-On.

By convention, bits within a byte are shown with the MSB to the left, and the LSB to the right. Thus the relays in the first byte are 27 …20, from left to right. The next byte has relays 35 … 28, left to right. As the bits are transmitted serially, they flow from LSB to MSB: 20 …27, 28 …30, and so on.

In the last data byte, the status of relays 56 …52 is shown as the byte value $1B_{H,}$ or binary 0001 1011. Relay 56 is in the fourth bit position from the left, and relay 52 is the LSB of this byte. The status of relays 56 … 52 is On-On-Off-On-On. The remaining three bits (towards the high order end) are zero filled.

4.2 Read input status (1*x* references)

Reads the On/Off status of the digital outputs. Broadcast is not supported. This command is used to read the incoming digital inputs. Addresses 10000-10015 correspond to slot positions 1 … 16 in the DigiMux and DigIO.

Query

The query message specifies the starting input and quantity of inputs to be read. Inputs are addressed starting at zero-inputs. Sixteen inputs are addressed as 0 … 15. Here is an example of a query to read inputs 10009 … 10016 from slave device 17:

Field Name	Example
Slave Address	11
Function Code	01
Starting Address Hi	00
Starting Address Lo	08
Number of Points Hi	00
Number of Points Lo	08
CRC Characters	LRC or CRC

Response

The input status in the response message is packed as one input per bit of the data field. Status is indicated as:1 = On; 0 = Off. The LSB of the first byte contains the input addressed in the query. The other inputs follow toward the high order end of the byte, and from low order to high order in subsequent bytes.

If the returned input quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeroes (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data. Here is an example of a response to the query.

Field name	Example
Slave Address	11
Function Code	01
Byte Count	01
Data (inputs 10016 …100009)	CD
Error Check	LRC or CRC

The status of inputs 16 … 09 is shown as the byte value CD_H, or binary 1100 1101. Input 16 is the MSB of this byte, and input 09 is the LSB. Left to right the status of coils 16 … 09 is On-On-Off-Off-On-On-Off-On.

By convention, bits within a byte are shown with the MSB to the left, and the LSB to the right. Thus the inputs in the byte are 16 …09, from left to right.

4.3 Read Input Registers (3xreferences)

Reads the binary contents of input registers in the slave. Broadcast is not supported. This command is used to retrieve binary data from the analog to digital converters. The input registers 30000 – 300015 correspond to slot positions 1 … 16 in the DigiMux.

Query

The query message specifies the starting register and quantity of registers to be read. Registers are addressed starting at zero-registers 1 …16 are addressed as 0 … 15. Here is an example of a request to read register 30009 from slave device 17.

Field Name	Example
Slave add	11
Function	04
Starting Add Hi	00
Starting add Lo	08
Number to ret Hi	00
Number to ret Lo	01
Error Check	CRC

Response

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. Here is an example of a response to the above query.

Field Name	Example
Slave add	11
Function	03
Byte count	02
Data Hi Reg 30009	07
Data Lo Reg 30009	D0
Error Check	CRC

The contents or register 30009 are shown as the two byte values 0f 07 D0 hex, or 2000 decimal. If a slot is read that does not have an analog to digital converter installed, the unit returns an 03 error code response as presented in section 4.4.

4.4 Force Multiple Relays (0x reference)

Forces each relay in a sequence of relays to either On or Off. This is used to set and reset the relay outputs of the DigiMux and DigIO. The addresses 00000-00015 correspond to slot positions 1 … 16 in the DigiMux and DigIO. Setting a slot position without a relay installed in the slot to an Off state will not return an error condition. Setting a slot position without a relay installed to an On state will result in an error code of 03 being returned. Other slot positions with relays installed will be set/reset accordingly.

Query

The query message specifies the relay references to be forced. Relays are addressed starting at zero-relay 1 is addressed as 0. The requested On/Off states are specified by contents of the query data field. A logical 1 in a bit position of the field sets the corresponding relay On. A logical 0 turns the relay Off.

The following shows an example of a query to force a series of ten relays starting at relay 1 (addressed as 0, 0r 00_H) in slave device 01. The query data contents are two bytes: CD_H 01_H (1100 1101 0000 0001 binary). The binary bits correspond to the relays in the following way:

```
Bit     1  1  0  0  1  1  0  1  0  0  0  0  0  0  0  1
Relay   8, 7, 6, 5, 4, 3, 2, 1 16,15,14,13,12,11,10, 9
```

The first byte transmitted (CD_H) addresses bits 8 … 1, with the least significant bit addressing the lowest relay. The next byte transmitted (01H) addresses relays 10 and 9, with the least significant bit addressing relay 9 in this set. Unused bits in the last data byte should be zero filled.

