LadderWORK APPLICATION NOTES

Code : APP-LW-008
Title : MODBUS 8x8 estension
File Ref. : applw008.doc
Date : 30 September 2002
Author : MaxMT

REQUIREMENTS

- LadderWORK for 8051 2.x or greater

RELATED FILES

- modbus_exp8x8.pjn

MODBUS PROTOCOL OVERVIEW

The MODBUS protocol supports two modes of operations, ASCII and RTU (binary data). MicroSHADOW has selected to support only the RTU mode of transmission which allow better performance.

The MODBUS protocol specifies one master and up to 247 slaves on a common communication line, each slave is assigned a fixed unique device address in the range 1 to 247. The master always initiates a transaction. Transactions are either a query/response type (only a slave is accessed at a time) or a broadcast/no response type (all slaves are accessed at the same time). A transaction comprises a single query an single response frame or a single broadcast frame

Remote terminal unit (RTU) framing
Frame syncronization can be mainteined in RTU transmission mode only by simulating a syncronous message. The LadderWORK kernel monitors the elapsed time between receipt of characters. If 3.5 character times elapse without a new character or completion of the frame, then the frame is reset and the next bytes will be processed looking for a valid address.

Baud Rate	Byte Time Considering 10 bits (Start+8+Stop)	Byte Time * 3.5
300	33ms	0.12s
600	17ms	58ms
1200	8ms	29ms
2400	4ms	15ms
4800	2ms	7ms
9600	1ms	4ms
19200	0.5ms	2ms

Table 1 - RTU framing

Function codes
The function codes field tells the addressed device what function to perform
LadderWORK kernel integrates the following functions

- **Read boolean - Function Code 01**
- **Read numeric - Function Code 03**
- **Set single boolean - Function Code 05**
- **Set single numeric - Function Code 06**
- **Write multi boolean - Function Code 15**
- **Write multi numeric - Function Code 16**

Error responses
The high order bit of the function code field is set by the device to indicate that other than a normal response is be transmitted to the master computer. This bit remains zero (0) if the message is a query or normal response message.
Operation errors are those invoving illegal data in a message, or difficulty in communicating. These errors result in an exception response from either the master computer or the RTU device, depending upon the type of error. The exception response codes are listed in Table 2. When a slave detects one of these errors, it sends a response message to the master consisting of a slave address, function code, error code and error check fields. To indicate the response is a notification of an error, the high order bit of the function code is set to one (1).

Code	Exception	Description
01	Illegal function	The message function received is not an allowable action for the device
02	Illegal data address	The address referenced in the data field is not an allowable address for the device
03	Illegal data value	The value referenced in the data field is not allowable in the addressed location
04	Busy, message rejected	The message was received without error, but the device is engaged in processing a long duration command. Retransmit later, when the device may be free

Table 2 - Exceptions codes

Read Boolean (Function Code 01)

Boolean points are numbered as from 1001 (Boolean number 1 = 1001). The data is packed one bit for each Boolean flag variable. The response includes the slave address, function code, quantity of data characters, the data characters and error checking. Data will be packed with one bit for each boolean flag (1=ON, 0=OFF). The low order bit of the first character contains the addressed flag, and the reminder follow. For Boolean quantities that are not even multiples of eight, the last characters will be filled in with zeroes at high order end

Master to Slave

ADDRESS	FUNCTION	DATA START HIGH	DATA START LOW	NUMBER OF POINTS HIGH	NUMBER OF POINTS LOW	CRC CHECK 16 BIT
01	01	04	60	00	0C	XXXX

Slave to Master

ADDRESS	FUNCTION	BYTE COUNT	DATA BYTE # 1	DATA BYTE # 2	CRC CHECK 16 BIT
01	01	02	XX	XX	XXXX

Read Numeric (Function Code 03)

Function code 03 allows the MASTER to obtain the binary contents of holding registers in the addressed slave. The protocol allows for a maximum of 125 16 bit registers to be obtained at each request. Broadcast mode is not allowed for function 03.

