Model Electronic Railway Group

TECHNICAL BULLETIN G16/4

Issue 2

Gordon Hopkins [M328]

RPC Interface Specification

November 2001

Introduction

In order to provide reliable communications between a PC of any kind and a Model Railway layout, a strict communication protocol must be used. As far as is known, there are no ready defined standards which can be applied for this type of general purpose input/output communication. Interface standards such as RS232 and RS485 do not specify message format or content, only voltage levels, loadings and timing constraints.

The Interface Specification which follows has been devised to provide a means of achieving the requirement for the RPC System. Since the first issue of this document was published, developments have continued with the System, resulting in a number of observations and new designs which have shown the benefits to be gained from some minor adjustments and extensions to the original standard. The original design of Remote Panel Interface (RPI) is being replaced with a new version, the RPIC, which is functionally equivalent, but can be easily configured to operate in either RS232, RS485 or Universal Serial Bus (USB) modes by fitting the appropriate components.

One topic in particular had been found to cause considerable confusion in the original scheme, namely the numbering of output bits and bytes. The original definition was intended to provide maximum flexibility, with a variable message size, effectively only limited by the available RAM in the microcontroller on the RPI. This principle, coupled with the way the RPC Module output shift registers operate, resulted in a numbering scheme where adding any output modules to a stack at a later date necessitated renumbering of the whole output structure, as the microcontroller expanded its internal buffer size and data storage locations dynamically to accommodate data for the new modules. While this process could certainly be made to operate after some practical experience, it was wholly undesirable and time consuming to define.

However, it was eventually realised that since the number range of the single bit read/write Message Types was only 0 to 255, there was little point in allowing byte numbers above 32 (8 bits x 32 bytes = 256). Thus, by defining the maximum number of bytes per RPIC as 32, a fixed buffer size and position in the microcontroller memory can be allocated, making it a simple matter to send data to the output shift registers in reverse order. This is just what's needed to correct the original difficulty, and results in the output bytes numbering away from the RPIC as modules are plugged in, just like the input bytes always have - a much easier concept to grasp, and independent of the number of modules being used!

A further change to byte numbering is that in the revised scheme they count from zero, rather than one, as originally. This brings them into line with the bit numbering method and generally accepted binary counting principles. Thus, the first output module attached to the RPIC always contains byte zero (and incidentally, bit zero). In the case of an SRO4, this would comprise bytes 0, 1, 2 & 3 (bits 0 to 31 inclusive). Next plug in a DPR and this becomes byte 4 (bits 32 to 39), and so on… The same principle applies to input modules and their bit and byte numbers, as always.

Other additions have resulted from work on USB interfacing and difficulties encountered when using non-real time operating systems such as Windows.

This Technical Bulletin only covers the commands and messages available via the Serial Communication Interface as this is the area where the User has direct access. The precise operation of the Shift Register Module Interface is transparent to the User and is covered in the RPIC Technical Bulletin G16/5 Issue 3 dated November 2001.

'Conventional' Serial Communication Formats

The original 8051-based RPI exploited a very useful feature of its built-in serial port. This is the 9 bit 'multi-processor' mode, which uses the Parity bit of a received byte to determine whether the byte is to be interpreted as an address or data. On a multidrop (i.e. multi-panel) RS485 network, this can be arranged so that each RPI only responds to messages sent to its unique address, programmed by a series of switches on each board. Parity set to '1' (or Mark) indicates an address byte. Any subsequent bytes used in a message must have their Parity set to '0' (or Space) to indicate that they contain data. Many recently released Microchip PIC devices also incorporate this 9 bit feature, allowing the principle to be maintained on RS485 versions of the new RPIC. However, this feature is not needed for single point RS232 systems as no addressing is required.

[cont. over >>>]

In the early days of system development, great difficulties were encountered trying to implement the RS485 multidrop principle with the Windows operating system. The unpredictable timing of hardware operation under Windows made it virtually impossible to make the communication reliable, or work at anything like an acceptable speed. As a result, an alternative approach was devised, involving the use of a new 'intelligent' RS232 to RS485 buffer, the RSB, which replaces the earlier 'dumb' RSA which was basically a simple voltage level shifter. With this arrangement, the PC communicates to the multidrop layout indirectly via the RSB, using the more conventional RS232 standard of 9600 Baud, 8 bits, no parity. To maintain commonality between similar set-ups, RS232 versions of the RPI and RPIC are arranged to work with the same data format from the PC as that used in conjunction with an RSB.

