USC FIELDBUS, MODBUS and TALKONLY

Introduction

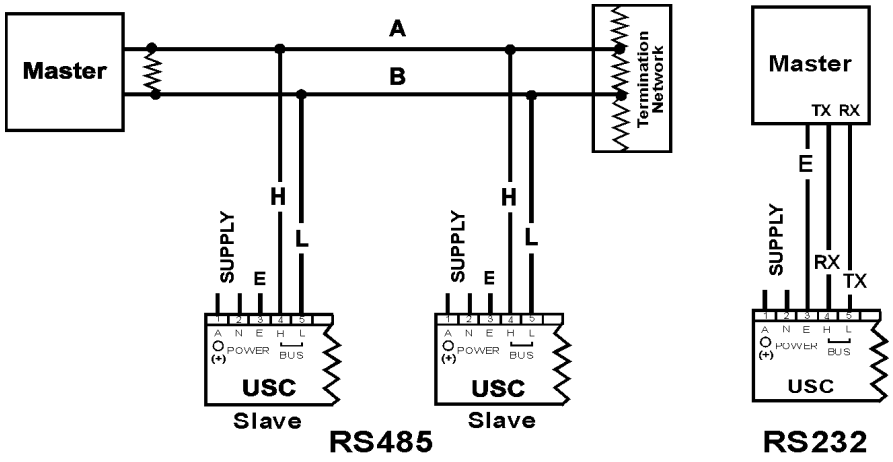
Protocol options: The same field bus card supports MODBUS RTU/ASCII and talk only protocols

Physical Layer: The output driver is factory set to RS485 or RS232. All protocols will work with either driver.

When configured for MODBUS the USC701 is a MODBUS client. A MODBUS master is required to request data from the USC701 or can optionally change data in the USC701. Example applications written with Visual Basic (with source code) will shortly be available on the APCS web site to demonstrate the capabilities of the USC as a MODBUS client and help connect the USC701 to commercial MODBUS master products.
The Modbus protocol provides the passing of messages during communications and determines how each USC701 will recognize a message addressed to it, determine the kind of action to be taken, and extract any data or other information contained in the message.

When configured as TALKONLY, predefined memory locations are sent at a predefined interval in ASCII format (no binary data is sent). This data can be collected directly on a line printer, terminal program or some form of data collection program. TALKONLY mode is one-way communication only.

Connections



TALKONLY mode (ASCII Output)

Communication parameters:

The options for the communications are as follows.

baud rate:	2400, 4800, 9600, 19200 or 38400.
parity:	none, even or odd.
stop bits:	1 or 2.
data bits:	8.
output channel:	can be any memory location from 0…7. (The USC has to be setup to send the required data automatically [either in the general purpose formula or by enabling the related input for memory locations 0…3])
Transmit time:	the time between messages can be set from 0.5...1000 seconds.
output format:	single or multiple channel format.

If there is only one channel being sent then the reading is sent and a carriage return, linefeed pair terminates the message

The output is in scientific notation SD.DDDDDDeSDD <CR><LF>

S	positive or negative sign
D	numeric digit
e	exponent
<CR>	carriage return
<LF>	line feed

If multiple readings are sent then commas and a space separate them. Where the first reading refers to the lowest number channel being sent and the last reading refers to the highest number channel being sent.

SD.DDDDDDeSDD, ……. , ……. , …… <CR><LF>
There is no error checking in this mode.

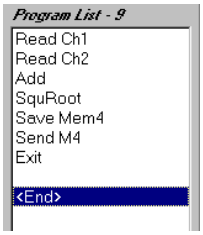
This is a “TALK ONLY” mode of the USC and as such will not respond to any communications it receives.

TALK ONLY Demonstration

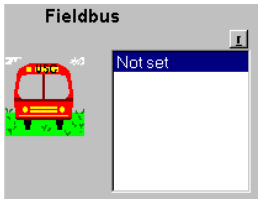
Example	Send the values measured by CH1, CH2 and Mem4 to the windows terminal application every 5seconds where $M4 = \sqrt{CH1 + CH2}$.