Field Name	Example
Slave add	11
Function	0F
Relay add Hi	00
Relay add Lo	00
# relays Hi	00
# relays Lo	0A
Byte count	02
Force relay 8 … 1	CD
Force relay 16 … 9	01
Error Check	CRC

Response

The normal response returns the slave address, function code, starting address, and quantity of relays forced. Here is an example of a response to the above query.

Field Name	Example
Slave add	17
Function	0F
Relay add Hi	00

Relay add Lo	00
# relays Hi	00
# relays Lo	0A
Error Check	CRC

If a query is received that tries to set a relay in a slot position the does no have a relay installed, the following error code is returned. All other bits properly addressing slots with relays are set or reset as commanded.

Error Response

Field Name	Example
Slave add	17
Function	8F
Error code	03
Error Check	CRC

The 03 indicates an illegal data value in the query data field.

4.5 Preset Multiple Registers (4x references)

Preset values into a sequence of holding registers. This is used to send analog output values to the digital to analog converters. The addresses 40000-40015 correspond to slot positions 1 … 16 in the DigiMux.

Query

The query message specifies the register references to be preset. Registers are addressed starting at zero-register 1 is addressed as 0. The preset values are specified in the query data field. Data is packed as two bytes per register. Here is an example of a query to preset two registers starting at 40002 to 00 0A and 01 02 hex, in slave address 05.

Field Name	Example
Slave add	05
Function	10
Starting add Hi	00
Starting add Lo	01
Number of reg Hi	00
Number of reg Lo	02
Byte count	04
Data1 Hi	00
Data1 Lo	0A
Data2 Hi	01
Data2 Lo	02
Error Check	CRC

Response

The normal response returns the slave address, function code, starting address, and quantity of registers preset. Here is an example of a response to the query shown above.

Field Name	Example
Slave add	05
Function	10
Starting add Hi	00
Starting add Lo	01
Number of reg Hi	00
Number of reg Lo	02
Error Check	CRC

If a slot is addressed that does not have a digital to analog converter installed, the 03 error message discussed above is returned. Slots addressed properly will be updated with the new register value.

4.6 Report Slave ID

This query returns a description of the type and capabilities of the installed unit. An example is given below, and the various responses are provided.

Query

Here is an example of a request to report the type and status of slave device 17.

Field Name	Example
Slave add	11
Function	11
Error Check	CRC

Response

The format of the response is shown below

DigiMux

Field Name	Example
Slave add	11
Function	11
Byte count	07
Type	02
Digital I/O Hi	00
Digital I/O Lo	FE
Analog Out Hi	F0

DigIO

Field Name	Example
Slave add	11
Function	11
Byte count	02
Type	01
# Digital I/O	10 or 8
Error Check	CRC

Analog Out Lo	00
Analog In Hi	0C
Analog In Lo	00
Error Check	CRC

The DigIO response is straightforward. It either has 8 or 16 digital inputs and outputs. The DigiMux response provides somewhat more information. The slot positions in the DigiMux are numbered 1 … 16, which corresponds to 0 … 15 in the Modbus addressing scheme. The information returned above provides the slot number that each type of DigiPak is installed.

In the above example;

The digital I/O modules are installed in slot positions 8, 7, 6, 5, 4, 3, and 2, corresponding to the FE_H in the LSB. This corresponds to addresses 00001 … 00007. The analog output modules are installed in slot positions 16, 15, 14, and 13, corresponding to $F0_H$ in the MSD. This corresponds to addresses 00015 …00012 Analog output modules are installed in slot positions 12 and 11 which corresponding to the 0CH in the MSD. This corresponds to addresses 00011 and 00010.

4.7 Force Single Relay

This query allows a single relay to be set and reset, and in addition, allows the relay to be set as a momentary closure.

Query

The query message specifies the relay reference to be set or reset. The On/Off state is specified in the query field. A value of FF 00 hex sets the relay and a value of 00 00 hex resets the relay. If the command is to set the relay and the second byte in the data field is not zero, this is interpreted as a momentary closure. The value specified in the second byte is accepted as the number of seconds to close the relay. A value 0f FF 05 hex, would specify a relay closure of 5 seconds.

Field Name	Example
Slave add	11
Function	05
Relay add Hi	00
Relay add Lo	02
Force data Hi	FF
Force data Lo	05
Error Check	CRC

Response

The normal response is a return of the query.

