EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

2390 EMAC, Way
Carbondale, IL 62902
Phone: (618) 529-4525
Fax: (618) 457-0110
http://www.emacinc.com

iPac HCS12 Modbus Users Manual

Rev1.3

iPac HCS12 Modbus Users Manual Rev1.3

Table of Contents

1. INTRODUCTION

The EMAC iPac HCS-12 conforms to Modbus protocol standard v1.1 over a serial line. This document describes the Modbus functions, protocols and registers implemented on the iPac and how they are specifically applied to control the hardware.

2. FUNCTIONS

The iPac implements the following standard Modbus functions:

Table1 iPac FUNCTIONS

Function	Code
READ_COILS	1
READ_DISCRETE_INPUTS	2
READ_HOLDING_REGISTERS	3
READ_INPUT_REGISTERS	4
WRITE_SINGLE_COIL	5
WRITE_SINGLE_REGISTER	6
DIAGNOSTICS	8
WRITE_MULTIPLE_COILS	15
WRITE_MULTIPLE_REGISTERS	16
MASK_WRITE_REGISTER	22

2.1 Function Descriptions

2.1.1 (0x01) Read Coils

This function code is used to read from 1 to 2000 contiguous statuses of coils in a remote device. The Request PDU specifies the starting address, i.e. the address of the first coil specified, and the number of coils. In the PDU Coils are addressed starting at zero. Therefore coils numbered 1-16 are addressed as 0-15.

The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1= ON and 0= OFF. The LSB of the first data byte contains the output addressed in the query. The other coils follow toward the high order end of this byte, and from low order to high order in subsequent bytes.

If the returned output quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data. In the iPac digital coils refer to the bi-directional digital ports 0,1, and 2. This function addresses the output lines of all 3 ports in one

memory map from 0-24. With Port0,0 corresponding to coil 0, Port0.1 is coil1 etc. Addressing a coil greater than 24 will result in an invalid address error.

Request
Function code 1 Byte **0x01**
Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of coils 2 Bytes 1 to 2000 (0x7D0)

Response
Function code 1 Byte **0x01**
Byte count 1 Byte **N***
Coil Status **n** Byte n = N or N+1
***N** = Quantity of Outputs / 8, if the remainder is different of 0 => N = N+1

Error
Function code 1 Byte **Function code + 0x80**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.2 (0x02) Read Discrete Inputs

This function code is used to read from 1 to 2000 contiguous statuses of discrete inputs in a remote device. The Request PDU specifies the starting address, i.e. the address of the first input specified, and the number of inputs. In the PDU Discrete Inputs are addressed starting at zero. Therefore Discrete inputs numbered 1-16 are addressed as 0-15.

The discrete inputs in the response message are packed as one input per bit of the data field. Status is indicated as 1= ON; 0= OFF. The LSB of the first data byte contains the input addressed in the query. The other inputs follow toward the high order end of this 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 zeros (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data. On the iPac the discrete inputs are the bi-directional ports 0,1 and 2. This function works identically to function (0x01) Read Coils.

Request
Function code 1 Byte **0x02**
Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of Inputs 2 Bytes 1 to 2000 (0x7D0)

Response
Function code 1 Byte **0x02**
Byte count 1 Byte **N***
Input Status **N*** x 1 Byte
***N** = Quantity of Inputs / 8 if the remainder is different of 0 => N = N+1

Error
Error code 1 Byte **0x82**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.3 (0x03) Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.

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. This function is used to read any register in the map marked as R/W or WO.

The Holding register memory map of the iPac is identical to the input register memory map except that the AtoD registers are not addressable and reading the digital port registers with this function returns the status of the output latch rather than the state of the lines.

Request
Function code 1 Byte **0x03**

Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of Registers 2 Bytes 1 to 125 (0x7D)

Response
Function code 1 Byte **0x03**
Byte count 1 Byte 2 x **N***
Register value **N*** x 2 Bytes
***N** = Quantity of Registers

Error
Error code 1 Byte **0x83**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.4 (0x04) Read Input Registers

This function code is used to read from 1 to approx. 125 contiguous input registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.

