# 2236 MXE Software Commands

The following article describes a few useful commands of the new 2236 MXE controller board software. It is based on the Customer Engineering Product Maintenance Manual entitled "2236MXE Terminal Controller (Early Field Support) July 1982"

## Entering MXE Command Mode

The Model 2236 MXE responds to certain user commands by entering a "Command Mode". By using this feature, the user can set both the transmission rate and which port is to perform as Terminal 1 on the system by means of the software alone, without manually resetting any switches on the MXE. Command mode can be used to set a system password, to print a message on all terminals connected to the Model 2236 MXE, to test the memory in the RAM of the board, to restart the power on diagnostics, to lock the current transmission rate of the port issuing the command, and to list all possible commands. Command mode can also be used to test particular ports on the Model 2236 MXE. All Command mode statements are password protected so that only a privileged user has access.

To enter the MXE command mode press the "LOAD" key three times (do not type LOAD). After pressing LOAD three times the prompt on the screen will display "ENTER MXE COMMAND:" followed by a new line of "%". The user should then enter the desired MXE command. A description of the commands and how to enter the commands will follow. The MXE will process the command and prompt for another command until a blank line is entered thus putting the user back in the previous mode. Reset will also cause an exit from MXE command mode, but will also reset the attached partition and clear the screen. The MXE command should not be entered until the MXE prompts for it.

Any terminal may enter MXE COMMAND MODE at ANY time. If the 2200 is printing to the screen the printing will be suspended during MXE Command mode to prevent the two outputs from becoming intermixed. MXE command mode will not restore the screen to its pre-command mode state.

## COMMAND DESCRIPTIONS

In the command descriptions that follow, use these definitions:

The command 'psw' is a six character password containing no blanks, (the default is 'MXEPSW').

The command 'port designator' is a one character designator that is as follows:

The port at which the command is being typed is zero so that the user need not know which port s/he is connected to. The absolute MXE port addresses are 1, 2, 3, and 4.

NOTE: System terminal numbers bear no relationship to the MXE port numbers in command mode.

All commands begin with a ONE character command code. Most commands then have a six character password, which is the MXE password (similar in use to the 2200 system password) followed by any needed parameters in the order specified below. The user can always type help (while in MXE command mode) to obtain a list of command codes. Various releases of the MXE software may allow various subsets of or additions to the command list.

The command line is divided into three different fields. The first field is a one character field that states the command to be run. The second field is six characters and contains the desired password for the MXE command.

NOTE: The password will not be printed on the screen. Instead, the MXE will print the numbers 0 through 5 as characters are typed so that the password can be protected.

The third field provides the needed parameters, such as port designator, etc. Each field is separated by at least one blank.

## COMMANDS

This section covers the commands and a step by step procedure for using them. There are two command codes that are very important to the C.E. in setting up a 2236 MXE board.

NOTE: Not every release of the operating system allows every possible MXE command. Some commands may not be allowed in certain system modes.

The first important command code is 'H' (HELP). This command allows you to see all the command code letters and definitions. Proceed with the following steps to enter the 'H' (HELP) command code:

### 'H' (HELP) ALL COMMAND CODES

Steps:

1. Type 'H'
2. Return

Example: %H

CRT Displays:

```
A  Set VP user
B  Set Baud Rate
C  Set psw
D  Download
E  Analog Loopback
F  Digital Loopback
G  Print
H  Help
I  Ram Test
J  Restart
L  Lock
?  Status
```

NOTE: Refer to Troubleshooting Section 5.4.1 for 2236 MXE command codes E,F,G, and I.

The second important command code is '?' (Status). '?' shows the status of all ports on the 2236 MXE board. In detail it shows the following: 'ON' or 'OFF' line condition (connect or not); 'Software Baud Rate'; 'L'(baudrate lock); '(Hardware Baud Rate)'; 'Local' or 'Remote' connection; 'Wang Terminal'. Proceed with the following steps to enter the ? (Status) command code:

'?' (STATUS)

Steps:

1. Type '?'
2. Return

Example: %?

CRT DISPLAYS:

```
2236 MXE RO (BOOTSTRAP)
* PORT 1 ON  19200 L / (19200)  LOCAL  WANG TERMINAL
  PORT 2 OFF 19200   / (19200)  LOCAL  WANG TERMINAL
  PORT 3 OFF 19200   / (19200)  LOCAL  WANG TERMINAL
  PORT 4 OFF 19200   / (19200)  LOCAL  WANG TERMINAL
```

NOTE *-Indicates the port issuing the command.

'A' SET PRIMARY USER

Steps:

1. Type 'A' blank
2. Password(psw), type 'MXEPSW' blank (any 6 characters)
3. Port Designator, type 'X' (ports 1 through 4)
4. Return

Example: %A 012345 2

NOTE: The primary user must be set while in the bootstrap (VP mode). Once the MVP operating system is loaded the primary user may not be changed. A later release will support this feature.