These 16 bit registers are also grouped in sets of registers and accessed as one variable. The numeric range of the point number defines the variable type and indicates how many 16 bit registers make up that variable.

POINT# RANGE	VARIABLE TYPE	16 BIT REGISTER/POINT	# OF BYTES/POINT	MAX POINTS
3XXX or 13XXX	SHORT INTEGER	1 REGISTER	2 BYTES	125
4XXX	8CH ASCII STRING	4 REGISTERS	8 BYTES	31
5XXX or 15XXX	LONG INTEGER	2 REGISTERS	4 BYTES	62
7XXX or 17XXX	IEEE FLOATING POINT	2 REGISTERS	4 BYTES	62
14XXX	16CH ASCII STRING	8 REGISTERS	16 BYTES	15

Example : Read short integer 3012 through 3013 from slave # 2

ADDRESS	FUNCTION	STARTING HIGH	STARTING LOW	POINT# HIGH	POINT# LOW	CRC CHECK 16 BIT
02	03	0B	C4	00	02	XXXX

Set Single Boolean (Function Code 05)

This message forces a single boolean variable either ON or OFF. Boolean variables are points numbered 1XXX or 11XXX. Writing the 16 bit value 65,280 (FF00 HEX) will set the Boolean ON, writing the value zero will turn it OFF. All other values are illegal and will not affect the Boolean. Using a slave address 00 (Broadcast Address Mode) will force all slaves to modify the desired Boolean.

Example : Turn Single Boolean Point 1711 on Slave # 2

Master to Slave

ADDRESS	FUNCTION	BOOLEAN POINT HIGH	BOOLEAN POINT LOW	DATA HIGH	DATA LOW	CRC CHECK 16 BIT
02	05	06	AF	FF	00	XXXX

Slave to Master

ADDRESS	FUNCTION	BOOLEAN POINT HIGH	BOOLEAN POINT LOW	DATA HIGH	DATA LOW	CRC CHECK 16 BIT
02	05	06	AF	FF	00	XXXX

Set Single Numeric (Function Code 06)

Any numeric variable that has been defined on the 16 bit integer index table can have its contents changed by this message. The 16 bit integer points are numbered from 3XXX or 13XXX . When used with slave address zero (Broadcast Mode) all slaves will load the specified points with the contents specified. The following example sets 1 16 bit integer at address 3106 of slave number # 2.

Master to Slave

ADDRESS	FUNCTION	POINT# LOW	POINT# HIGH	DATA HIGH	DATA LOW	CRC CHECK 16 BIT
02	06	0C	22	00	03	XXXX

Slave to Master

ADDRESS	FUNCTION	POINT# LOW	POINT# HIGH	DATA HIGH	DATA LOW	CRC CHECK 16 BIT
02	06	0C	22	00	03	XXXX

Write Multi Boolean (Function Code 15)

Function code 15 (0FH) writes to each boolean variable in a consecutive block of boolean variables to a desired ON or OFF state. Each boolean is packed in the data field, one bit for each boolean flag (1 = ON , 0 = OFF). The data field consist of increments of 2 bytes and can be up tp 250 bytes (2000 points). Boolean points are packed right to left 8 to a byte with unused bits set to '0'. The use of slave address 00 (Broadcast mode) will force all slaves to modify the desired boolean bits. The following example writes to 14 boolean variables starting at address 1703. The data field, 05 (points 1703 through 1710) and 20 (points 1711 to 1716) are transmitted as 0000 0101 and 00100000, indicating that booleans points 1703, 1705, 1716 are to be forced ON and 1704 and 1706 through 1715 are to be forced OFF (the 2 most significant positions of the second byte are unused and set to 0).