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
Function code 1 Byte **0x04**
Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of Input Registers 2 Bytes 0x0001 to 0x007D

Response
Function code 1 Byte **0x04**
Byte count 1 Byte 2 x **N***
Input Registers **N*** x 2 Bytes
***N** = Quantity of Input Registers

Error
Error code 1 Byte **0x84**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.5 (0x05) Write Single Coil

This function code is used to write a single output to either ON or OFF in a remote device. The requested ON/OFF state is specified by a constant in the request data field. A value of FF 00 hex requests the output to be ON. A value of 00 00 requests it to be OFF. All other values are illegal and will not affect the output.

The Request PDU specifies the address of the coil to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0. The requested ON/OFF state is specified by a constant in the Coil Value field. A value of 0XFF00 requests the coil to be ON. A value of 0X0000 requests the coil to be off. All other values are illegal and will not affect the coil. The normal response is an echo of the request, returned after the coil state has been written.

Request
Function code 1 Byte **0x05**
Output Address 2 Bytes 0x0000 to 0xFFFF
Output Value 2 Bytes 0x0000 or 0xFF00

Response
Function code 1 Byte **0x05**
Output Address 2 Bytes 0x0000 to 0xFFFF
Output Value 2 Bytes 0x0000 or 0xFF00

Error
Error code 1 Byte **0x85**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.6 06 (0x06) Write Single Register

This function code is used to write a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0. The normal response is an echo of the request, returned after the register contents have been written.

Request
Function code 1 Byte **0x06**
Register Address 2 Bytes 0x0000 to 0xFFFF
Register Value 2 Bytes 0x0000 or 0xFFFF

Response
Function code 1 Byte **0x06**
Register Address 2 Bytes 0x0000 to 0xFFFF
Register Value 2 Bytes 0x0000 or 0xFFFF

Error
Error code 1 Byte **0x86**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.7 08 (0x08) Diagnostics

MODBUS function code 08 provides a series of tests for checking the communication system between a client (Master) device and a server (Slave), or for checking various internal error conditions within a server.

The function uses a two-byte sub-function code field in the query to define the type of test to be performed. The server echoes both the function code and sub-function code in a normal response. Some of the diagnostics cause data to be returned from the remote device in the data field of a normal response.

In general, issuing a diagnostic function to a remote device does not affect the running of the user program in the remote device. User logic, like discrete and registers, is not accessed by the diagnostics. Certain functions can optionally reset error counters in the remote device.

A server device can, however, be forced into 'Listen Only Mode' in which it will monitor the messages on the communications system but not respond to them. This can affect the outcome of your application program if it depends upon any further exchange of data with the remote device. Generally, the mode is forced to remove a malfunctioning remote device from the communications system.

The following diagnostic functions are dedicated to serial line devices. The normal response to the Return Query Data request is to loopback the same data. The function code and sub-function codes are also echoed.

Request
Function code 1 Byte **0x08**
Sub-function 2 Bytes
Data N x 2 Bytes

Response
Function code 1 Byte **0x08**
Sub-function 2 Bytes
Data N x 2 Bytes

Error
Error code 1 Byte **0x88**
Exception code 1 Byte 01 or 03 or 04

2.1.8 Diagnostic Sub-Function Codes Supported By The Serial Line Devices

Here the list of sub-function codes supported by the serial line devices. Each sub-function code is then listed with an example of the data field contents that would apply for that diagnostic.

Sub-function code

Hex	Dec	Name
00	00	Return Query Data
01	01	Restart Communications Option
02	02	Return Diagnostic Register
03	03	Change ASCII Input Delimiter
04	04	Force Listen Only Mode
05 - 09		NOT USED
0A	10	Clear Counters and Diagnostic Register
0B	11	Return Bus Message Count
0C	12	Return Bus Communication Error Count
0D	13	Return Bus Exception Error Count
0E	14	Return Slave Message Count
0F	15	Return Slave No Response Count
10	16	Return Slave NAK Count
11	17	Return Slave Busy Count
12	18	Return Bus Character Overrun Count
13	19	PRIVATE
14	20	Clear Overrun Counter and Flag
21 ...		RESERVED

2.1.8.1 00 Return Query Data

The data passed in the request data field is to be returned (looped back) in the response. The entire response message should be identical to the request.