From the PC's perspective, all serial messages it sends now start with a two byte 'header' of 'AA', '55' Hexadecimal. These values were chosen arbitrarily as a means of identifying the start of a message, should reception by the RPIC ever get out of step for any reason. In such a case, the RPIC will revert to looking for this specific two byte combination before accepting any more data. Clearly, there is a small chance the actual layout data might produce this combination, but further security is provided by the final byte in every message - the exclusive-or checksum. The checksum is calculated as each byte of a message arrives, and compared with the final byte to ensure data integrity. If it is incorrect, the whole message is ignored, and the routine reverts to waiting for another valid header. If correct, the message is interpreted by the RPIC using the Message Type number, which is embedded in the first byte after the header. Messages which prompt a reply cause the addressed RPIC to perform the task of acquiring the requested data, and then returning it as one or more bytes, plus an exclusive-or checksum. In many cases, this will simply appear to be a duplicate of the data byte, especially in Message Types giving 'one byte' replies. In general, checksum calculations only apply to the data content within a message. The 'house keeping' bytes, i.e. Headers, Address/Message and Message Lengths are not included. In an RS485 system the data direction is controlled by the RSB, whereas RS232 systems are bi-directional by default.

Details of the communication format are given below:

Serial Message Types

There are currently ten message types supported :

Type 0: Complete Layout Data (Input & Output)	Type 5: Single Bit Read (Input)
Type 1: Single Byte Update (Output)	Type 6: LCD Text (Output)
Type 2: Single Byte Update (Input)	Type 7: Direct Echo (Test Only)
Type 3: Single Bit Set (Output)	Type 9: DCC Control
Type 4: Single Bit Clear (Output)	Type F: USB Extensions

Types 0 to 5 are used for layout accessory control. Layout initialisation will generally use Type 0 messages, as this is the most efficient method to set all outputs to a known state. Each Module kit provides its own information on byte numbers and bit allocations. All that is required is to total up the output bytes and input bytes separately for the stack. *Note that there is no Type 8 or Types A - E (hexadecimal notation) as at Issue 2.*

Having initialised each RPIC with its settings, subsequent updates can be achieved by using either message Type 0 again, the single byte messages Types 1 and 2, or the single bit control messages Types 3,4 & 5. The choice of which message types to use will largely be determined by the philosophy of the PC software running the system. A certain amount of consideration needs to be given to topics such as signal wiring, as it would be logical to keep the control lines for a particular signal within an individual byte, allowing single byte updating.

Type 9 DCC Control messages comprise Speed, Direction and Programming commands intended for use with the RPC System's DCC facility, the RCS1 Command Station. This unit can simultaneously control a stack of RPC modules, plugged into it using the Remote Stack Extension method (see G16/24) by using the other Message Types.

Type F USB Extensions are additional commands, made necessary by the limited packet size available with USB 1.1 low speed devices. These can also be used by non-USB systems if any advantage is identified.
Other message types are reserved for future use.

Specific Message Details.

Type 0 messages transfer to the layout as many bytes of data as are specified in the message to the RPIC. An equivalent number of input bytes are then transmitted back to the PC. It should be noted that if the number of output bytes in use is not the same as the number of input bytes in use, the higher of the two figures should determine the Type 0 message length to ensure complete data transfer, even if this means 'padding out' with null data to equalise the numbers. The PC software must be ready to receive the returned data, otherwise it will be lost.

Type 1 messages set an individual Module output byte to a new value. The byte position and new data are included in the message. No reply is expected.

Type 2 messages read an individual Module input byte. The byte required is specified in the message, and the byte is transmitted back as per Type 0.

Type 3 messages set an individual Module output bit to '1'. The bit number is specified in the message. Note that certain types of module with output drivers will effectively invert the logic signal emerging from the shift register device. No reply is expected.

Type 4 messages clear an individual Module output bit to '0'. The bit number is specified in the message. Note that certain types of module with output drivers will effectively invert the logic signal emerging from the shift register device. No reply is expected.

Type 5 messages read an individual Module input bit. The bit required is specified in the message, and the bit value is transmitted back as per Type 0.

Type 6 messages contain text strings for the LCD and are sent directly to the display. Characters may include any of the normal ASCII alphanumeric codes as well as the many special characters normally available on this type of display. No reply is expected. Carriage Return (ASCII Character 13) may be used to move between lines on multi-line LCDs. Character numbers less than 13 are ignored. The cursor position is reset to the start (top left) for any new message.