CRT DISPLAYS: OK

The port designated has become the new primary user in the VP Mode and is thereby capable of booting up the system. Only the first MXE (at address 00) may accomplish this.

### 'B' SET BAUD RATE

Steps:

| | |
|---|---|
| 1. Type | 'B' blank |
| 2. Password(psw), Type | 'MXEPSW' blank (any 6 characters) |
| 3. Port Designator, Type | 'X' (ports 1 through 4) blank |
| 4. Baud Rate, Type | '1200' (1200=example) |

Example: %B 012345 2 1200

CRT DISPLAYS:
```
2236 RO (BOOTSTRAP)
* PORT 1 ON  19200 L / (19200) LOCAL WANG TERMINAL
  PORT 2 OFF  1200   / (19200)  LOCAL  WANG TERMINAL
  PORT 3 OFF 19200   / (19200)  LOCAL  WANG TERMINAL
  PORT 4 OFF 19200   / (19200)  LOCAL  WANG TERMINAL
Are you SURE?(Y/N): Y (answer Y or N)
OK
```

NOTE: A blank answer exits command mode for this command without changing baud rate.

To assure that the new Baud Rate has been set properly, perform a '?' (STATUS) check. Refer to '?' (STATUS).

NOTE: Setting the baud rate through software overrides any hardware switch setting. The baud rate desired is now set at the port designated. Rebooting the system ($INIT) will reset ports to their hardware baud rates.

### 'C' SET PASSWORD

Steps:

| | |
|---|---|
| 1. Type | 'C' blank |
| 2. Password(psw), Type | 'MXEPSW' blank (any six char) |
| 3. New password(newpsw), type | 'NEWPSW' (any six char) |
| 4. Return | |

Example: %C 102345 NEWPSW

CRT DISPLAYS:
```
C///////////////////////////
OK
```

'NEWPSW" becomes the new password.

# MODEL 2236MXC

# MULTIPLEX CONTROLLER

# INTERIM MANUAL

(16 September 1977)

# PREFACE

This interim manual includes information about the Wang Model 2236MXC Multiplex Controller. It provides some information necessary to create a user-written application program using logic control characters to control multiple Model 2236 Interactive Terminals via the Multiplex Controller. Related information is available in the following manuals:

In order to use the Wang-supplied subroutines contained in the Terminal Access Method (TAM) software package, the Terminal Access Method User Manual should be referenced.

This manual assumes its reader has a programming knowledge of Wang's BASIC or BASIC-2 language, as described in the BASIC Language Reference Manual supplied with the CPU.

Operational features associated with Model 2236 Interactive Terminals are described in the Interactive Terminal Introductory Manual supplied with a Model 2236 Interactive Terminal.

INTRODUCTION

The Model 2236MXC Multiplex Controller contains either four (Model 2236MXC-1) or eight (Model 2236MXC-2) "ports" which allow it to control up to four or eight terminals respectively. Each "port" handles asynchronous full-duplex communication and related I/O buffering for the terminal to which it is connected.

By virtue of its microcoded firmware and individual "ports" for each terminal, the Model 2236MXC Controller relieves the CPU (and thus the programmer) of the byte-by-byte transfer of data between the keyboard, CRT screen and optional printer of all the terminals, and the single CPU. For example, when a terminal's keyboard is enabled by the program, the Controller stores each keyed character's code in that port's line buffer, and simultaneously "echos" the character back to that terminal's appropriate CRT coordinates for operator verification. Meanwhile, the program is free to enable the keyboard and set up prompts at other terminals, poll the terminals and obtain a completed field, check and process completed fields, and if polling indicates no completed fields, perform background functions such as printing at the CPU site or a remote terminal site, or completing a search operation. Fields are indicated as complete by an operator-keyed terminator key (RETURN, FN, or S.F. keys) and are CPU-buffered after polling along with (1) the terminal number which facilitates setting up the next field and (2) the terminator key's hex value for special functions including menu selection.

A terminal, once "selected" by a program, can function as a normal 2200 CRT/keyboard for displayed or printer output (as well as keyed input) without interrupting keying. A programmer need not be concerned with buffer timing problems other than checking to ensure that previously filled output buffers have been emptied by the Multiplex Controller before filling them with new output data (especially at low baud rates). Each port contains a separate CRT buffer and print buffer; each terminal contains its own 80 x 24 CRT with refresh memory and a print buffer. Either device nicely facilitates the output of retrieved data and/or program-generated information.

In summary, the program is responsible for processing functions, data transfer between the CPU buffers and the CPU's storage devices, and controlling the Model 2236MXC which automatically handles multi-terminal buffering, timing and communication.

The illustration below shows the physical buffering characteristics associated with an Interactive Terminal and its port in the Multiplex Controller. The following letters indicate data transfer as shown in the illustration and explained below.