Sub-function Data Field (Request) Data Field (Response)
00 00 Any Echo Request Data

2.1.8.2 01 Restart Communications Option

The remote device serial line port must be initialized and restarted, and all of its communications event counters are cleared. If the port is currently in Listen Only Mode, no response is returned. This function is the only one that brings the port out of Listen Only Mode. If the port is not currently in Listen Only Mode, a normal response is returned. This occurs before the restart is executed.

When the remote device receives the request, it attempts a restart and executes its power–up confidence tests. Successful completion of the tests will bring the port online. A request data field contents of FF 00 hex causes the port's Communications Event Log to be cleared also. Contents of 00 00 leave the log as it was prior to the restart.

Sub-function Data Field (Request) Data Field (Response)
00 01 00 00 Echo Request Data
00 01 FF 00 Echo Request Data

2.1.8.3 02 Return Diagnostic Register

The contents of the remote device's 16-bit diagnostic register are returned in the response. Currently the diagnostic register is unused on the iPac and will always return 0 when read.

Sub-function Data Field (Request) Data Field (Response)
00 02 00 00 Diagnostic Register Contents

2.1.8.4 03 Change ASCII Input Delimiter

The character 'CHAR' passed in the request data field becomes the end of message delimiter for future messages (replacing the default LF character). This function is useful in cases of a Line Feed is not required at the end of ASCII messages. The new delimiter is saved in volatile memory only, and will revert to its default "LF" upon restart.

Sub-function Data Field (Request) Data Field (Response)
00 03 CHAR 00 Echo Request Data

2.1.8.5 04 Force Listen Only Mode

Forces the addressed remote device to its Listen Only Mode for MODBUS communications. This isolates it from the other devices on the network, allowing them to continue communicating without interruption from the addressed remote device. No response is returned.

When the remote device enters its Listen Only Mode, all active communication controls are turned off. The Ready watchdog timer is allowed to expire, locking the controls off. While the

device is in this mode, any MODBUS messages addressed to it or broadcast are monitored, but no actions will be taken and no responses will be sent. The only function that will be processed after the mode is entered will be the Restart Communications Option function (function code 8, sub-function 1).

Sub-function Data Field (Request) Data Field (Response)
00 04 00 00 No Response Returned

2.1.8.6 10 (0A Hex) Clear Counters and Diagnostic Register

The goal is to clear all counters and the diagnostic register. Counters are also cleared upon power–up.

Sub-function Data Field (Request) Data Field (Response)
00 0A 00 00 Echo Request Data

2.1.8.7 11 (0B Hex) Return Bus Message Count

The response data field returns the quantity of messages that the remote device has detected on the communications system since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 0B 00 00 Total Message Count

2.1.8.8 12 (0C Hex) Return Bus Communication Error Count

The response data field returns the quantity of CRC errors encountered by the remote device since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 0C 00 00 CRC Error Count

2.1.8.9 13 (0D Hex) Return Bus Exception Error Count

The response data field returns the quantity of MODBUS exception responses returned by the remote device since its last restart, clear counters operation, or power–up. Exception responses are described and listed in chapter 7.

Sub-function Data Field (Request) Data Field (Response)
00 0D 00 00 Exception Error Count

2.1.8.10 14 (0E Hex) Return Slave Message Count

The response data field returns the quantity of messages addressed to the remote device, or broadcast, that the remote device has processed since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 0E 00 00 Slave Message Count

2.1.8.11 15 (0F Hex) Return Slave No Response Count

The response data field returns the quantity of messages addressed to the remote device for which it has returned no response (neither a normal response nor an exception response), since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 0F 00 00 Slave No Response Count

2.1.8.12 16 (10 Hex) Return Slave NAK Count

The response data field returns the quantity of messages addressed to the remote device for which it returned a Negative Acknowledge (NAK) exception response, since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 10 00 00 Slave NAK Count

2.1.8.13 17 (11 Hex) Return Slave Busy Count

The response data field returns the quantity of messages addressed to the remote device for which it returned a Slave Device Busy exception response, since its last restart, clear counters operation, or power–up.

Sub-function Data Field (Request) Data Field (Response)
00 11 00 00 Slave Device Busy Count

2.1.8.14 18 (12 Hex) Return Bus Character Overrun Count

The response data field returns the quantity of messages addressed to the remote device that it could not handle due to a character overrun condition, since its last restart, clear counters operation, or power–up. A character overrun is caused by data characters arriving at the port faster than they can be stored, or by the loss of a character due to a hardware malfunction.