Type 7 messages cause the transmitted data to be immediately sent back to the host PC without being sent to the Modules. This is intended to be used for test purposes only.

Type 9 messages are defined for use with RPI modules having DCC capabilities. A sub-Message Type structure is used to allow for the various different commands available with DCC such as Speed & Direction, Functions, Service Mode Programming etc.

Type F (15 decimal) messages provide a means of reading and writing a maximum of two byte values at a time to the interface using the limited packet size available with low speed USB 1.1. These can be any two bytes out of the 32, as the byte numbers are included for each, as well as the byte values. The data content of a Type F message is 4 bytes. To speed up this otherwise apparently slow process, a special facility has been provided to check the status of all input bytes in a single action to determine whether any changes have occurred since the last reading. In this case, a flag bit is provided for each of the 32 input bytes of a stack, which will be set (1) if a change has occurred, or clear (0) if no change has occurred. Thus the 4 bytes available in the data packet provide information about any change which occurs. For example, if bit 0 is set to '1', a change has occurred in byte 0, so this byte should be read in and investigated. Generally, if bit 'n' has changed, then go and read byte 'n'. Clearly, any software written to use these facilities must account for the possibility of multiple changes being indicated.

Data Link Format Summary.

EIA RS485 or RS232 standard, depending on Users choice of System.

Baud Rate	-	9600
Byte Format	-	1 Start Bit, 8 Data Bits, No Parity, 1 Stop Bit

Message Format Summaries

a) Message Type 0 (Variable length - All Bytes Out, All Bytes In)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	0 H, 0 D	Message Type 0
4	7..0	02 to 21 H, 2 to 33 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to FF H, 0 to 255 D	First Layout Output Control Data Byte
6	7..0	00 to FF H, 0 to 255 D	Second Layout Output Control Data Byte
↓	7..0	00 to FF H, 0 to 255 D	Next Layout Output Control Data Byte
N	7..0	00 to FF H, 0 to 255 D	Final Layout Output Control Data Byte
N+1	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Layout Output Control Data Bytes ONLY

RPIC Reply to PC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	00 to FF H, 0 to 255 D	First Layout Input Feedback Data Byte
2	7..0	00 to FF H, 0 to 255 D	Second Layout Input Feedback Data Byte
↓	7..0	00 to FF H, 0 to 255 D	Next Layout Input Feedback Data Byte
N	7..0	00 to FF H, 0 to 255 D	Final Layout Input Feedback Data Byte (same number as Output bytes)
N+1	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Layout Input Feedback Data Bytes

b) Message Type 1 (Change Single Output Byte value)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	1 H, 1 D	Message Type 1
4	7..0	03 H, 3 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to 1F H, 0 to 31 D	Number of Layout Output Control Data Byte to be changed
6	7..0	00 to FF H, 0 to 255 D	New value for this byte
7	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Bytes 5 & 6 ONLY

No Reply to PC expected

c) Message Type 2 (Read Single Input Byte value)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	2 H, 2 D	Message Type 2
4	7..0	02 H, 2 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to 1F H, 0 to 31 D	Number of Layout Input Feedback Data Byte to be read
6	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Byte 5 ONLY

RPIC Reply to PC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	00 to FF H, 0 to 255 D	Layout Input Feedback Data Byte
2	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Layout Input Feedback Data Byte

d) Message Type 3 (Set Single Output Bit value)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	3 H, 3 D	Message Type 3
4	7..0	02 H, 2 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to FF H, 0 to 255 D	Number of Layout Output Control Data Bit to be set
7	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Byte 5

No Reply to PC expected

e) Message Type 4 (Clear Single Output Bit value)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	4 H, 4 D	Message Type 4
4	7..0	02 H, 2 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to FF H, 0 to 255 D	Number of Layout Output Control Data Bit to be cleared
7	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Byte 5

No Reply to PC expected

f) Message Type 5 (Read Single Input Bit value)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	5 H, 5 D	Message Type 5
4	7..0	02 H, 2 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to FF H, 0 to 255 D	Number of Layout Input Feedback Data Bit to be read
6	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Byte 5 ONLY

