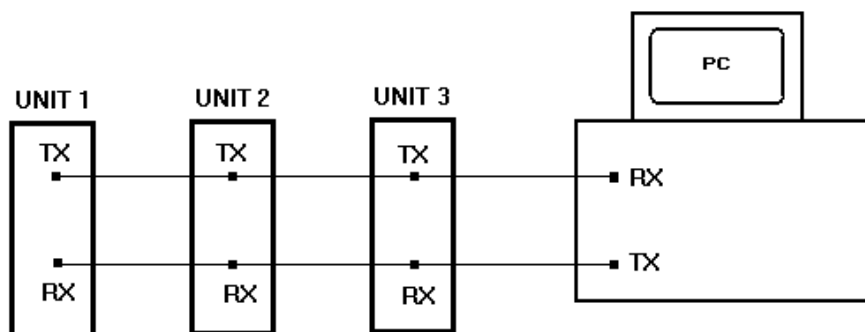


# APPENDIX E

## USER COMMUNICATION SOFTWARE

This section aims to provide sufficient information to enable a user to write software for a Personal Computer to interface directly with instruments on a network. As all configuration and runtime data are available via the comms, there is great potential to tailor a system to a users individual requirements.

Information for electrically connecting a network of units is dealt within the wiring section of this manual. This section explains the software interface and the basic comms operation of the instrument. The schematic of a typical network showing three instruments is shown below. Note that this is not wiring detail, only a schematic of signal interconnections.



You will notice that the transmit lines from the units are connected together. This means that only one unit can transmit at a time without clashing the signal. For this reason the communication software in the instrument(s) only responds to messages issued by the PC. No messages are generated spontaneously by the units, in this way the PC maintains control over the network. Thus the PC is regarded as the MASTER and the units on the network are SLAVES.

The other problem obvious from the above schematic is that even though the MASTER transmits to all of the SLAVES simultaneously, only one may respond, otherwise signals will clash together. This is arranged by allocating each SLAVE unit a unique address. This address is called the device number and is programmed into the unit before it is connected to the network by programming the CO NET parameters menu within the SYS menu. See the programming section for details.

The comms messages issued by the host can be one of two types.

**Data Request** - MASTER requests data from a SLAVE.

**Data Imposition** - MASTER *writes* data to a SLAVE.

The most convenient way to describe these message types is to show an example for each message type. It is not important to understand the full detail of the message at this stage as this will be covered later, however the following control code definitions will probably be useful.

<SOH>	Start Of Header
<STX>	Start Of Text
<ETX>	End Of Text
<ACK>	Acknowledge
<BCC>	Block Check Character

For the purposes of this example assume that the MASTER is connected to three SLAVE units as shown in the above schematic. The Device numbers for the units are 1,2 and 3 and device number 2 has an Thermocouple input measuring 79.8 degrees C and an Alarm Output.

**EXAMPLE 1 DATA REQUEST**

The MASTER requests the Process Variable from SLAVE device 2. The process is initiated by the MASTER sending the following message.

**02<STX>?CH000<ETX><BCC>**

In order to see what actual data is sent from the host, see the table below. Note that the data is in hexadecimal.

PC MASTER TO SLAVE DEVICE(S) DATA REQUEST											
SOH	"02"		STX	"?CH000"						ETX	BCC
01	30	32	02	3F	43	48	30	30	30	03	BC

All three units receive the message, although units 1 and 3 will disregard it as the 02 carried in the initial part of the message designates the message as being for device 2 only. Device 2, on receipt of the message recognises the 02 address as its own device number and examines the main part of the message to see what action to take. The main part of the message in this case is ?CH000. The question mark is the first character and is used to denote that the message is a data request. CH is a mnemonic representing CHannel value ( Process Variable ). A full list of mnemonics are available later on in this Appendix. The remainder of the message is called the index, and as there is only one Channel on this device it is superfluous and is set to zero. Having recognised a valid message, the instrument first acknowledges back to the host and then replies with the current Process Variable value. The format for the message is shown below, notice there is no device number embedded in the message-there is only one MASTER device.

SLAVE DEVICE TO MASTER REPLY										
ACK	SOH	STX	" 79.8"						ETX	BCC
06	01	02	20	20	37	39	2E	38	03	16

**EXAMPLE 2 DATA IMPOSITION**

For a second example, take the case of the MASTER issuing a message, this time to change an Alarm Setpoint value again directed at device 2.

The MASTER sends the following sequence.

PC MASTER TO SLAVE DEVICE(S) DATA IMPOSITION															
SOH	"02"		STX	!AS00180.0										ETX	BCC
01	30	32	02	21	41	53	30	30	31	38	30	2E	30	03	0C

The receipt of the message by device number 2 is exactly as in the previous example. This time, however, the ! indicates the message is a data imposition and applies to the Alarm Setpoint (AS). There are two alarms, the index 001 indicates that it applies the first Alarm ( Alarm 2 would have an index of 002). Device 2 updates the Alarm setpoint with 80.0 and then responds with an acknowledgement as shown below.