Sub-function Data Field (Request) Data Field (Response)
00 12 00 00 Slave Character Overrun Count

2.1.8.15 20 (14 Hex) Clear Overrun Counter and Flag

Clears the overrun error counter and reset the error flag.

Sub-function Data Field (Request) Data Field (Response)
00 14 00 00 Echo Request Data

2.1.9 16 (0x10) Write Multiple registers

This function code is used to write a block of contiguous registers (1 to approx. 120 registers) in a remote device. The requested written values are specified in the request data field. Data is packed as two bytes per register. The normal response returns the function code, starting address, and quantity of registers written.

Request
Function code 1 Byte **0x10**
Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of Registers 2 Bytes 0x0001 to 0x0078
Byte Count 1 Byte 2 x **N***
Registers Value **N*** x 2 Bytes value
***N** = Quantity of Registers

Response
Function code 1 Byte **0x10**
Starting Address 2 Bytes 0x0000 to 0xFFFF
Quantity of Registers 2 Bytes 1 to 123 (0x7B)

Error
Error code 1 Byte **0x90**
Exception code 1 Byte 01 or 02 or 03 or 04

2.1.10 22 (0x16) Mask Write Register

This function code is used to modify the contents of a specified holding 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 holding 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. Therefore registers 1-16 are addressed as 0-15. The function's algorithm is:

Result = (Current Contents AND And_Mask) OR (Or_Mask AND And_Mask)

For Example	**Hex**	**Binary**
Current Contents	= 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 contents of the register can be read with the Read Holding Registers function (function code 03). They could, however, be changed subsequently as the controller scans its user logic program. The normal response is an echo of the request. The response is returned after the register has been written.

Request
Function code 1 Byte **0x16**
Reference Address 2 Bytes 0x0000 to 0xFFFF
And_Mask 2 Bytes 0x0000 to 0xFFFF
Or_Mask 2 Bytes 0x0000 to 0xFFFF

Response
Function code 1 Byte **0x16**
Reference Address 2 Bytes 0x0000 to 0xFFFF
And_Mask 2 Bytes 0x0000 to 0xFFFF
Or_Mask 2 Bytes 0x0000 to 0xFFFF

Error
Error code 1 Byte **0x96**
Exception code 1 Byte 01 or 02 or 03 or 04

3. REGISTERS

Control of the iPac is done by reading and writing its registers. The full memory map can be seen below in *Table 2 iPac REGISTERS* . Registers are accessed via the Modbus functions described in the above section *FUNCTION DESCRIPTIONS* . Some registers will return different values depending on the function used to read them, and some registers are only accessible with certain functions, but the memory map is the same for all the functions.

For example, address 0 contains the digital port 0 register. If this register is read using the read input register function it will return a bitmap of the current state of the ports lines. On the other hand, if the same register is read using the read holding register function, it will return a bitmap of the data latched for output on that port. Therefore if the port is set to be an output port the register will contain identical data regardless of the function used to read it, but if it's set for an input port the two functions could return entirely different readings.'

All Modbus registers by default are 16 bits. On the iPac, many of the registers only require 8 bits or less. The USED BITS column of table 2 illustrates this. Zeros are unused bits.

3.1 NV Mirror registers

The bottom half of the register table is made up of NV register mirrors. These mirrors can be used to store power up defaults for many of the configuration registers. Reading these registers as input registers will return the same value as their counterparts in upper memory, and modifying one will modify the other. However, if a mirror register is written to, its value is stored in non-volatile memory and every time the board resets this value will be loaded into the register. If the mirrors are read as holding registers the EEPROM is read, so the defaults can be checked.

