https://www.imoticondrives.co.uk/

Document number	Imoticon-001
Revision	0.1
Author	Gareth Lloyd
Product	Imoticon ID700

Title	Imoticon ID700 Modbus Serial Communications

Summary	This document gives information on the Imoticon ID700 Modbus serial communications

A serial communications link enables one or more drives to be used in a system controlled by a host controller such as a PLC or computer.

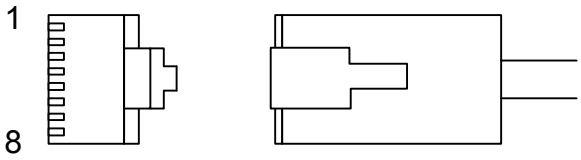
Communications Port & Wiring

Hardware interface of ID700 drive communication:
The Imoticon ID700 has 2 communication interfaces:

- RJ45 port
- Two rising clamp terminals (A/RS484+, B/RS485−)

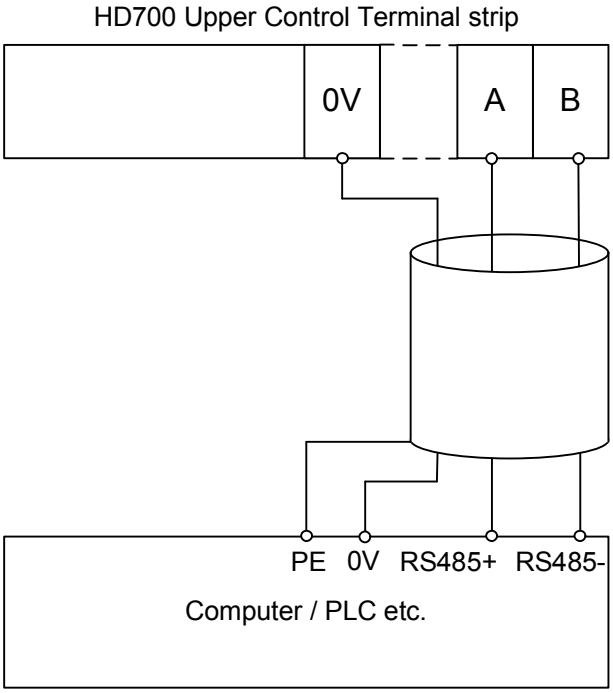
The two interfaces are connected in parallel so either can be used depending on the type of system wiring employed.

RJ45 Port Connections

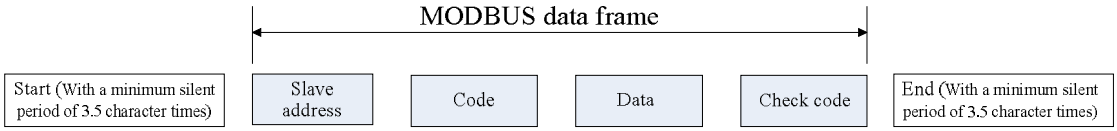


Pin number	Function
1	Not connected
2	A (RS485+)
3	0V
4	+24V
5	Not connected
6	Enable
7	B (RS485-)
8	B (RS485-)

Terminal connection



HD700 uses Modbus RTU. Modbus RTU supports read/write normal registers. The frame has the following basic format:



Modbus RTU uses byte type of "big-endian" to state address and data (except the CRC, which is "little-endian"), sends high byte firstly, then low byte. The frame is terminated with a minimum silent period of 3.5 character times at start and end. Use CRC-16 to check the message information.

Function codes

The function code determines the different requests:

Code (Hex)	Description
03H	Read multiple registers
06H	Write single register, not save when power off
10H	Write multiple registers, not save when power off
17H	Read and write multiple registers, not save when power off

Parameter mapping

The mapping rules between parameter number and register address as below:
Register address (hexadecimal): MNH
M= decimal convert to hexadecimal from "m"
N= decimal convert to hexadecimal from "n"
"m" and "n" calculation is as below, use a parameter Px.y as the example:
x.y*100=m*256+n+1
For example:
Modbus register address of parameter P04.01
4.01×100=401=1×256+144+1
Then
m=1, n=144
by the decimal to hexadecimal converting
M= 01, N=90
So, the Register address=0190H
Note: Modbus register addresses for all HD700 parameters can be found at the beginning of each parameter section of the HEDY HD700 Advanced User Guide V3.1

Function code example 1 (03H)

The example is to read the contents in P04.01～P04.10 of HD700 drive, details as below table:

Master Require							
Drive Node	Code	Start Register Address		Number Of Register Read		CRC Checking	
		MSB	LSB	MSB	LSB	LSB	MSB
01H	03H	01H	90H	00H	0AH	C4H	1CH

Slave (HD700 drive) Response									
Drive Node	Code	Number Of Register Read	Contents of P04.01～P04.10					Check Sum Of CRC	
			P04.01		……	P04.10		LSB	MSB
			MSB	LSB		MSB	LSB		
01H	03H	14H	01H	F4H	……	07H	D0H	B9H	76H

Function code example 2 (06H)

The example is to write 8 into P03.27.

Master Require							
Drive Node	Code	Register Address		Register Data		Check Sum Of CRC	
		MSB	LSB	MSB	LSB	LSB	MSB
01H	06H	01H	46H	00H	08H	68H	25H
Slave (HD700 drive) Response							
Drive Node	Code	Register Address		Register Data		Check Sum Of CRC	
		MSB	LSB	MSB	LSB	LSB	MSB
01H	06H	01H	46H	00H	08H	68H	25H

Abnormal communication

If the communication is abnormal, HD700 drive will turn back to the response frame, the format is in the below table:

Abnormal response format:

Drive node	code	Abnormal code	CRC checking sum	
1 bit	1 bit	1 bit	LSB	MSB

Abnormal code description

Code	description
81H	Not support the parameter
82H	Register address is beyond limit, the registers being read is too
83H	The content of register is over limit

CRC checking

CRC is 16 bit cycle redundance checking, normally the standard CRC-16 is called: x16+x15+x2+1.

