TO: Distribution

FROM: M.J. Grady

DATE: August 2, 1973

SUBJECT: Remote Printer DIM Design

This document describes the implementation of a DIM to support the Mohawk Data Sciences 2400(MDS2400) as a remote printer/reader on Multics.

GENERAL

The Mohawk Data 2400 is a mini-computer with a variety of peripherals and a high speed communications channel. Mohawk supplies a number of emulators for the 2400, one to emulate an IBM 2780 Remote Batch Terminal and another to emulate the Honeywell G115 Remote Batch Terminal. The IBM 2780 runs on a binary synchronous communications channel to the central computer, while the G115 uses an ASCII synchronous channel. Since the 355/HSLA will support an ASCII synchronous channel directly, i.e. with no additional hardware, it has been decided that the initial implementation of remote peripherals on Multics will use the G115 interface. Honeywell has standardized this interface, called the GRTS/355 Remote Computer Interface. It is documented in the Honeywell publication Remote Terminal System(GRTS), form number DA79. This interface requires the synchronous transmission of messages, or blocks of data. These messages are composed of a header, which contains various control information, the actual text or data and a block check character, which is used to detect errors in the message. The interface also requires that every message be acknowledged either positively or negatively. This permits retry operations for messages which had errors during transmission, and the detection of multiple transmission of the same data.

REQUIRED SOFTWARE

The implementation of this interface on Multics requires work in 3 different software areas. Since the device is connected to the High Speed Line Adapter(HSLA) and therefore the 355 communications computer, the tty Device Interface Module(DIM) must be used for the hardcore interface to the device. The areas of change are: 1) Work will be done in the 355 to allow it to handle synchronous ASCII communications channels. The buffering strategy of the 355 must be changed to allow it to transmit and receive data synchronously at 4800 baud, the proposed speed of the communications line for the 2400. 2) The ring zero tty DIM will be changed to add the new device type and the corresponding

DCW lists, which will pipeline all data between the 355 and the g115_ DIM in ring four. Also, the buffering strategy of the tty DIM must be changed to allow it to accept whole messages from the 355. This implies two changes: a) the 2400 device type must be guaranteed a larger percentage of the total buffer space and b) more than two buffers must be allocated on a read request to the 355, to allow enough time to add new buffers when the interrupt indicating that the first buffer has been filled is received. It is essential that the tty DIM pass only complete messages back and forth to the ring four DIM. 3) A ring four MDS2400 Device Interface Module must be written to support the standard interface and all of the devices connected to the 2400. This DIM will have complete responsibility for running the device, formatting the output, checking for errors and acknowledging messages.

The Ring Four MDS2400 DIM

Since the MDS2400 DIM will be driven by an IO Daemon driver process, it will want to talk to each device connected to the 2400 over a different stream. To implement this we will construct a transfer vector and an entry in the attach module for each device on the 2400. All other entries in the transfer vectors will call the same entries in the DIM. Thus we will determine the stream-device coordination at attach time, and during all other calls we will be able to coordinate the transfer of data between each of the devices over the single communications line through the DIM. Each stream attached to the DIM will allocate a structure similar to the one in Appendix A. This structure will identify the device and contain a pointer to the master control structure of the DIM.

The Daemon driver will begin running the device when a select command is received from the 2400. The select is issued by the operator after he has prepared an appropriate command deck as outlined in MSB-106, and it will cause one card to be read. Note that a card reader is not required, and if it is not present the Daemon driver process will attach its command input stream to user_input and read commands from there. This will allow the MDS2400 to function only as a remote printer, without the added expense of the card reader.

Since the DIM will spend most of its time running the remote printer, optimum use should be made of the data space available in each message. To do this two steps will be performed in the DIM in formatting the message: 1) the DIM will make use of the data compression techniques allowed by the GRTS/355 interface; i.e. when more than three identical characters are transmitted they will be compressed into the sequence "char,US,count" where "US" is a control character indicating compression, and 2) records, which consist of individual print lines, will be split across messages to make full use of the 324 data characters allowed per message. The DIM will also be responsible for code

converting the data for the printer and supplying the correct carriage control(slew) characters. The code conversion routine for the PRT300 should be easy to adapt this purpose.