Table 2 iPac REGISTERS

ADDR	REGISTER	DESCRIPTION	READ/WRITE	USED BITS
0	Dig0/DigH0	digital port 0	R/W	00000000xxxxxxxx
1	Dig1/DigH1	digital port 1	R/W	00000000xxxxxxxx
2	Dig2/DigH2	digital port 2	R/W	00000000xxxxxxxx
3	Dig3/DigH3	digital port 3	R/W	00000000xxxxxxxx
4	Dig4/DigH4	digital port 4	R/W	00000000xxxxxxxx
5	Dig5/DigH5	digital port 5	R/W	00000000xxxxxxxx
6	Digcfg0	digital config 0	R/W	00000000xxxxxxxx
7	Digcfg1	digital config 1	R/W	00000000xxxxxxxx
8	Digcfg2	digital config 2	R/W	00000000xxxxxxxx
9	Digcfg3	digital config 3	R/W	00000000xxxxxxxx
10	Digcfg4	digital config 4	R/W	00000000xxxxxxxx
11	Digcfg5	digital config 5	R/W	00000000xxxxxxxx
12	Count0	counter 0	R/W	xxxxxxxxxxxxxxxx
13	Count1	counter 1	R/W	xxxxxxxxxxxxxxxx
14	Count2	counter 2	R/W	xxxxxxxxxxxxxxxx
15	Count3	counter 3	R/W	xxxxxxxxxxxxxxxx
16	Count4	counter 4	R/W	xxxxxxxxxxxxxxxx
17	Count5	counter 5	R/W	xxxxxxxxxxxxxxxx
18	Count6	counter 6	R/W	xxxxxxxxxxxxxxxx
19	Count7	counter 7	R/W	xxxxxxxxxxxxxxxx
20	CountCtrl0	count control0	R/W	00000000x00xxxxx
21	CountCtrl1	count control1	R/W	00000000x00xxxxx
22	CountCtrl2	count control2	R/W	00000000x00xxxxx
23	CountCtrl3	count control3	R/W	00000000x00xxxxx
24	CountCtrl4	count control4	R/W	00000000x00xxxxx
25	CountCtrl5	count control5	R/W	00000000x00xxxxx
26	CountCtrl6	count control6	R/W	00000000x00xxxxx
27	CountCtrl7	count control7	R/W	00000000x00xxxxx
28	CountR0	reload0	R/W	xxxxxxxxxxxxxxxx
29	CountR1	reload1	R/W	xxxxxxxxxxxxxxxx
30	CountR2	reload2	R/W	xxxxxxxxxxxxxxxx

ADDR	REGISTER	DESCRIPTION	READ/WRITE	USED BITS
31	CountR3	reload3	R/W	xxxxxxxxxxxxxxxx
32	CountR4	reload4	R/W	xxxxxxxxxxxxxxxx
33	CountR5	reload5	R/W	xxxxxxxxxxxxxxxx
34	CountR6	reload6	R/W	xxxxxxxxxxxxxxxx
35	CountR7	reload7	R/W	xxxxxxxxxxxxxxxx
36	Unused	N/A	R/W	0000000000000000
37	Unused	N/A	R/W	0000000000000000
38	AddressR1	Address	R/W	xxxxxxxxxxxxxxxx
39	ConfigR1	Configuration	R/W	00000000x00xx00x
40	DAC0	dig. to analog0	R/W	0000000xxxxxxxxx
41	DAC1	dig. to analog1	R/W	0000000xxxxxxxxx
42	AtoD0	AtoD port 0	RO	000000xxxxxxxxxx
43	AtoD1	AtoD port 1	RO	000000xxxxxxxxxx
44	AtoD2	AtoD port 2	RO	000000xxxxxxxxxx
45	AtoD3	AtoD port 3	RO	000000xxxxxxxxxx
46	AtoD4	AtoD port 4	RO	000000xxxxxxxxxx
47	AtoD5	AtoD port 5	RO	000000xxxxxxxxxx
48	AtoD6	AtoD port 6	RO	000000xxxxxxxxxx
49	AtoD7	AtoD port 7	RO	000000xxxxxxxxxx
50	AtoD8	AtoD port 8	RO	000000xxxxxxxxxx
51	AtoD9	AtoD port 9	RO	000000xxxxxxxxxx
52	AtoD10	AtoD port 10	RO	000000xxxxxxxxxx
53	AtoD11	AtoD port 11	RO	000000xxxxxxxxxx
54	AtoD12	AtoD port 12	RO	000000xxxxxxxxxx
55	AtoD13	AtoD port 13	RO	000000xxxxxxxxxx
56	AtoD14	AtoD port 14	RO	000000xxxxxxxxxx
57	AtoD15	AtoD port 15	RO	000000xxxxxxxxxx
Start of NV mirror				
58	Dig0NV	digital port 0	R/W	00000000xxxxxxxx
59	Dig1NV	digital port 1	R/W	00000000xxxxxxxx
60	Dig2NV	digital port 2	R/W	00000000xxxxxxxx
61	Dig3NV	digital port 3	R/W	00000000xxxxxxxx
62	Dig4NV	digital port 4	R/W	00000000xxxxxxxx
63	Dig5NV	digital port 5	R/W	00000000xxxxxxxx
64	Digcfg0NV	digital config 0	R/W	00000000xxxxxxxx
65	Digcfg1NV	digital config 1	R/W	00000000xxxxxxxx
66	Digcfg2NV	digital config 2	R/W	00000000xxxxxxxx
67	Digcfg3NV	digital config 3	R/W	00000000xxxxxxxx
68	Digcfg4NV	digital config 4	R/W	00000000xxxxxxxx
69	Digcfg5NV	digital config 5	R/W	00000000xxxxxxxx
70	CountCtrl0NV	count control0	R/W	00000000x00xxxxx
71	CountCtrl1NV	count control1	R/W	00000000x00xxxxx
72	CountCtrl2NV	count control2	R/W	00000000x00xxxxx
73	CountCtrl3NV	count control3	R/W	00000000x00xxxxx
74	CountCtrl4NV	count control4	R/W	00000000x00xxxxx
75	CountCtrl5NV	count control5	R/W	00000000x00xxxxx