Send the 16 bit CRC message to LSB, in a frame do the calculation of all bits.

```
const unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
```

Low Order Byte Table

```
const char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
```

```
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

/* CRC Generation for Modbus messages */
 // The function returns the CRC as a unsigned short type unsigned short CCRC_ModbusRTUCRC16
(unsigned char *puchMsg, short usDataLen )
{
unsigned short ReturnValue;
// high byte of CRC initialized
unsigned char uchCRCHi = 0xFF;
 // low byte of CRC initialized unsigned char uchCRCLo = 0xFF;
// will index into CRC lookup table unsigned char uIndex;
// pass through message buffer
while (usDataLen--) {
 // calculate the CRC
uIndex = uchCRCHi ^ *puchMsg++;
uchCRCHi = uchCRCLo ^ auchCRCHi[ uIndex ];
uchCRCLo = auchCRCLo[ uIndex ];
 }
ReturnValue = uchCRCHi;
ReturnValue <<= 8;
ReturnValue |= uchCRCLo;
return ReturnValue;
 }
```

HEDY HD700 Serial Communications Parameters

Parameter ID	Function	Range	Default	Change Mode	Modbus Address
P00.03 (P10.07)	Control mode	0: Keypad 1: Control terminal **2: Serial Communications**	0	Stop Only	0002H (03EEH)
P00.04 (P01.01)	Reference channel	0: Keypad 1: E-pot 2: Preset 3: AI1 4: AI2 **5: Serial Communications** 6: DI7 pulse 7: Fieldbus option 8: User programmed	0	Run & Stop	0003H (0064H)
P03.27	Comms. Control word	0～65535	0	Run & Stop	0146H
P03.28	Comms. Control word enable	0～1	0	Run & Stop	0147H
P10.02	address	0～247	1	Run & Stop	03E9H
P10.03	Baud rate	0: 2.4KBPS 1: 4.8KBPS 2: 9.6KBPS **3: 19.2KBPS** 4: 38.4KBPS 5: 57.6KBPS	3	Run & Stop	03EAH
P10.04	Communicat-ion configuration	0: 1-8-1, RTU, no checking **1: 1-8-2, RTU, no checking** 2: 1-8-1, RTU, odd checking 3: 1-8-1,RTU,even checking	1	Run & Stop	03EBH
P10.05	Response delay	0～250ms	2	Run & Stop	03ECH
P10.14	Drive status word	0～65535	N/A	Read only	03F5H

Parameters P03.27 and P03.28 provide a method of controlling the sequencer inputs and other functions directly from a single control word. If P03.28 = 0, the control word has no effect, if P03.28 = 1, the control word is enabled. Each bit of the control word corresponds to a sequencing bit or function as shown below:

Control Word Description (P03.27)

Bit	Decimal	Function
0	1	Drive enable (P03.19) (1 to disable, 0 to enable)
1	2	Run (P03.20)
2	4	Not Stop (P03.21)
3	8	Run forward (P03.22)
4	16	Run reverse (P03.23)
5	32	FWD/REV (P03.24)
6	64	Jog forward (P03.25)
7	128	Jog reverse (P03.26)
8	256	Trip reset (P12.15)
9	512	Save parameters
10	1024	Clear trip parameters
11	2048	Upadate floating logic
12	4096	Communications watchdog enable
13	N/A	Reserved
14	N/A	Reserved
15	N/A	Reserved

If P10.02 (drive address) = 0, drive will not respond to the master controller.

Drive Status Description (P10.14)

Bit	Decimal	Function
0	1	Drive healthy
1	2	Drive is active
2	4	100% load
3	8	At frequency
4	16	At zero speed
5	32	Running reverse
6	64	Current limit is working
7	128	Set time is met
8	256	Drive control changed to by control terminal
9	512	Under voltage
10	1024	Overload is calculating
11	2048	Alarm warning
12	4096	Length is met
13	8192	Counting is met
14	16384	PLC running is complate
15	32768	PLC is working

Scale definition
Frequency: 1:100
If the drive reference is 50.00Hz, then for comms it hex value is 1388H (5000 Decimal)

Time rate: 1:10
If the accelerating time is 10.0s, then for comms its Hex is 0064H (100 decimal)

Current rate: 1:10
If the current is 10.0A, then for comms its Hex is 0064H (100 decimal)

Voltage rate: 1:1
If voltage is 380V, then for comms its hex value is 017CH (380 decimal)

Examples using control word (P03.27):

Set **P03.28 = 1** (control word enable)
No terminal connections are required.

To 'enable' and 'run' the drive in the forward direction:
Send 2 to P03.28

To ramp down and 'stop' the drive from the above condition:
Send 4 to P03.28

To 'enable' and 'run' the drive in the 'reverse' direction:
Send 34 to P3.28

To 'coast to stop' the drive from the above condition:
Send 1 to P03.28

To 'enable' and 'run forward' the drive in the forward direction:
Send 8 to P03.28

To 'enable' and 'run reverse' the drive in the reverse direction:
Send 16 to P03.28

Speed reference
The speed reference is set in parameter P04.01 – preset speed 1

To run at 50Hz
Send 5000 to P04.01

To run at 40Hz
Send 4000 to P04.01
(Direction is controlled by the FWD/REV and Run Forward/Run Reverse bits of P03.27)