⇒ *First program the application into the USC701*

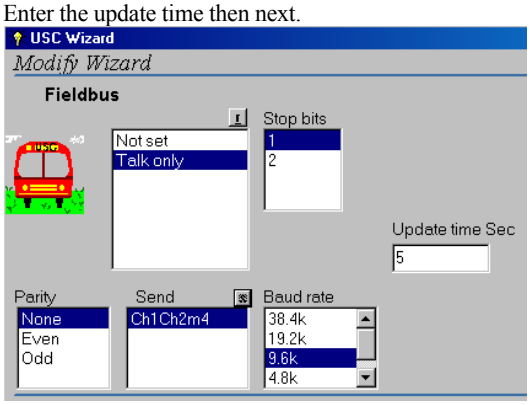
1. Start "USC Config 106" and run the 'New Module Wizard'.
2. Select Type= dc signal, Range=Adder wizard then next.
3. Select range for channel 1 (10Vdc) then next.
4. Select range for channel 2 (10Vdc) then next.
5. Select range for output 2 (20Vdc) then next.
6. Select next until wizard closes.
7. On the properties screen select the 'Equation' button then press 'Configure'
8. On the program list select the line that reads 'Save Mem4' and select $\sqrt{X}$ from the keyboard. The final equation is listed.



9. Press OK to save and exit the equation editor.
10. On the properties screen select the 'Fieldbus' button then press 'Configure'.
11. Select the small 'r' button to add the required range.
12. Select Type= 'Serial Talk Only' and Range='Serial Talk Only' then next.
13. Select 'Talk only in the main list box then fill in the required COM port parameters.



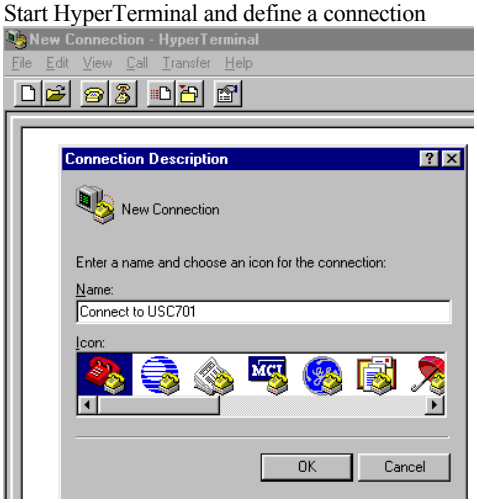
14. Select the small 's' button to add the required channels to be sent (Ch1,Ch2,m4).
15. Enter the update time then next.



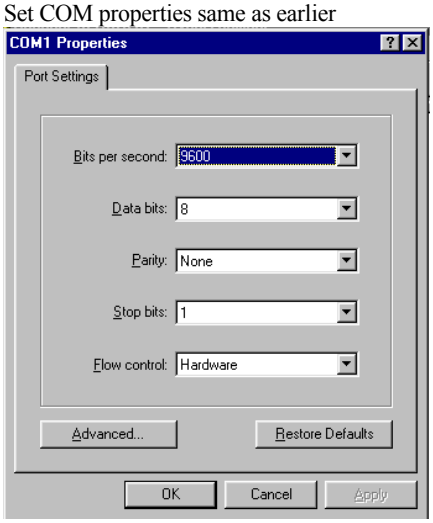
16. Return to the properties screen and press program.
17. After the USC701 is programmed disconnect power, wait 2 seconds and reconnect power. This step is necessary for the new communication parameters to take effect in the field bus card.

⇒ *Second connect Windows Terminal program.*

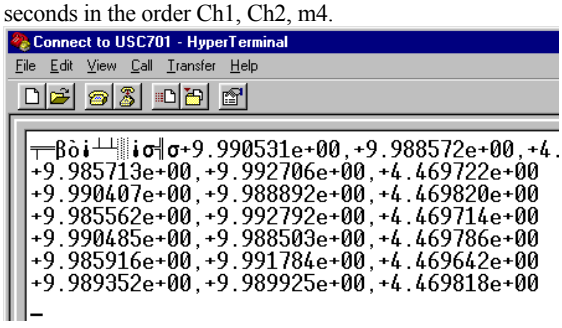
18. Start HyperTerminal and define a connection



19. Set COM properties same as earlier



20. Open the connection, new readings are added every 5 seconds in the order Ch1, Ch2, m4.



MODBUS RTU, MODBUS-ASCII

Function Codes

The function code is used in a query addressed to a slave USC701 device to determine an action to perform. Refer to section "Excerpt From Modbus Protocol" for more information.

Function Code	Description
02	read input status. (dummy response, for compatibility only).
03	read holding registers.
04	read input registers.
06	preset single register.
08	diagnostic. sub function 0000: return query data. sub function 0002: return diagnostic register.
10 (hex)	preset multiple registers.