ADDR	REGISTER	DESCRIPTION	READ/WRITE	USED BITS
76	CountCtrl6NV	count control6	R/W	00000000x00xxxxx
77	CountCtrl7NV	count control7	R/W	00000000x00xxxxx
78	CountR0NV	reload0	R/W	xxxxxxxxxxxxxxxx
79	CountR1NV	reload1	R/W	xxxxxxxxxxxxxxxx
80	CountR2NV	reload2	R/W	xxxxxxxxxxxxxxxx
81	CountR3NV	reload3	R/W	xxxxxxxxxxxxxxxx
82	CountR4NV	reload4	R/W	xxxxxxxxxxxxxxxx
83	CountR5NV	reload5	R/W	xxxxxxxxxxxxxxxx
84	CountR6NV	reload6	R/W	xxxxxxxxxxxxxxxx
85	CountR7NV	reload7	R/W	xxxxxxxxxxxxxxxx

3.2 Register Descriptions

3.2.1 Digital Ports

Dig0-5/DigH0-5
Address's 0-5
Mirrored – Yes
R/W

D7	D6	D5	D4	D3	D2	D1	D0

Digital port registers.
Dx **-** set or read the current state of bit x.

Reading these registers with the input register function (4) will return the current state of the port. Using the holding register function (3) will return the state of the latched output, regardless of the configuration of the port.

Writing to these registers will set the addressed port's latch to the value written. This value will drive the port if its configuration register bit is set to 1. The latch can be written even if the port is set to be an input, but it's value will not appear on the pins until the pins are configured as outputs.

Digcfg0-5
Address's 6-11
Mirrored - Yes
R/W

C7	C6	C5	C4	C3	C2	C1	C0

Digital port configuration registers.
Cx – configures a line as input or output
1 – bit x is configured as an output line.
0 – bit x is configured as an input line.

Writing to a digital configuration register configures the bits of the port to be input or output lines. For example, if Digcfg is set to 0xAA, the odd lines will be set to input, and the even lines to output.

Note: Digcfg1 is both a digital port and a counter. When a pin is enabled as a counter it is set to be an input, If Digcfg1 is used to change the pin back to an output the counter will be disabled until the counter register is re-written or the pin is changed back into an input. A NV counter value will take precedence over a NV digital port configuration.

Hardware Notes:
Digital ports 0-2 can be input or output. The higher digital ports, however, have some restrictions.

Port 3 – pins 0 and 1 are input only, pins 5,6 and 7 are output only.

Port 4 – Output only, high drive outputs

Port 5 – Input only

These restrictions are apparent in the software. For example, setting output only configuration bits will have no effect, the bit will not change.

4. COUNTERS

Count0-7
Address's 6-13
Mirrored - No
R/W

C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

4.1 Counter state registers

Reading these registers returns the current value of the counter. Setting the register sets the counter to that value.