The 2400 will also be used for reading cards, most of which will be control cards used to communicate with the Daemon. These cards must start with "$*$" in order to be transmitted to the Daemon as control cards. The DIM will remove the "$*$" before returning the command to the Daemon so that they appear to be the same as console commands. The Daemon will be requesting a card between each print request, to determine if any special operator action was requested. To read a card the DIM will send a transmit data request to the 2400 which may respond with a card or a negative acknowledgement. It is important that if the Daemon aborts the request for a card, that the DIM clean up its request to the 2400. The device transmits ASCII, and when reading non-control cards it transmits them compressed, multiple records per message, one message per request. Although the device will run at 4800 baud on input there should be no buffer allocation problem since each message must be acknowledged before the next message is sent by the remote computer.

Since the remote device acknowledges every message sent to it, and since the DIM must send an acknowledgement for each message received, some thought must be given to the amount of asynchronous processing which can be done by the DIM. When writing to the remote printer, the DIM will be given a large amount of data which it must break up and send out in messages. Thus on a write call the following steps seem likely:

1) convert the data for printer and make one output buffer
2) write the buffer to the remote device
3) code convert and make the next output buffer
4) read the acknowledgement, block if not received yet
5) if the transmission was good switch buffers and go to step 2
6) otherwise rewrite the first buffer and go to step 4

Notice that step 3 should give the device time to process the data and send the acknowledgement and there would be no purpose served by preparing more buffers at that time. Additional complications may arise if the acknowledgement message had errors during its transmission. The standard interface allows for that case by providing a sequence code which alternates between successive messages which contain new data. Thus if the acknowledgement had errors, the last message would be retransmitted without changing its sequence code. This is done even if there were no errors in its first transmission to allow the remote device to retransmit its acknowledgement.

The case for reading data from the remote device is similar and goes as follows:

1) send a Transmit Data request to the remote device
2) read the message sent by the remote device, block if not received
3) check the message for errors; if bad, send negative acknowledgement and go to step 2
4) send positive acknowledgement combined with a transmit request
5) process the data received from the first request
6) read the message sent in response to last request, block if not arrived yet
7) check the message; if bad, send a negative acknowledgement, a transmit request and go to step 6
8) switch input buffers and go to step 4

Note that in all cases the acknowledgement is a part of a standard message which requests other actions. Thus step 1 just sends a transmit request, while step 4 sends a positive acknowledgement combined with a transmit request. Also each data message from the remote device contains an acknowledgement of the last message sent by the central computer, which was probably an acknowledgement message, which must also be checked. One can easily see the endless problems of trying to run on a terrible communications line.

It appears from the above discussion that some asynchronous processing can be done while the I/O is going on but it is not clear if it will significantly improve the processing.

OPERATOR INTERVENTION

There are 3 buttons on the 2400 which allow operator communication with the central system. The operator's primary communication will be with the control cards, but to get things going and to interrupt printing the buttons may be used.

The first is the select button, which causes a Select service message to be sent to the central computer. It is used to initiate communication with the central computer, and causes one card to be read. This should only be used after the initial connection following the dial-up.

The second button is the backspace-file button which causes a backspace-file service message to be sent to the central computer. The DIM will act without notifying the Daemon and will restart the request from the point of the last write call, probably the beginning of the segment.

The third button is the abort button, which sends the abort service message, and will cause the DIM to signal the quit condition. The Daemon will catch the quit and will respond by reading a card from the remote device.

CONCLUSIONS

The DIM to drive the Mohawk Data 2400 will be fairly complex and will be required to do a large amount of work in setting up and transmitting data and in receiving and acknowledging responses from the 2400. The GRTS/355 Remote Computer Interface seems complete for error detection and correction, but may be difficult to implement. The required changes to the 355 HSLA software to run synchronous ASCII communications channels will require some thought, along with the changes to build the right kind of DCW lists in the tty DIM.

APPENDIX A

```
/* stream data block for g115_ dim */

dcl 1 sdb based(sdbp) aligned, /* one per stream */
      2 outer_module_name char(32),
      2 device_namep ptr init(addr(device_name)),
      2 device_name aligned,
        3 next_ptr ptr init(null),
        3 name_size fixed bin init(32),
        3 name char(32),
      2 type bit(4), /* device type of this stream */
      2 dikp ptr; /* pointer to DIM control structure */
```