RPIC Reply to PC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	00 to 01 H, 0 to 1 D	Layout Input Feedback Data Bit value
2	7..0	00 to 01 H, 0 to 1 D	Exclusive-OR checksum of Layout Input Feedback Data Bit value

g) Message Type 6 (LCD Text String - Max 32 characters)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	6 H, 6 D	Message Type 6
4	7..0	02 to 21 H, 2 to 33 D	Length of character string to follow + 1 to allow for checksum
5	7..0	0D to 7FH, 13 to 127D	ASCII value of first Text Character for display
↓	7..0	0D to 7FH, 13 to 127D	ASCII value of next Text Character for display
N	7..0	0D to 7FH, 13 to 127D	ASCII value of final Text Character for display
N+1	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Text Character Bytes ONLY

No Reply to PC expected

Note: ‘Carriage Return’ character 0D H (13 D) may be used to select the second line of two line LCD modules. Strings may embed the ‘0D’ at any appropriate point in their content. Characters before the ‘0D’ appear on the first line, characters after the ‘0D’ appear on the second line. ‘Line Feed’ character 0A H (10 D) has no effect, and will be ignored automatically, even if one is included as part a conventional ‘Line Feed’, ‘Carriage Return’ sequence.

h) Message Type 7 (Echo Test)

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	7 H, 7 D	Message Type 7
4	7..0	02 to 21 H, 2 to 33 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 to FF H, 0 to 255 D	First Data Byte
6	7..0	00 to FF H, 0 to 255 D	Second Data Byte
↓	7..0	00 to FF H, 0 to 255 D	Next Data Byte
N	7..0	00 to FF H, 0 to 255 D	Final Data Byte
N+1	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Bytes ONLY

Continued over >>>

i) Message Type 7 (Echo Test) continued

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to FF H, 0 to 255 D	Copy of First Data Byte
2	7..0	00 to FF H, 0 to 255 D	Copy of Second Data Byte
↓	7..0	00 to FF H, 0 to 255 D	Copy of next Data Byte
N	7..0	00 to FF H, 0 to 255 D	Copy of final Data Byte (same number as Output bytes)
N+1	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Bytes

j) Message Type 9 (DCC Operation)

Note that Sub Type 0 is used for Speed/Direction, Function and Advanced Consisting Controls due to the general similarity of their format. Only the data content is different.

i) Sub Type 0 Speed & Direction

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	9 H, 9 D	Message Type 9
4	7..0	07 H, 7 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 H, 0 D	DCC Sub Message Type 0
6	7..0	00 to 0F H, 0 to 15 D	DCC Controller Number (equivalent to Handset or Throttle number)
7	7..0	C0 to FF H, 192 to 255 D	Loco Address byte 1 (MS Byte) for 4 digit Advanced Addressing Mode
	7..0	00 to 7F H, 0 to 127 D	Loco Address for 2 digit Baseline Addressing mode
8	7..0	00 to FF H, 0 to 255 D	Loco Address byte 2 (LS Byte) for 4 digit Advanced Addressing Mode
	7..0	00 H, 0 D	Set to zero for 2 digit Baseline Addressing mode
9	7..0	3F H, 63 D	Speed byte 1 (MS Byte) for 128 Speed Step Mode (per RP 9.2.1)
	7..0	00 to 7F H, 0 to 127 D	Speed byte for 14/28 Speed Step Mode (per RP 9.2.1)
10	7..0	00 to FF H, 0 to 255 D	Speed byte 2 (LS Byte) for 128 Speed Step Mode (per RP 9.2.1)
	7..0	00 H, 0 D	Set to zero for 14/28 Speed Step Mode
11	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Bytes ONLY

No Reply to PC expected	

ii) Sub Type 0 Function Control

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	9 H, 9 D	Message Type 9
4	7..0	07 H, 7 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 H, 0 D	DCC Sub Message Type 0
6	7..0	00 to 0F H, 0 to 15 D	DCC Controller Number (equivalent to Handset or Throttle number)
7	7..0	C0 to FF H, 192 to 255 D	Loco Address byte 1 (MS Byte) for 4 digit Advanced Addressing Mode
	7..0	00 to 7F H, 0 to 127 D	Loco Address for 2 digit Baseline Addressing mode
8	7..0	00 to FF H, 0 to 255 D	Loco Address byte 2 (LS Byte) for 4 digit Advanced Addressing Mode
	7..0	0 H, 0 D	Set to zero for 2 digit Baseline Addressing mode
9	7..5	100 B	Function Group 1 (per RP 9.2.1)
	4	0 (off) or 1 (on)	F4 (depending on CV29 bit 1 setting, per RP 9.2.1)
	3	0 (off) or 1 (on)	F3
	2	0 (off) or 1 (on)	F2
	1	0 (off) or 1 (on)	F1
	0	0 (off) or 1 (on)	F0
	7..4	1011 B	Function Group 2 (per RP 9.2.1)
	3	0 (off) or 1 (on)	F8
	2	0 (off) or 1 (on)	F7
	1	0 (off) or 1 (on)	F6
	0	0 (off) or 1 (on)	F5
10	7..0	00 H, 0 D	Always set to zero
11	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 5 to 10 ONLY