Master to Slave

ADDRESS	FUNCTION	BOOLEAN POINT HIGH	BOOLEAN POINT LOW	POINT# HIGH	POINT# LOW	BYTE COUNT	DATA	DATA	CRC CHECK 16 BIT
03	0F	06	A7	00	0E	2	05	20	XXXX

Slave to Master

ADDRESS	FUNCTION	BOOLEAN POINT HIGH	BOOLEAN POINT LOW	DATA HIGH	DATA LOW	CRC CHECK 16 BIT
03	0F	06	A7	00	0E	XXXX

Write Multi Numeric (Function Code 16)

Function code 16 (10H) allows the master to change the binary contents of holding registers in the addressed slave. The protocol allows for a maximum of 125 16-bit registers to be changed at each download. Using an address of zero (00) allows the master to change registers in all slaves simultaneously (Broadcast mode)

These 16-bit registers are also grouped as sets of registers and accessed as one variable. The numeric range of the point number defines the variable type and indicates how many 16-bit registers make up that variable.

Point# Range	Variable type	16-bit regs. / Point	# of bytes	Max points for message
3xxx or 13xxx	Short integer (16 bits)	1 Register	2 bytes	125
5xxx or 15xxx	Double integer	2 registers	4 bytes	62

Master to Slave

ADR	FUNC.	STARTING POINT HIGH	STARTING POINT LOW	POINT# HIGH	POINT# LOW	BYTE COUNT	DATA	DATA	DATA	DATA	CRC CHECK 16 BIT
03	10	0B	C4	00	02	04	1F	40	1F	3E	XXXX

Slave to Master

ADDRESS	FUNCTION	BOOLEAN POINT HIGH	BOOLEAN POINT LOW	POINT# HIGH	POINT# LOW	CRC CHECK 16 BIT
03	10	0B	C4	00	02	XXXX

Current limitations in LadderWORK's MODBUS kernel

The current limitations must be keeped in consideration when using MODBUS kernel in LadderWORK

- The slave address is fixed to the value 3
- The broadcast mode isn't supported (Address field = 00)
- The protocol can transfer data objects of 1-bit (booleans), 16-bits (words) and 32-bits (double words)
- The CRC field is not handled during receiving and 0000H is always returned during transmission

Mapping of points in LadderWORK kernel

Range	Type	Descriptions
1000 - 1899	BOOL	Booleas mapped in static tables can be accesses within this range
1900 - 1998	BOOL	LadderWORK uses this range to map the watching variables
1999	BOOL	*** RESERVED POINT ***
3000 - 3899	WORD	Words mapped in static tables can be accesses within this range
3900 - 3999	WORD	LadderWORK uses this range to map the watching variables
5000 - 5899	DWORD	Double words mapped in static tables can be accesses within this range
5900 - 5999	DWORD	LadderWORK uses this range to map the watching variables
11000 - 11999	BOOL	I/O resource points can be accessed as booleans within this range
13000 - 13999	WORD	I/O resource points can be accessed as words within this range
15000 - 15999	DWORD	I/O resource points can be accessed as double words within this range

8 OUTPUTS 8 INPUTS EXTENSION WITH MODBUS PROTOCOL

Since release 2.x LadderWORK incorporates a complete MODBUS slave RTU kernel. Using this feature we show how to build a simple 8 INPUTs and 8 OUTPUTs I/O extension that is compatible with this popular automation protocol.
The schematic refers to the project file named **modbus_exp8x8.pjn** which is included into this application note.