SLAVE DEVICE TO MASTER REPLY					
ACK	SOH	STX	ACK	ETX	BCC
06	01	02	06	03	06

There is one important point to understand here. The new Alarm Setpoint has been programmed into the device and will be used to control the Alarm operation. However, it has not been programmed into non-volatile memory within the instrument, so when power is removed, it will be lost and on powering the instrument again, the original setpoint value will be restored.

The next example shows how programmed data is stored to non volatile memory.

**EXAMPLE 3 ACTION**

Although, at the start of this section, it was stated that there are two types of message; a Data Request and a Data Imposition; there is strictly a third type. This message has the same format as a data imposition except no data is transferred and it has the effect of making the instrument do an action. In this case the action is to save all of the settings to non volatile memory.

The MASTER sends the following sequence.

PC MASTER TO SLAVE DEVICE(S) DATA STORE								
SOH	"02"		STX	"!ds"			ETX	BCC
01	30	32	02	21	41	53	03	17

Again device 2 receives the message which is initially treated as a Data imposition. The **ds** mnemonic is taken as an instruction to do a data store to non volatile memory. An index or data is superfluous and is not included. On doing a data store an acknowledge is issued as shown below.

SLAVE DEVICE TO MASTER REPLY					
ACK	SOH	STX	ACK	ETX	BCC
06	01	02	06	03	06

**THE CONTROL CODES**

Up to now the control codes have broadly been ignored, although the function of most of them is probably self-evident from the above examples. These will be explained in more detail here.

The control codes have two functions. First of all, they provide markers to indicate the start of the message and separate the different types of data with the message. And secondly, they provide integrity checking of the message.

The **<SOH>** Start of Header control code will always be the first character in the message. This indicates that the Header will follow and has a value of 01.

The **<STX>** Start of Text character indicates that the Header information has finished and the Text or main body of the message will follow. this has a value of 02.

The **<ETX>** End of Text character indicates the end of the end of the main part of the message and the Block check character will follow. The End of Text character has a value of 03.

The **<BCC>** has not got a fixed value like the other control codes as it is a calculated value based upon a modulo 256 sum of all non control code characters in the message. This is calculated for each message before it is transmitted and the receiver confirms that a repeat calculation of the message results in the same block check value. If there has been any corruption in any part of the message, the block check character and the recalculated value will not tie up. If an instrument receives a corrupted message, it is ignored. It is up to the writer of PC software to determine what the MASTER does in such a situation.

The Block check character is for the message in example 1 is calculated as follows:-

30  
32  
3F  
43  
48  
30  
30  
30

**Total 1BC**

Modulo 256 ( least significant byte ) **BC**

**EXAMPLE BASIC PROGRAM**

The following basic program provides an example of a simple communications interface to run on a PC. The program is coded to use COM1 and communicate with device 1, but these may be modified as required.

```

10 OPEN ``COM1:9600,N,8,1" AS #1: header$="01"
20 INPUT ``Enter text string ( 0 to quit)";text$:if text$="0" goto 90
50 gosub 100: PRINT#1,TX$
70 gosub 200: if left$(text$,1)="!" then mid$(text$,1,1)="?":goto 50
80 print ``RECEIVED[`;rx$;" ]": goto 20
90 end
100 REM Calculates block check character based on header and text strings
150 sum=0: lh=len(header$):lt=len(text$)
170 for i=1 to lh:sum=sum+asc(mid$(header$,i,1)):next
180 for i=1 to lt:sum=sum+asc(mid$(text$,i,1)):next:bcc=sum-(256*int(sum/256))
190 tx$=chr$(1)+header$+chr$(2)+text$+chr$(3)+chr$(bcc)
195 return
200 REM Reads in received data without checking block check character
240 rx$=""
245 ch$ = input$(1,#1):if ch$ <> chr$(1) goto 245
250 ch$ = input$(1,#1):if ch$ <> chr$(2) goto 250
260 ch$ = input$(1,#1):if ch$ = chr$(3) goto 270
265 rx$=rx$+ch$:goto 260
270 return
    
```

The following tables list the available comms mnemonics for the instrument.

