MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

MAC Memorandum M-229
March 18, 1965

TO: F. J. Corbato

FROM: S. Dunten

SUBJECT: Proposed Datanet-30 program

Project MAC plans to use the Datanet-30 as a temporary subsitute for the Multi Line Controller (MLC) to communicate with teletype or teletype like devices. The Datanet-30 will have no I/O equpiment attached except a Computer Interface Unit (CIU) and 64 bit buffers. Initially the bit buffers will service 34 IBM 1050's and 30 model 35 teletypes, however past experience has shown that the quantity and kinds of terminals connected to the MAC system are very apt to change. Also the possibility that the MLC will not be ready as soon as planned requires that the Datanet-30 is capable of servicing as many lines as possible. These two considerations place certain requirements on the Datanet-30 program:

1. The quantity and speeds of terminals should be assembly parameters.
2. The Datanet-30 program will perform minimal functions so that a maximum number of lines may be handled.
3. Some means should be available for determining the amount of idle time in order to estimate the maximum number of lines.

The functions performed by the Datanet-30 program will be only those necessary to interface the console lines with the GE 635. The processes of Datanet-30 buffer allotment, code conversion, data set control, and message construction will be performed in the 635. Thus the Datanet-30 program must allow the 635 to:

1. Read each data character as it arrives in the Datanet-30. The Datanet-30 will not format messages but instead will place each input

character in a common input queue for transmission to the 635.

2. Read the status of the data set sense leads and be informed of any change in their state.

3. Activate the data set control lead or control the echo mode.

4. Transmit output characters in blocks. Block transfers are used instead of single character transfers to gain efficiency, however the 635 must retain the ability to terminate output at any time and determine what portion of the block had been transmitted.

5. Have a second block of 32 output data words in the Datanet-30 while the first block is being typed. This double buffering will allow continuous output without pauses between blocks.

The Datanet-30 program will consist of several sections:

1. A bit scanning program which is called at each bit time for each bit rate. This routine executes the SCAN instruction for the lines of this bit rate.

2. A character scanning program operating each character time. Here is where characters are moved to and from the hardware scan words and where the control functions are performed.

3. A program which scans and tests the conditions of the data set sense leads informing the 635 of any change. This should be called every second or so.

4. A section devoted to moving information to and from the 635. This routine operates in whatever time is available between scans.

5. A controlling metronome routine which uses the elapsed time clock (Q counter) to call the various scans at the appropriate times. This program is controled by tables containing the bit time of each device, the character time of each device, the data set polling period, and the priority of each scan. The 4 priority levels should be used in such a way that a character scan, for example, may be interupted to perform a bit scan.

The Datanet-30 which project MAC has ordered does not have enough memory to allow the extravagance of double buffers dedicated to each line. Instead, each buffering requirement should obtain a buffer from a buffer pool. The 635 will know the size of the buffer pool, allow for input buffers, and not write more than the remaining buffers can hold. When the Datanet-30 goes to the online mode of operation all buffers should be reset.

The logic required to control the 635's writing into the double buffers is straight foward. Data from the 635 is written into an output secondary buffer while the character scan routine takes characters from an output primary buffer. When the primary buffer exhausts and the secondary buffer contains data the scan program moves the secondary buffer to the primary buffer and sends a buffer completion to the 635. Whenever both buffers are depleted it sends a "no more output" signal.

Only one input buffer is needed since each word in the buffer contains the number of the line which it pertains to. This buffer will obtain memory blocks from the buffer pool and chain these together forming a variable length buffer. The character scan program will add words to the back of the input buffer while the 635 is reading from the the front.

The Datanet-30 program operates in two modes. When the program is first loaded it is in the off-line mode. While in this mode the Datanet-30 should not attempt to send information to the 635. When transferred to the online mode all buffers and switches should be reset and communications initiated with the 635. In addition to 635 control there needs to be buttons of some sort on the Datanet-30 to set the on-line/off-line mode.

The control and data formats between the Datanet-30 and the 635 follow. The functional control message has no special meaning; it is merely the first 4 words of the transmission. All transmissions end with a word of all ones.