5.0 Exception responses

Except for broadcast messages, when a master device sends a query to a slave device it expects a normal response. One of four possible events can occur from the master's query:

- If the slave device receives the query without a communication error, and can handle the query normally, it returns a normal response.
- If t he slave does not receive the query due to a communication error, no response is returned. The master program will eventually process a timeout condition for the query.
- If the slave receives the query, but detects a communication error (parity, LRC, or CRC), no response is returned. The master program will eventually process a timeout condition for the query.

If the slave receives the query without a communication error, but cannot handle it (for example, if the request is to read a non-existent relay or register), the slave will return an exception response informing the master of the nature of the error.

The exception response message has two fields that differentiate it from a normal response:

Function Code Field: In a normal response, the slave echoes the function code of the original query in the function code field of the response. All function codes have a most-significant bit (MSB) of O (their values are all below 80 hexadecimal). In an exception response, the slave sets the MSB of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response. With the function code's MSB set, the master's application program can recognize the exception response and can examine the data field for the exception code.

Data Field: In a normal response, the slave may return data or statistics in the data field (any information that was requested in the query). In an exception response, the slave returns an exception code in the data field. This defines the slave condition that caused the exception.

Shown below is an example of a master query and slave exception response. The field examples are shown in hexadecimal.

QUERY

Byte	Contents	Example
1	Slave Address	OA
2	Function	01
3	Starting Address Hi	04
4	Starting Address Lo	AI
5	No. of Coils Hi	00
6	No. of Coils Lo	01
7	LRC	48

EXCEPTION RESPONSE

Byte	Contents	Example
1	Slave Address	OA
2	Function	81
3	Exception Code	02

Master Query and Slave Exception Response

In this example, the master addresses a query to slave device 10 (OA hex). The function code (01) is for a Read Relay Status operation. It requests the status of the relay at address 1245 (04A1 hex). Note that only that one relay is to be read, as specified by the number of relays field (0001).

If the relay address is non-existent in the slave device, the slave will return the exception response with the exception code shown (02). This specifies an illegal data address for the slave.

A listing of exception codes is presented below.

Exception Codes

Code	**Name**	**Meaning**
01	ILLEGAL FUNCTION	The-function code received in the query is not an allowable action for the slave. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the slave
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the slave.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the slave was attempting to perform the requested action.
05	ACKNOWLEDGE	The slave has accepted the request and is processing it, but a long duration of time wilt be required to do so. This

.

	response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
06 SLAVE DEVICE BUSY	The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.
07 NEGATIVE ACKNOWLEDGE	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave.
08 MEMORY PARITY ERROR	The slave attempted to read extended memory, but detected a parity error in the memory. The master can retry the request, but service may be required on the slave device.

6.0 LRC Generation – ASCII protocol

The Longitudinal Redundancy Check (LRC) field is one byte, containing an 8-bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results.

The LRC is calculated by adding together successive 8-bit bytes in the message, discarding any carries, and then two's complementing the result. The LRC is an 8 bit field, therefore each new addition of a character that would result in a value higher than 255 decimal simply 'rolls over' the field's value through zero. Because there is no ninth bit, the carry is discarded automatically

A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting 'colon' and ending CRLF. Add them into an 8-bit field, so that carries will be discarded.

2. Subtract the final field value from FF hex (all I's), to produce the ones-complement.

3. Add 1 to produce the twos-complement.

Placing the LRC into the Message

When the 8-bit LRC (2 ASCII characters) is transmitted in the message, the high-order character will be transmitted first, followed by the low-order character. For example, if the LRC value is 61 hex (0110 0001), the high order byte is 36_H and the low order byte is 31_H.

Colon	Addr	Func	Data Count	Data	Data	Data	Data	LRC Hi	LRC Lo	CR	LF

LRC Character Sequence

7.0 CRC Generation

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a I6-bit register to all 1's. Then a process begins of applying successive 8 bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit if one is used, do not apply to the

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive Ored with a preset, fixed value ($A001_H$). If the LSB was a O, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1. Load a 16-bit register with FFFF hex (all I's). Call this the CRC register.

2. Exclusive OR the first 8-bit byte of the message with the low-order byte of the 16-bit CRC register, putting the result in the CRC register.

3. Shift the CRC register one bit to the right (toward the LSB), zero-filling the MSB. Extract and examine the LSB.

4. (If the LSB was 0): Repeat Step 3 (another shift).
   (If the LSB was I): Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).

5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.

6. Repeat Steps 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.

7. The final contents of the CRC register is the CRC value.

Placing the CRC into the Message

When the 16-bit CRC (2 8-bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte. For example, if the CRC value is 1241 hex (0001 0010 0100 0001): The low order byte is 41H and the high order byte is 12_H.

Colon	Addr	Func	Data Count	Data	Data	Data	Data	CRC Lo	CRC Hi		

CRC Byte Sequence