**INSTRUMENT COMMS MNEMONICS**

DESCRIPTION	MNEM	INDEX	DATA	FORMAT
PROCESS VARIABLE IN ENGINEERING UNITS	CH	NA	VALUE	ENGINEERING UNITS
NO OF DECIMAL PLACES FOR ENG UNITS	DP	NA	OPTAIN	0,1,2,3
SENSOR INPUT TYPE	IT	NA	OPTION	VOLTS,RTD,CURRENT,T/C
VOLTAGE RANGE	RV	NA	OPTION	100mV,1V,1-5V,10V
CURRENT RANGE	RC	NA	OPTION	4-20,0-20,0-10mA
THERMOCOUPLE TYPE	ST	NA	OPTION	K,J,T,R,S,E,F,N,B
TEMPERATURE SENSOR BURN OUT	BO	NA	OPTION	HIGH, LOW
TEMPERATURE SENSING UNITS	UT	NA	OPTION	°C, °F
ENGINEERING UNITS HIGH RANGE	IH	NA	VALUE	REAL
ENGINEERING UNITS LOW RANGE	IL	NA	VALUE	REAL
INPUT CONDITIONING	IC	NA	OPTION	LINEAR,SQUARE-ROOT,USER-DEFINED
FILTER FACTOR TIME CONSTANT (SECONDS)	FF	NA	OPTION	OFF, 0.5,1,2, 4, 8, 16, 32
FILTER JUMP-OUT (PERCENTAGE)	FJ	NA	OPTION	NONE, 1, 5, 10
USER LINEARISATION ELECTRICAL UNITS	EX	1-13	VALUE	ELECTRICAL UNITS
USER LINEARISATION ENGINEERING UNITS	IY	1-13	VALUE	ENGINEERING UNITS
ALARM ACTION	AA	1,2,3,4	OPTAIN	OFF,LOW,HIGH,DEVIATION
ALARM LATCH ENABLE	AL	1,2,3,4	OPTAIN	FALSE,TRUE
ALARM SETPOINT	AS	1,2,3,4	VALUE	ENGINEERING UNITS
ALARM HYSTERESIS	AH	1,2,3,4	VALUE	PERCENTAGE OF ENGINEERING RANGE
ALARM DEVIATION	AD	1,2,3,4	VALUE	PERCENTAGE OF ENGINEERING RANGE
ALARM RELAY SENSE	As	1,2,3,4	OPTAIN	NON-INVERTED,INVERTED
ALARM DELAY (SECONDS)	Ad	1,2,3,4	OPTAIN	OFF,1,2,5,10,15,20
ALARM CONDITION	AC	1,2,3,4	OPTAIN	NO-ALARM,SENSED,DETECTED,LATCHED
CURRENT RTX SPAN OUTPUT	SI	1,3	OPTION	4-20,0-20mA
CURRENT RTX MODE OF OPERATION	TI	1,3	OPTION	RETRANSMISSION,PRESET
CURRENT RTX HIGH RANGE	HI	1,3	VALUE	ENGINEERING UNITS
CURRENT RTX LOW RANGE	LI	1,3	VALUE	ENGINEERING UNITS
CURRENT RTX PRESET VALUE	PI	1,3	VALUE	ENGINEERING UNITS
PROGRAMMABLE VOLTAGE OUTPUT	OV	1,3	OPTION	2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10, 12, 15, 20
AUTOCYCLE	Ac	NA	OPTION	ENABLED,DISABLED
PASSWORD CODE	PC	NA	VALUE	UNSIGNED INTEGER
PASSWORD LEVEL	PL	NA	OPTION	SETPOINT,INPUT,OUTPUT,SYSTEM,CALIB
USER OFFSET	UO	NA	VALUE	ENGINEERING UNITS
GENERAL SYSTEM STATUS REPORT	GS	NA	STRING	
SYSTEM CONFIGURATION REPORT	SY	NA	STRING	
STORE DATA TO NON-VOLATILE MEMORY	ds	NA	NODATA	

## DATA FORMAT DEFINITION

**NO DATA** No data actually transferred.

This type of message initiates an activity within the instrument rather than accessing data. For instance, sending the Text string **!ds** will cause the unit to store its scratch parameter data area to EEPROM. This must always be done after configuration parameters have been modified via the Comms. If this is not done the changes will be lost when the instrument is switched off.

**OPTION** An ASCII number corresponding to the position of an item within the list of parameters denoted in the associated FORMAT column of the Message table. The numbering starts from zero. E.G An Alarm action (AA) High Alarm will have a data field corresponding to 2.

**VALUE** These are ASCII numeric fields which are scaled and formatted according to the entry in the FORMAT column of the message table.

TYPE OF DATA	LOW LIMIT	HIGH LIMIT	FIELD WIDTH	DECIMAL PLACES
ENGINEERINGUNITS	-ENG HI - 7%	+ENG HI +7%	6	DP
REAL	-32000	64000	6	DP
ELECTRICALUNITS				
10 Volt range	-10	10	6	2
1-5 Volt range	0	5	6	3
1 Volt range	-1	+1	6	3
100mV range	-100mV	+100mV	6	1
4-20mA	0	20	6	2
0-20mA	0	20	6	2
0-10mA	0	10	6	2
% of Eng range	0.00	99.99	5	2
Unsigned integer	0	65535	5	0

**STRING** This is an individually formatted string, usually (but not always) read-only. See below for details.

### SY: System request

This reports upon the identity of the instrument. This may only be data requested the returned data format is as follows.

aabbbbb0ccde0000ff0000

- aa:** device type DM for Digital Meter
- bbbbbb:** Issue date of software
- cc:** Variant type; U=01, C=02; A=03
- d:** Contents of option slot 1;
  - Nothing fitted=0;
  - Single Relay board=2;
  - Dual Relay board=3;
  - Volt o/p=4;
  - RTX board=5
- e:** Contents of option slot 2; Format as above.
- ff:** Number of channels; 01 always