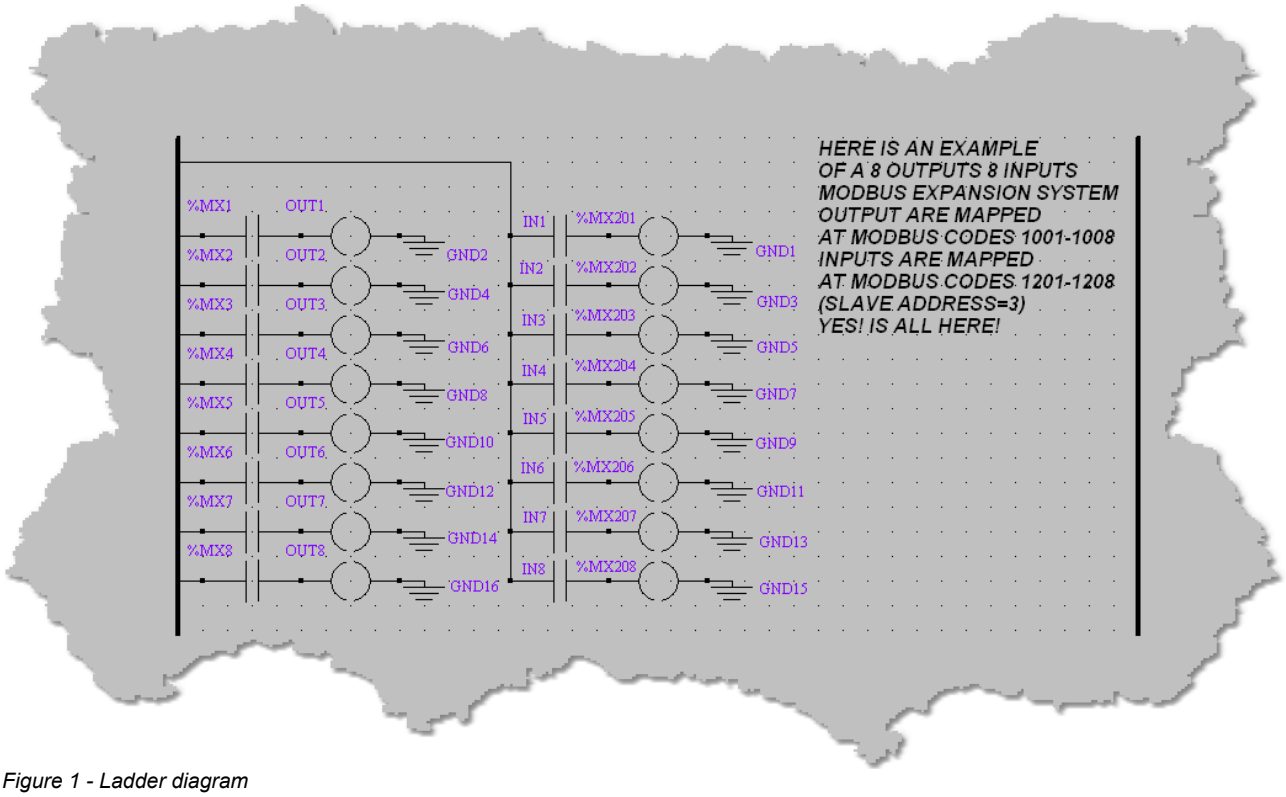

Figure 1 - Ladder diagram

Looking at the ladder diagram of Figure 1 we can evidence the two main sections. The section at the left (%MX1-%MX8 , OUT1-OUT8) realize the MODBUS driveable blocks. Through the protocol you can access the boolean variables %MX1-%MX8 which boolean state is directly transmitted to a PLC phisical output OUT1-OUT7.
In the same way the PLC inputs named IN1-IN8 directly drive the variables %MX201-%MX208 which can be readed always using this protocol.

OUT1 - OUT8 can be setted ON or OFF using the MODBUS function code 05 (Set single boolean) specifing an address in the range 1001 - 1008

The value present at inputs IN1 - IN8 are transferred to MODBUS variables 1201 - 1208 and are accessible using the function 01 (Read boolean) within the range 1201 - 1208

Quick start using MODBUS in a generic 8051 core

Follow the indicated steps to create a new project that embed the MODBUS kernel

- Create your design
- In LadderWORK's IDE activate the flag named "**Uncoditionally include MODBUS RTU kernel**" present in the dialog **Options Compiler** under the tab **MODBUS Kernel**
- Since, for default, the software uses the standard UART inside the 8051 core, you have to free the standard RX and TX pins that are normally routed to port 3. This can be done using the "**Advance Settings**" dialog that is accessed pressing the button present at the bottom-left corner of the **System Edit** dialog. Uncheck the box named "**Enable capture and refresh for port P3**" to avoid the using of port P3 in the ladder cycle.