635 to Datanet-30

Each transmission contains a maximum of 34 words and involves only one line. The first word specifies the line number to which the remaining data and/or control words apply. When bit 18 of this word is on, the output buffers for this line should be reset after returning a sense 3, but before reading the remaining information. During the character scan if a data or control word has bit 18 off it is a data word and is ready to be placed into scan word 3 of the bit buffer. Otherwise, the low order bits specify the control function to be performed as listed below.

```
 _____________________________________
|R|              LINE                 |
|S|___________________________________|   ____
|                                     |    A
|_____________________________________|    |
|0|              CHARACTER            |    |
|_|___________________________________|    |
|          or                         |  32 words
|_____________________________________|  or less
|1|              CONTROL              |    |
|_|              FUNCTION_____________|    |
|                                     |    |
|_____________________________________|  __V__
|1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1|
|_____________________________________|
18 17                               1
```

1 Place all lines in the on-line mode.
2 Place all lines in the off-line mode.
3 Turn data set control lead on. (DEF 3)
4 Turn data set control lead off. (DEF 4)
5 Return the state of the data set sense leads for all lines. (sense 4 and 5)
6 Set echo mode on. (DEF 6)
7 Reset echo mode. (DEF 7)

Datanet-30 to 635

Each of the 32 or fewer words in a transmission is independent of any other and involves only one line. If bit 18 is off, bits 9 through 2 contain the character received from the line specified by bits 17 through 10. Otherwise the sense data (bits 6 through 1) is defined by the sense code (bits 9 through 7) for line N (bits 17 through 10).

```
 ____________________________________
|0|       LINE       |     CHAR     | |
|_|__________________|______________|_|
18 17              10 9             2 1
```

or

```
 ____________________________________
|1|       LINE       |SENSE|  SENSE   |
|_|__________________| CODE|  DATA____|
18 17              10 9   7 6        1
```

Sense code	Meaning	Sense data
1	Completion	none
2	No more output	none
3	Output buffer word count	Word count

4	Data set lead 1	0=off, 1=on
5	Data set lead 2	0=off, 1=on

The following pages illustrate the suggested logic of the Datanet-30 program. The language used is shown below.

```
               DO through alpha, for a group of lines
               Perform these statements for
               each line in the group.
alpha          continue
```

```
               IF condition
               THEN
                    Do these statements
                    if the condition is true.
               OTHERWISE
                    Do these statements
                    if the condition is false.
               Always do these statements.
```

```
Each bit time:
               execute SCAN instruction for the lines of this bit rate
               transfer to return


Each character time:
               DO through charloop, for lines of this character rate
               IF a character has been read
               THEN
                    format and place in "to 635 buffer"
more           IF the output switch is set
                  AND the BBC is ready for a new character
               THEN
                    IF buffer1 is empty
                    THEN
                         IF buffer2 is empty
                         THEN
                              send "no more output" (sense 2)
                              turn off output switch
                              transfer to charloop
                         OTHERWISE
                              move buffer 2 to buffer1
                              send completion (sense 1)
                    get next word from buffer1
                    IF bit 18 is on
                    THEN
                         perform function
                         transfer to more
                    OTHERWISE
                         place word in scan word 1
charloop       continue to next line
               transfer to return


Each 1 sec. interval:
               DO through dsloop, for every line
               IF the data set leads are different than old leads(line)
               THEN
                    place sense 4 and/or 5 in "to 635 buffer"
                    old leads(line)=current data set leads
dsloop         continue to next line
               transfer to return


During idle time:
xmitloop       IF there is anything in "to 635 buffer"
               THEN
                    send it
               OTHERWISE
                    count idle time
               IF the 635 wants to talk
               THEN
                    listen to line number
```

```
     IF bit 18 is on
     THEN
          send sense 3
          reset output buffers
     read remainder of message into output buffer2
     set output switch
OTHERWISE
     count idle time
transfer to xmitloop
```

The metronome routine uses the following variables;

run	Index of scan program currently running.
nextrun	Index of next scan program to be run.
time	Current time.
traptime	Time of this or next clock trap.
clock	The Q counter.
dt	Preset vector of scan periods.
priority	Preset vector of scan priorities.
entry	Preset vector of the entrys to the scanning programs.
mach	Vector of machine conditions saved on clock traps.
nextime	Vector containing the time when each scan should next be performed.
trapped	Vector of scan programs interupted.
missed	Count of scans not started in time.

and has the following logic:

```
On each clock trap:
                mach(run)= machine conditions
                trapped(nextrun)=run
loop            run=nextrun
loop1           nextrun=index of smallest nexttime whoes
                   priority is greater than priority(run)
                time=traptime-clock
                clock=-(nexttime(nextrun)-time)
                traptime=nexttime(nextrun)
                IF clock is less than 0
                THEN
                              missed(run)=missed(run)+1
                              transfer to loop
                restored machine conditions=mach(run)
                transfer to the program counter
return          nexttime(run)=nexttime(run)+dt(run)
                mach(run)=entry(run)
                run=trapped(run)
                transfer to loop1
```

Additional programming which is desireable:

1. While in the off-line mode, format and place input data characters into BUFFER2. Upon receiving the repeat character set the output switch causing the buffer to be typed back. This repeat feature makes it possible to checkout teletype to Datanet-30 communications without involving the 635.

2. A Datanet-30 core dump program using the 635 printer.

3. The necessary I/O packages to assemble the Datanet-30 source deck using 635 I/O for the input, output and scratch files, and a loader to load the resulting object program.