Memory Map

In addition to an address and function code each query includes a MODBUS **register** to write or read data. The table below gives the register number and the format of the data at that location. Refer to section "USC701 Data Types" for more information on the data formatting used.

Key to data type: `S16I` = signed 16 bit integer
`S32I` = signed 32 bit integer
`FLOAT` = floating point number
`U8I` = unsigned 8 bit integer,
`8ASC` = 8 ASCII characters
`32ASC` = 32ASCII characters.

Location (hex)	Register	Data Type	Content.
0	1	S16I	Analogue channel CH1
1	2	S16I	Analogue channel CH2
2	3	S16I	Digital channel P1
3	4	S16I	Digital channel P2
4	5	S16I	Equation memory M4
5	6	S16I	Equation memory M5
6	7	S16I	Equation memory M6
7	8	S16I	Equation memory M7
8, 9	9	FLOAT	Analogue channel CH1
A, B	11	FLOAT	Analogue channel CH2
C, D	13	FLOAT	Digital channel P1
E, F	15	FLOAT	Digital channel P2
10,11	17	FLOAT	Equation memory M4
12,13	19	FLOAT	Equation memory M5
14,15	21	FLOAT	Equation memory M6
16,17	23	FLOAT	Equation memory M7
100	257	S16I	Relay 1 high trip
101	258	S16I	Relay 1 low trip
102	259	S16I	Relay 2 high trip
103	260	S16I	Relay 2 low trip
104, 105	261	FLOAT	Relay 1 high trip
106, 107	263	FLOAT	Relay 1 low trip
108, 109	265	FLOAT	Relay 2 high trip
10A, 10B	267	FLOAT	Relay 2 low trip
10C, 10D	269	S32I	Relay 1 on delay
10E, 10F	271	S32I	Relay 1 off delay
110, 111	273	S32I	Relay 1 power on delay

Communication Parameters

The options for the communications are as follows.

Baud rate:	2400, 4800, 9600, 19200 or 38400.
Parity:	none, even or odd.
Stop bits:	1 or 2.
Data bits:	8.
Min reply delay:	programmable from 3.5 character lengths to 255mS in 1-millisecond lengths. The actual reply time may be longer depending on the request. (Modbus RTU only)
Time out:	programmable from 0.5 sec to 1000 seconds.

Location (hex)	Register	Data Type	Content.
112, 113	275	S32I	Relay 2 on delay
114, 115	277	S32I	Relay 2 off delay
116, 117	279	S32I	Relay 2 power on delay
200	513	U8I	Remote control byte
201	514	U8I	Digital input flags
202	515	U8I	Remote digital flags
203	516	U8I	Excitation voltage output
204	517	U8I	Digital input threshold
300…303	769	8ASC	USC701 identification
304…307	773	8ASC	Engineering units for CH1
308…30B	777	8ASC	Engineering units for CH2
30C…30F	781	8ASC	Engineering units for P1
310…313	785	8ASC	Engineering units for P2
314…317	789	8ASC	Engineering units for M4
318…31B	793	8ASC	Engineering units for M5
31C…31F	797	8ASC	Engineering units for M6
320…323	801	8ASC	Engineering units for M7
400…40F	1025	32ASC	Firmware version number

Remote Control Byte (Register 513)

Enables direct remote control of the analogue output and relays. When remote control for a relay is not activated the relay status is reflected in the state of bit 0, 1.

- bit 0: relay 1 status / control , 1=relay on.
- bit 1: relay 2 status / control, 1=relay on.
- bit 2:
- bit 3:
- bit 4: enable remote control of relay 1.
- bit 5: enable remote control of relay 2.
- bit 6:
- bit 7: enable remote control of analogue output.

Digital Input Flags (Register 514)

this byte can be used to monitor the state of the control inputs to the USC

- bit 0: ch1 relay power on delay , 1= the power on delay has not expired.
- bit 1: ch2 relay power on delay , 1= the power on delay has not expired.
- bit 2: digital input 1 is active.
- bit 3: digital input 2 is active.
- bit 4: hold flag, the hold input is active.
- bit 5: tare flag, the tare input is active.
- bit 6: reset flag, the reset input is active.
- bit 7:

Remote Digital Flags (Register 515)

This byte is used to simulate the activation of a control input.