No Reply to PC expected	

ii) Sub Type 0 Advanced Consisting Control

PC Command to RPIC			Functional Definition
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always 'AA' Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always '55' Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	9 H, 9 D	Message Type 9
4	7..0	07 H, 7 D	Length of data to follow + 1 to allow for checksum
5	7..0	00 H, 0 D	DCC Sub Message Type 0
6	7..0	00 to 0F H, 0 to 15 D	DCC Controller Number (equivalent to Handset or Throttle number)
7	7..0	C0 to FF H, 192 to 255 D	Loco Address byte 1 (MS Byte) for 4 digit Advanced Addressing Mode
	7..0	00 to 7F H, 0 to 127 D	Loco Address for 2 digit Baseline Addressing mode
8	7..0	00 to FF H, 0 to 255 D	Loco Address byte 2 (LS Byte) for 4 digit Advanced Addressing Mode
	7..0	00 H, 0 D	Set to zero for 2 digit Baseline Addressing mode
9	7..0	12 H, 18 D	Enter Advanced Consist mode with Normal Direction (per RP 9.2.1)
	7..0	13 H, 19 D	Enter Advanced Consist mode with Reverse Direction (per RP 9.2.1)
10	7..0	00 to 7F H, 0 to 127 D	2 digit Baseline address of Consist being formed
11	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 5 to 10 ONLY

No Reply to PC expected	

ii) Sub Type 1 Page Mode Programming (Write only)

PC Command to RPIC			Functional Definition
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always 'AA' Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always '55' Hex (85 decimal)
3	7..4	0 to F H, 0 to 15 D	RS485 address of board, value ignored in RS232 systems
	3..0	9 H, 9 D	Message Type 9
4	7..0	05 H, 5 D	Length of data to follow + 1 to allow for checksum
5	7..0	01 H, 1 D	DCC Sub Message Type 1
6	7..0	01 to 40 H, 1 to 64 D	CV Page Number (per RP 9.2.3 Paged Mode)
7	7..0	00 to 03 H, 0 to 3 D	CV Register Number within the Page (per RP 9.2.3 Paged Mode)
8	7..0	00 to FF H, 0 to 255 D	CV Data to be written
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 5 to 8 ONLY

No Reply to PC expected	

Further DCC related facilities may be added to the system as progress is made, which may necessitate the definition of additional Sub Message Types.

k) Message Type F (USB Operation)

Note that these commands refer to USB 1.1 Low Speed (1.5 MBit/s) operation with packet size restricted to 8 bytes (7 usable). All have the same length and all reply to the PC. Blocks number from zero to seven (32 bytes total). The contents of the actual PC USB HID Report Message Packet is exactly as shown below, except the 'AA' '55' serial header is formed by an automatically generated USB header byte instead. This is transparent to the User.

In a USB with RS485 Slave system, the USB RPIC strips the USB header byte before re-transmitting the message in the RS485 9 bit format described earlier. The 9 bit format is recognised by all RS485 RPIs and RPICs, and remains unchanged from the original system.

An RS232 RPIC will respond to the USB extension message formats as shown (with the 'AA, '55' header), but will ignore any board address content.

For RS485 systems using an RSB as the PC interface, the format below can also be used, as the RSB expects to see the 'AA', '55' header.

continued over >>>

i) Sub Type 0 Block Initialisation (4 byte block, input read and output write)

Note: This Sub Type does NOT prompt an RPC Stack read/write operation, it simply writes new values to the RPIC output data store and reads back existing input data store values, both held in the processor RAM.