CountCtrl0-7
Address 14-21
Mirrored - Yes
R/W

EN	*X*	*X*	*ARL*	*ASP*	*TR1*	*TR0*	*STR*

4.2 Counter Control registers

These registers are used to convert the lines of Port1 into counters. This will flip the addressed port1 configuration bit to 0 to be used as an input.

X - unused

EN – enable
1 – Counter is enabled
0 – Counter is disabled

ARL – auto reload
1 - Counter auto reloads the value in its reload register when it rolls over
0 - Counter rolls over to 0xffff

ASP – auto stop
1 – counter stops at zero
0 – counter rolls over after zero

TR1	TR0	Edge Trigger
0	0	The count is unaffected by edges on the line
0	1	Rising edges decrement the count by one
1	0	Falling edges decrement the count by one
1	1	Rising and falling edges decrement the count by one

STR – start
1 – the counter is active and counting edges
0 – the counter is stopped.

CountR0-7
Address's 22-29
Mirrored - Yes
R/W

C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

4.3 Counter Reload registers

These register hold the value that is auto reloaded into a counter if it's ARL bit is set. These registers hold the address of the Modbus port. Writing to this register permanently changes the address of the port; this value is stored in non-volatile memory and will remain even if power is removed from the board. If a board is accidentally set to an unknown address it can be restored with a broadcast write to the address register (make sure all other units on the network are in listen only mode before you do this).

ConfigR1
Address 33
Mirrored – No, always non-volatile
R/W

PRO	X	X	PAR	PT	X	X	BR

Default value = 0x17

4.4 Modbus Configuration registers

AddressR1
Address 32
Mirrored – No, always non-volatile
R/W

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Default value = 2

PRO – protocol
1 – ASCII protocol
0 – RTU protocol

PAR – Parity enable
1 – parity is enabled
0 – no parity is used

PT – Parity type
If Par is set this bit sets the type of parity.
1 – odd parity
0 – even parity

BR – Baud Rate
1 – 19200
0 – 9600

These registers hold the configuration of the Modbus port. Writing to this register permanently changes the configuration of the port; this value is stored in nonvolatile memory and will remain even if power is removed from the board.

4.5 Digital To Analog registers

DAC0-1
Address's 34&35
Mirrored - No
R/W

0	***0***	***0***	***0***	***0***	***0***	***0***		***D7***	***D6***	***D5***	***D4***	***D3***	***D2***	***D1***	***D0***

DACx registers zero and one contain the current voltage, measured in units of 10mv of DAC line x. The max is 500 (5V) and the min is zero (0V). These DACs have a granularity of 50mVs. The 10mv granularity of the register is for future software updates, which will support higher end DAC chips over the iPac's SPI port.

4.6 Read Only registers

4.6.1 Analog To Digital registers

AtoDx
Address's 36-43
Mirrored - No
RO

0	***0***	***0***	***0***	***0***	***A10***	***A9***	***A8***	***A7***	***A6***	***A5***	***A4***	***A3***	***A2***	***A1***	***A0***

The AtoDx registers hold the current reading of the 10 bit A/D channel x.

5. TIMING

Below are some typical timing values for Modbus communication with the iPac. These values were obtained based on a READ_MULTIPLE_REGISTERS request for 8 registers at 19200.

Table 3 PACKET TIMING

Protocol	Request Tx	Response Time	Response Tx	Total
ASCII packet	10ms	1.5ms	25ms	36.5ms
RTU packet	5ms	4ms	11ms	20ms

As Illustrated by the above table RTU protocol is a more efficient method of communication, having only a 20ms round trip time, as opposed to the ASCII packets 35.5ms. ASCII protocol, having twice as many characters to transmit will take close to twice as long. It uses a delimiter rather than a timeout, and therefore the packet can be interpreted and rebuilt in a smaller length of time, However, at 19200 Baud this is more than offset by the transmission time required sending the packet.

6. OTHER SOURCES OF INFORMATION

The iPac Modbus was designed to be 100% percent compliant with Modbus version 1.1. Information on Modbus 1.1, Modbus serial ports, and all other things Modbus can be found at www.modbus.org. This website also contains some useful links to Modbus freeware and Modbus for Linux.

7. QUESTIONS & FEEDBACK

Please address any questions or comments about the iPac Modbus protocol or it's documentation to support@emacinc.com with iPac in the subject line.