- bit 0: simulate closure of the power on delay input.
- bit 1:
- bit 2: simulate digital input 1 is active.
- bit 3: simulate digital input 2 is active.
- bit 4: simulate closure of the hold input.
- bit 5: simulate closure of the tare input.
- bit 6: simulate closure of the reset input.
- bit 7:

Excitation Voltage Output (Register 516)

This sets the excitation voltage output can be varied from about 3V to 18V.
Only the least significant 8 bits are used the other bits are ignored.

Digital Input Threshold (Register 517)

This sets the threshold for the digital inputs, the threshold range is from 0…20V
Only the least significant 8 bits are used the other bits are ignored.

Relay Delays

These set the delays associated with the relays. The times available range up to 2275 seconds, one counter unit is equal to 1.06 microseconds.

For example 10 second delay = 10 / 1.06e-6 = 9436800.

Exception Responses.

If a query is made and the USC can't give a valid answer then an exception is returned. The possible values of an exception are as follows.

- 1: message too long or an unsupported function. i.e. the number of registers is too large or the function is not recognised.
- 2: illegal address. Generally unallocated addresses are ignored when writing to the USC. if reading from the USC unallocated addresses usually return the value zero.
- 7: USC is in configure mode or trying to write to an address greater than 1C00 (hex) or 7168 (dec).

Diagnostic Register.

The diagnostic register is bit mapped and the bits have the following meaning.

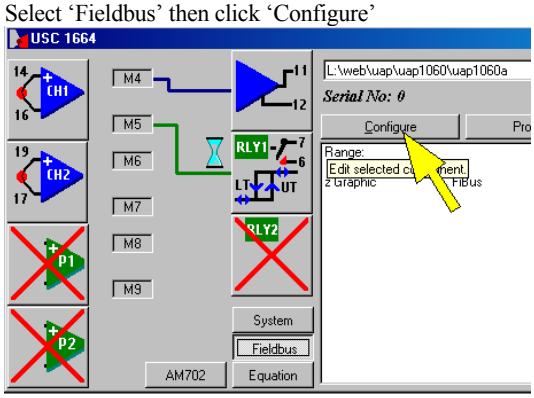
- bit 0: restricted access to USC.
- bit 1: Internal communication failure.
- bit 2….15: not used.

MODBUS RTU Demonstration

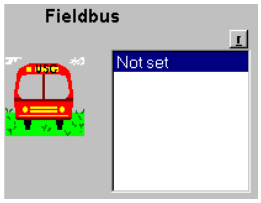
Two zones of a furnace are to be read, each zone has two thermocouples that should stay within a deviation temperature or an alarm is activated.
The higher temperature of the two thermocouples on each zone is retransmitted to a controller for each zone. Each USC701 will perform the required control function without supervision and will report temperatures relay state when queried by the MODBUS master.
The first USC701 is at MODBUS address 1
The second USC701 is at MODBUS address 2

Program Two USC701 Units For Use

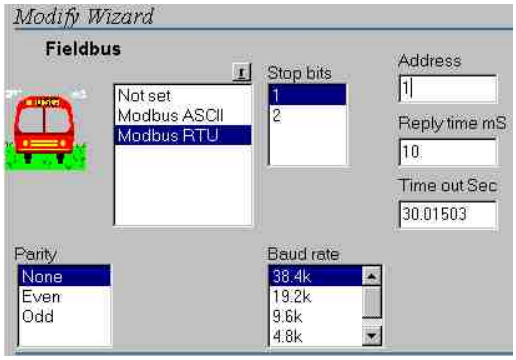
1. Start "USC Config 106" and import the pre-made application program UAP1060a from the APCS web site.
2. Select 'Fieldbus' then click 'Configure'



3. Select the small 'r' button to add the required range.
4. Select Type= 'Modbus' and Range='Modbus' then next.
5. Select 'Modbus RTU' in the man list box then fill in the required COM port parameters.



6. Select the required MODBUS address, 1 for first USC701 and 2 for the second.



7. Select 'Next' then 'Finish'.
8. Program the first USC701.
9. Change COA703 from the first USC701 to the second USC701.
10. Change the MODBUS address to 2.
11. Program the second USC701.
12. After the USC701s are programmed, disconnect power wait 2 seconds and reconnect power. This step is necessary for the new communication parameters to take affect in the field bus card.

♦ The two USC701 modules are now operating; data can now be obtained using MODBUS RTU.