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to15 D	RS485 address of board, used for slave RPICs, USB RPIC address = 0
	3..0	F H, 15 D	Message Type F
4	7..4	0 to 7 H, 0 to 7 D	Block number (e.g. block 0 = bytes 0, 1, 2, 3; block 1 = bytes 4, 5, 6, 7)
	3..0	0 H, 0 D	USB Sub Message Type 0
5	7..0	00 to FF H, 0 to 255 D	Value for first output byte in specified block (e.g. byte 0 for block 0)
6	7..0	00 to FF H, 0 to 255 D	Value for second output byte in specified block (e.g. byte 1 for block 0)
7	7..0	00 to FF H, 0 to 255 D	Value for third output byte in specified block (e.g. byte 2 for block 0)
8	7..0	00 to FF H, 0 to 255 D	Value for fourth output byte in specified block (e.g. byte 3 for block 0)
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 4 to 8 ONLY

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to FF H, 0 to 255 D	First input byte of specified block
2	7..0	00 to FF H, 0 to 255 D	Second input byte of specified block
3	7..0	00 to FF H, 0 to 255 D	Third input byte of specified block
4	7..0	00 to FF H, 0 to 255 D	Fourth input byte of specified block
5	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of Data Bytes

ii) Sub Type 1 Write Maximum of 2 Output bytes & Read Input Byte Change Status

Note: This Type DOES prompt an RPC Stack read/write operation.

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to15 D	RS485 address of board, used for slave RPICs, USB RPIC address = 0
	3..0	F H, 15 D	Message Type F
4	7..4	0 H, 0 D	Always zero
	3..0	1 H, 1 D	USB Sub Message Type 1
5	7..0	00 to 1F H, 0 to 31 D	Number of first output byte
6	7..0	00 to FF H, 0 to 255 D	Value for first output byte
7	7..0	00 to 1F H, 0 to 31 D	Number of second output byte (set to 0 if second byte not required)
8	7..0	00 to FF H, 0 to 255 D	Value for second output byte (set to 0 if second byte not required)
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 4 to 8 ONLY

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to FF H, 0 to 255 D	First 8 flags of input byte status change since last read. Bits 0 to 7 reflect
2	7..0	00 to FF H, 0 to 255 D	Second 8 flags of input byte status change since last read. Bits 0 to 7 reflect
3	7..0	00 to FF H, 0 to 255 D	Third 8 flags of input byte status change since last read. Bits 0 to 7 reflect
4	7..0	00 to FF H, 0 to 255 D	Fourth 8 flags of input byte status change since last read. Bits 0 to 7 reflect
5	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of reply bytes 1 to 4

iii) Sub Type 2 Read Maximum of 2 Input bytes

Note: This Type does NOT prompt an RPC Stack read/write operation, it simply reads back existing input data store values, held in processor RAM.

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to15 D	RS485 address of board, used for slave RPICs, USB RPIC address = 0
	3..0	F H, 15 D	Message Type F
4	7..4	0 H, 0 D	Always zero
	3..0	2 H, 2 D	USB Sub Message Type 2
5	7..0	00 to 1F H, 0 to 31 D	First input byte number of to be read
6	7..0	00 H, 0 D	Always zero
7	7..0	00 to 1F H, 0 to 31 D	Second input byte number of to be read
8	7..0	00 H, 0 D	Always zero
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 4 to 8 ONLY

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to 1F H, 0 to 31 D	Copy of first input byte number requested
2	7..0	00 to FF H, 0 to 255 D	Value of first input byte requested
3	7..0	00 to 1F H, 0 to 31 D	Copy of second input byte number requested
4	7..0	00 to FF H, 0 to 255 D	Value of second input byte requested
5	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of reply bytes 1 to 4

iv) Sub Type 3 Read Input Change Status Only

Note: This Type DOES prompt an RPC Stack read/write operation.

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always ‘AA’ Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always ‘55’ Hex (85 decimal)
3	7..4	0 to F H, 0 to15 D	RS485 address of board, used for slave RPICs, USB RPIC address = 0
	3..0	F H, 15 D	Message Type F
4	7..4	0 H, 0 D	Always zero
	3..0	3 H, 3 D	USB Sub Message Type 3
5	7..0	00 H, 0 D	Always zero
6	7..0	00 H, 0 D	Always zero
7	7..0	00 H, 0 D	Always zero
8	7..0	00 H, 0 D	Always zero
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 4 to 8 ONLY

