

# Serial Port Handling

*Summary note*

Fri, Feb 20, 1998

## *Initialization*

Initialization of the serial port used by IRMs is performed at system reset time, using parameters that are held in non-volatile memory, in fields of PAGESM system table entry #0 that is used for system-wide parameters. The default serial port parameters are for 8 bits, one stop, no parity, at 19200 baud. The baud rate may be changed by altering the non-volatile serial port parameters appropriately and resetting the system. Serial input data bytes are received under interrupt control and written into a serial input queue, the SERIQ system table (#19). Serial output is processed under interrupt control, where the data is contained in message blocks whose pointers are placed into a print queue, the PRNTQ system table (#13).

For operation with other baud rates, here are a few sample baud rate constants for use with IRMs using the MVME-162 CPU board with the Intel 85230 serial chip. For each two-byte value, the high byte is written to control register 13 and the low byte is written to control register 12. In the PAGESM entry at 0040202A are found the two words 0E0C and 000D by default, which specifies 19200 baud. The two bytes of the constant are divided into the high bytes of each of the words, with the control register number in the low bytes.

<i>Baud rate</i>	<i>Constant</i>
1200	0102
2400	0080
4800	003F
9600	001F
19200	000E
38400	0006

## *Serial port data reception*

Each character received generates an interrupt. If the character is an ascii NUL code (00) or LF (0A), it is ignored; otherwise it is entered into the SERIQ. This queue uses one IN offset and OUT1 and OUT2 offsets. When the queue is empty and idle, IN=OUT1=OUT2. As the serial interrupt routine places a character into the queue, it advances the IN offset. (If the advanced value of IN matches OUT2, the queue is full, and the IN advance must be inhibited, so that the character must be ignored.)

If the character just entered into the queue was a CR (0D), event #1 is sent to the Serial task to trigger it for further processing. If too many (250) characters have been received without a CR code, then one is entered into the queue and the task triggered. If the queue becomes full, then the CR overwrites the current character.

If the character received is X-OFF (13), a bit is set in the SERIQ header that inhibits further serial *output*. If the character received is X-ON (11), that bit is cleared, and serial output is restarted, if any serial output message blocks are queued to PRNTQ.

The Serial task awaits event #1, then reads from the queue each line of text that it finds there, advancing the OUT1 offset toward IN. A test is made for the special NBS serial

clock data, which is a 13-character sequence beginning with a SOH (01) character. In addition, a test is made (via SerCall) for an application's declared interest in the serial port data. If such interest is active, then the application is invoked with a special trigger type that indicates serial port data received. This feature was added to support a serial protocol used for a PLC interface. The protocol required several messages to be sent back and forth to process one transaction, so that it was important to be able to react quickly to new input, which may only be a simple acknowledgment string, so that the next message can be emitted right away.

As stated above, the IN offset of the SERIQ is advanced by the serial data receive interrupt. The OUT1 offset is advanced by the Serial task processing. What about the OUT2 offset? It is advanced by data request processing. A listype is defined for use with serial port I/O. A user may issue a data request for serial port data using that listype (#36). The reply data to such a request are formatted into a simple structure. The first two bytes of data in the user's buffer are an integer value of the number of characters that form complete lines—each ending with a CR code—that follow in the remaining part of the buffer.

Normally, only whole lines are delivered to a user. If no lines of text are available at the time the reply is due, according to the data request period specified, the first two bytes will be zero. Note that the user must be somewhat sensitive to the length of the lines of text that are expected; the number of bytes requested (and the buffer size presented) must be large enough to contain at least one maximum-sized text line including the CR code, plus the two bytes to contain the count value.

In the case that a line of text is read that is too large to be contained within the user's buffer, then those characters that will fit are placed in the buffer along with the count, and the rest of the line or lines awaiting in the SERIQ (up to the OUT1 offset) are skipped. This means that the user gets a truncated line of text, without a CR code.

As another example, if there are two lines waiting to be copied into the user's buffer, but only the first one fits in the buffer, then only the first line will be passed to the user in that reply. The next line must wait until the next reply is due. This means that the user must also be sensitive to the rate of serial input, so that the replies can keep up with that rate of input. At 19200 baud, each character requires 0.5 ms of serial time, so that characters can theoretically be received at 2000 per second. The default size of the SERIQ is 1024 bytes. It may be wise to make 15 Hz serial input data requests. See the next paragraphs for an additional incentive to do this.