Using MODBUS With USC701 Modules

There are commercial software packages available to read and write data to equipment connected to a MODBUS network. The purpose of this article is to demonstrate the use of the instruction set and memory map already documented on previous pages. The USC701 analogue inputs and analogue output are scaled using the 'USC Config 106' PC software. When reading to and from these devices is performed in the engineering units they are scaled to. To control the analogue output it should be allocated to one of the memories M4 to M7 and scaled to suit your application, then the field bus application writes to the allocated memory to control the output.

⇒ *The following examples use RTU*

Using Function 02

⇒ *Read input status*

Has no useful function with the USC701 except that many commercial packages poll the network on code 02 to determine if any hardware exists at each device address,

Using Functions 03 and 04

⇒ *03 read holding registers*
04 read input registers.

These functions are interchangeable on the USC701, both will return the same data.

Example: Read value from USC701 address 2 and CH1 as an integer value (Register1)

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Starting address Hi	00
Starting address Lo	00
Number of points Hi	00
Number of points Lo	01
Error Check CRC	84 39

Response	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Byte Count	02
Data	00 1E
Error Check CRC	7C 4C

The returned data '001E' = 30

Example: Read value from USC701 address 2 and CH1 as in IEE 32 bit floating point value (Register 9)

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Starting address Hi	00
Starting address Lo	08
Number of points Hi	00
Number of points Lo	02
Error Check CRC	45 FA

Response	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Byte Count	04
Data	41 F1 62 F9
Error Check CRC	65 DE

The returned data '41 F1 62 F9' = 30.1733265

⇒ *Using Functions 03 and 04 continued.*
All queries above register 23 require the encoder switch on top of the USC701 set to any position except 0

Example: Get the engineering units for USC701 address 2, CH1 (Register 773)

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Starting address Hi	03
Starting address Lo	04
Number of points Hi	00
Number of points Lo	04
Error Check CRC	05 BF

Response	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Byte Count	08
Data	44 65 67 20 43 20 20 20
Error Check CRC	2F D0

The returned data '44 65 67 20 43 20 20 20 2F D0' is plain text = Deg C

Example: Get CH1 to M4 from USC701 address 2 in IEE 32 bit floating point value (Register 9 to 23)

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Starting address Hi	00
Starting address Lo	08
Number of points Hi	00
Number of points Lo	10
Error Check CRC	C5 F7

Response	
Field Name	**Example (Hex)**
Slave address	02
Function	03
Byte Count	20
Data	42 02 23 D4 42 02 F3 39 00 00 00 00 00 00 00 00 42 02 F3 39 3E 4F 65 00 00 00 00 00 00 00 00 00
Error Check CRC	97 25

```
CH1 data '42 02 23 D4' = 32.535
CH2 data '42 02 F3 39' = 32.738
```

```
P1  data '00 00 00 00' = 00.000
P2  data '00 00 00 00' = 00.000
M4  data '42 02 F3 39' = 32.738
M5  data '3E 4F 65 00' = 00.203
M6  data '00 00 00 00' = 00.000
M7  data '00 00 00 00' = 00.000
```

Using Function 06

⇒ *Preset single register*
All queries above register 23 require the encoder switch on top of the USC701 set to any position except 0.
Function 06 writes data to the USC701 the encoder switch must also be to any position except 0 for all registers.

Useful for registers 513 to 517.

Example: The USC701 application is using CH1, CH2 and RLY1 as a deviation alarm.
Take control of RLY2 (Register 513)

To take control of RLY2 without affecting the operation of RLY1 and the analogue output bits 4 and 7 must remain at 0 while bit 5 is set to 1 and bit 1 is used to set RLY2 on and off (00100010 binary =

⇒ *Set RLY2 to ON*

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	06
Starting address Hi	02
Starting address Lo	00
Preset data Hi	00
Preset data Lo	22
Error Check CRC	08 58

Normal response is to echo the query

⇒ *Set RLY2 to OFF*

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	06
Starting address Hi	02
Starting address Lo	00
Preset data Hi	00
Preset data Lo	20
Error Check CRC	89 99

Normal response is to echo the query

Using Function 10

⇒ *Preset multiple registers.*

Useful for all registers except 513 to 517.