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to FF H, 0 to 255 D	First 8 flags of input byte status change since last read. Bits 0 to 7 reflect
2	7..0	00 to FF H, 0 to 255 D	Second 8 flags of input byte status change since last read. Bits 0 to 7 reflect
3	7..0	00 to FF H, 0 to 255 D	Third 8 flags of input byte status change since last read. Bits 0 to 7 reflect
4	7..0	00 to FF H, 0 to 255 D	Fourth 8 flags of input byte status change since last read. Bits 0 to 7 reflect
5	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of reply bytes 1 to 4

As a general principle, Sub Type 3 would be used continuously, until an input change has been detected, then Sub Type 2 used to investigate details of the change(s) without having to re-read the RPC Stack.

v) Sub Type 6 LCD Text (maximum 32 characters)
Uses multiple 4 character blocks for long strings. String terminated with null character (0).
Carriage Return character 0D H (13 D) can be used to access second row of two line displays.

PC Command to RPIC			**Functional Definition**
Byte	Bits	Value Range	H = Hexadecimal, D = Decimal
1	7..0	AA H, 170 D	First Header byte, always 'AA' Hex (170 decimal)
2	7..0	55 H, 85 D	Second Header byte, always '55' Hex (85 decimal)
3	7..4	0 to F H, 0 to15 D	RS485 address of board, used for slave RPICs, USB RPIC address = 0
	3..0	F H, 15 D	Message Type F
4	7..4	0 H, 0 D	Always zero
	3..0	6 H, 6 D	USB Sub Message Type 6
5	7..0	0D to 7FH, 13 to 127D	ASCII value of next Text Character for display (0 = end of string)
6	7..0	0D to 7FH, 13 to 127D	ASCII value of next Text Character for display (0 = end of string)
7	7..0	0D to 7FH, 13 to 127D	ASCII value of next Text Character for display (0 = end of string)
8	7..0	0D to 7FH, 13 to 127D	ASCII value of next Text Character for display (0 = end of string)
9	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of bytes 4 to 8 ONLY

RPIC Reply to PC			**Functional Definition**
1	7..0	00 to 1F H, 0 to 31 D	Length of string transferred so far. Increases for each block sent.
2	7..0	00 H, 0 D	Always zero
3	7..0	00 H, 0 D	Always zero
4	7..0	00 H, 0 D	Always zero
5	7..0	00 to FF H, 0 to 255 D	Exclusive-OR checksum of reply bytes 1 to 4

Careful inspection of each variant of Message Type F indicates that bit 7 (the MSB) of the byte containing the USB Sub Message Type always remains at zero. A method of indicating USB Slave message failure has been devised to use this spare capacity, whereby if a USB to RS485 Slave message fails to obtain a correct response from the slave, the USB RPIC will detect the error, or timeout, and return to the PC a copy of the original USB output message with this bit 7 set to 1. Thus the PC is directly informed that an error has occurred. If all is well, the USB reply will contain the data requested, with this bit 7 set to 0. Clearly, this facility only applies to USB systems as there is no equivalent facility in the conventional serial formats. For these, it will be up to the PC Software to detect erroneous or absent responses.

The "TX" & "RX" LEDs fitted to each RPIC board can be used for investigating any problems with the data link.
If the "RX" LED lights and stays on, it indicates that the routine is still waiting for the specified number of bytes to arrive, indicating a possible length byte corruption or baud rate failure.

If the "TX" LED fails to light after a data message is received, this could indicate a number of problems including incorrect message type or length byte, incorrect checksum etc.

The RPIC processor software was developed to provide the facilities envisaged when the design was conceived. Further facilities may be added as required to suit any future developments.

Message Rates - Some examples of theoretically ideal cases:
(1 byte at 9600 Baud takes 10 bits x 104µs = 1.04ms)

A **Type 0** message containing 12 data bytes would take 2+1+1+12+1 (out) +12+1 (input) = 30 bytes x 1.04ms
= 31.25 ms per message: equivalent to a maximum message rate of 32 per second

A **Type 3** *bit* setting message would take 2+1+1+1+1 = 6 bytes x 1.04ms
= 6.25 ms per message: equivalent to a maximum message rate of 160 per second

A **Type 2** *byte* reading message would take 2+1+1+1+1 (out) +1+1 (input) = 8 bytes x 1.04ms
= 8.33 ms per message: equivalent to a maximum message rate of 120 per second

All of the above assume no delays between bytes and changes in direction, but are shown to give an idea of the rapid update rates possible with a serial communication system of this type. Indeed, the limiting factor is not so much the communication delay, but the speed at which the Layout Operating Program is able to make its decisions.

G16_4_2.doc