Two problems can occur with the form of serial input data handling described above. If a user cancels a serial port data request, there will be no further advancement of the OUT2 offset. Any lines of text received by the serial port under such conditions will advance the IN and OUT1 offsets, but not the OUT2 offset. As a result, the queue may eventually become full. The next time serial input data is requested, a user might receive some rather old text that is unrelated to what is sought. Another problem that can occur is a partial line of text received into the SERIQ without a terminating CR code,

perhaps due to noise. Again, this text might wait a long time in the queue, until future input includes a CR code that terminates the line. Eventually a user may see the old partial line of text prefixing the first line expected.

The SERIQ monitor routine in the QMonitor task watches over serial input data reception each cycle to resolve both of these potential problems. If the OUT2 offset remains constant and not equal to OUT1 for too long, then all lines of text waiting to be read via OUT2 advancement are skipped. The timeout for this logic is only 2 cycles. This means that a user who uses a serial port data request should specify a reply period of the maximum rate of one cycle.

If the OUT1 offset remains constant and not equal to the IN offset for too long, both the OUT2 and OUT1 offsets are advanced to the value of the IN offset. This means that all such unread serial input is skipped. The timeout for this logic is 900 cycles, which is 60 seconds at a 15 Hz cycle rate.

In summary, serial input queue handling has the IN offset advanced by characters being entered into the queue. The OUT1 offset is advanced as lines of text are accepted each of which ends with a CR code and checked for matching the NBS clock string format. The OUT2 offset is advanced by a user consuming lines of text, or as lines are tossed for which there is no user. A local application that has expressed an input in serial data is invoked whenever a line has been received in the SERIQ. Such a program will monitor the contents of the SERIQ to process new serial data, rather than using a serial port data request.

### *Serial port data output*

When data is to be written to the serial port, a routine is called that builds a serial data message block and copies the user's text into it and appends CR and LF characters. (It is also possible to invoke a routine that omits this appending.) A listype (#36) used for a setting message allows sending a line of such text either locally within a page or local application, or across a network. (Listype #84 may be used for outputting text without appending CR, LF.)

A serial message block is constructed to hold the text, a pointer to which is placed into the print queue PRNTQ. In this queue's header is a flag that is set when serial output is active and cleared when it is idle. After placing the pointer to a message block into the queue, if the flag indicates that serial output is idle, the PrntStrt routine is called to initiate serial output—to prime the pump. Further entries into the queue, when the flag indicates that the output is active, cause no special action to take place.

As each character is transmitted to the serial port hardware interface, an interrupt occurs that indicates that the interface can accept another character. The interrupt routine uses the OUT1 offset of this queue to find the current entry in PRNTQ and checks whether all characters from the current message block have been transmitted. If more remain in that buffer, it sends out the next one. When the last character in the text of

that message block has been transmitted, the OUT1 offset is advanced, and the next message block, if any, is checked. This continues until the IN offset is reached, at which point all characters in the print queue have been transmitted, so the flag is cleared to indicate that serial port output is idle.

As mentioned in the section above on serial port reception, there is a flag bit in the SERIQ header that is set when an X-OFF is received and cleared when an X-ON is received. The serial output processing respects this flag. If it is set, serial output processing is suspended. When it is cleared, by reception of the X-ON character, serial output processing is resumed.

The QMonitor task includes print queue monitoring that imposes a timeout of 8 seconds (at 15 Hz cycle rate) on each line of serial output. While the OUT1 offset is different from the IN offset, and the current message block times out, the assumption is made that the serial output interface is being held up or broken. In order to lessen the chance of running out of memory resources because of being unable to output all lines of text that are queued, the OUT1 offset is advanced past all message blocks until it matches the IN offset. Note that this timeout is suspended during the time that serial output is held up due to an X-OFF having been received. In addition to this timeout processing, print queue monitoring advances the OUT2 offset and frees all message blocks until it matches the OUT1 offset.

In summary, print queue handling has the IN offset advanced as text message blocks are queued. The OUT1 offset advances as lines of text are delivered to the serial output hardware interface. The OUT2 offset advances as message blocks are freed.