Example: Change the engineering unit of M4 to 'New Text' (Register 789)

This demonstrates writing plain text into the USC701

Query	
Field Name	**Example (Hex)**
Slave address	02
Function	10
Starting address Hi	03
Starting address Lo	14
Number of points Hi	00
Number of points Lo	04
Number of bytes	08
Data	4E 65 77 20 54 65 78 74
Error Check CRC	D9 00
Response	
Field Name	**Example (Hex)**
Slave address	02
Function	10
Starting address Hi	03
Starting address Lo	14
Number of points Hi	00
Number of points Lo	04
Error Check CRC	81 B9

Example: Change the upper trip point of RLY1 in module 1 (Register 257) to 655

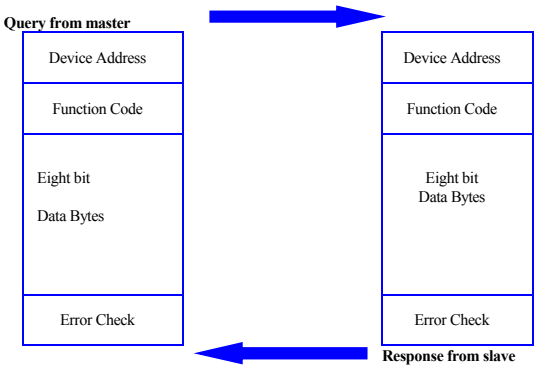
This demonstrates writing plain text into the USC701

Query	
Field Name	**Example (Hex)**
Slave address	01
Function	10
Starting address Hi	01
Starting address Lo	00
Number of points Hi	00
Number of points Lo	01
Number of bytes	02
Data	02 8F
Error Check CRC	F6 54
Response	
Field Name	**Example (Hex)**
Slave address	01
Function	10
Starting address Hi	01
Starting address Lo	00
Number of points Hi	00
Number of points Lo	01
Error Check CRC	00 35

Excerpt From Modbus Protocol

⇒ *http://www.modicon.com/TECHPUBS/intr7.html*

The master can address individual slaves. The Modbus protocol establishes the format for the master's query by placing into it the device address, a function code defining the requested action, any data to be sent, and an error-checking field. The slave's response contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.



The Query

The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response

If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

Two Serial Transmission Modes

Controllers can communicate on standard Modbus networks using either ASCII or RTU transmission modes along with the serial port parameters (baud rate, parity mode, etc), during configuration. The mode and serial parameters must be the same for all devices on a Modbus network.

When using **ASCII mode**, each eight-bit byte in a message is sent as two ASCII characters. The main advantage of this mode is that it allows time intervals of up to one second to occur between characters without causing an error.

When using **RTU mode**, each eight-bit byte in a message contains two four-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput.

Modbus Message Framing

In either mode a Modbus message is placed into a frame that has a known beginning and end. Partial messages can be detected and errors can be set as a result.

In **ASCII mode**, messages start with a colon (:) character and end with a carriage return-line feed pair. A typical message frame is shown below.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	# CHARS	2 CHARS	2 CHARS CRLF

In **RTU mode**, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network (shown as T1-T2-T3-T4 in the figure below). The entire message frame must be transmitted as a continuous stream.

The Error Checking Field

The error checking field contents depend upon the method that is being used.

In ASCII mode two ASCII characters are the result of a Longitudinal Redundancy Check (LRC) calculation that is performed on the message contents, exclusive of the beginning colon and terminating CRLF characters.

In RTU mode a 16-bit value implemented as two eight-bit bytes is the result of a Cyclical Redundancy Check calculation performed on the message contents.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
T21-T2-T3-T4	8 BITS	8 BITS	# x 7 BITS	16 BITS	T21-T2-T3-T4

USC701 Data Types

The USC variables and USC relay trip points are available in either 16-bit integer or IEEE32 floating point format. The variables are sent most significant byte first. The relay trip delays are 32 bit numbers and are sent most significant byte first, negative numbers for these are not valid.

IEEE32 floating point format

For single precision (32-bit-4 bytes) floating-point numbers, 1 bit determines the sign (+ or -), 8 bits determine the exponent (bias 127), and the remaining bits (23) determine the mantissa (or fraction) with implicit leading 1 before the fractional part.

$$V = 1.M * 2^{(E-127)}$$

V: value of number
E: exponent (range)
M: mantissa (precision)

Sign (1 bit)	Exponent (8 bits)	Mantissa (23 bits)
S	EEEEEEEE	MMMMMMMMMMMMMMMMMMMMMMM

Many MODBUS systems use a different byte order to the IEEE standard definition where the first bite read is read first.

For example ‘Citect’ uses the 1032 order, this can be changed in an ini file to 3210 order.