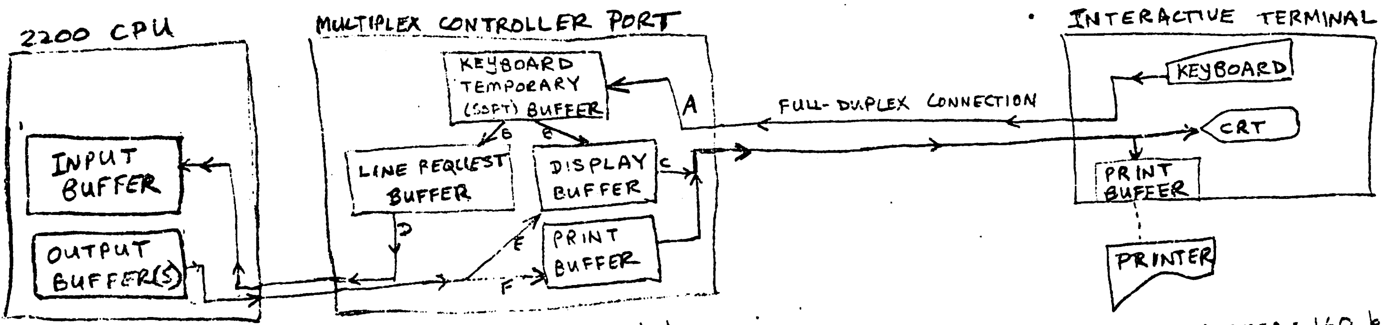
A - ASCII character code generated by keystroke is transmitted to the Multiplex Controller Port and stored in the keyboard temporary (soft) buffer.

B - Multiplex Controller firmware automatically places the character code in the Line Request Buffer (for subsequent CPU buffering) and the Display Buffer (for terminal's "echo").

C - A character code in the Display Buffer is automatically transmitted back ("echoed") to the Interactive Terminal's CRT screen for visual verification by the Multiplex Controller, or program-generated message is sent to the CRT (see E below).

D - When operator indicates completion of entry, and polling detects the ready condition, the Line Request Buffer's contents are transferred to the program-designated CPU buffer for processing. The parallel I/O interface between the CPU and the Multiplex Controller provides excellent data transfer rates.

E, F - Program-generated information is displayed (SELECT PRINT 007) or printed (SELECT PRINT 204) at the currently-selected terminal. Such information includes clear screen functions, operator prompts, the current entry field, retrieved data, and hardcopy output. Displayed CRT output positions are determined by cursor positioning. An entry field, once enabled, completes A-B-C loop automatically until completion (D).



CPU BUFFERS ARE PROGRAM-DESIGNATED

KEYBOARD BUFFER: 34 bytes
LINE REQUEST BUFFER: 216 bytes (80bytes w/TAM)
DISPLAY BUFFER: 512 bytes
PRINT BUFFER = 160 bytes

PRINT BUFFER: 160 bytes

## Program-Generated CRT Output

The allowable character set for displayed Interactive Terminal CRT output includes 112 graphic characters. Their hex values are (1) from hex 10 to 7F without underline and (2) from hex 90 to FF with underline (add hex 80 to underline a character). The character set and their respective hex values (without underline) appear below.

> NOTE:
>
> Hex codes 02,04, and 80 to 8F are reserved for system use and thus must not be used in any user-written program.

The following control code hex codes from 00 to 0F are valid:

Hex 01- home cursor
Hex 03- clear CRT, home cursor
Hex 05- cursor ON
Hex 06- cursor OFF
Hex 07- audio tone (alarm)
Hex 08- backspace (←)
Hex 09- non-destructive space (→)
Hex 0A- cursor ↓
Hex 0C- cursor ↑
Hex 0D- carriage return

## Program-Generated Printer Output

Hex codes used for printing purposes are valid if (1) they are in the ranges listed for displayed output, and (2) are valid characters supported by the printer used. This assumes that the printer is attached to an Interactive Terminal (address 204).

A printer connected to the CPU is addressable via its 2XX printer address, e.g., SELECT PRINT 215. Such printing is not influenced by the Model 2236MXC Multiplex Controller.

| HEX | CHAR | HEX | CHAR | HEX | CHAR | HEX | CHAR |
|---|---|---|---|---|---|---|---|
| 10 | â | 30 | 0 | 50 | P | 70 | p |
| 11 | ê | 31 | 1 | 51 | Q | 71 | q |
| 12 | ↑ | 32 | 2 | 52 | R | 72 | r |
| 13 | ô | 33 | 3 | 53 | S | 73 | s |
| 14 | û | 34 | 4 | 54 | T | 74 | t |
| 15 | ä | 35 | 5 | 55 | U | 75 | u |
| 16 | ë | 36 | 6 | 56 | V | 76 | v |
| 17 | ï | 37 | 7 | 57 | W | 77 | w |
| 18 | ö | 38 | 8 | 58 | X | 78 | x |
| 19 | ü | 39 | 9 | 59 | Y | 79 | y |
| 1A | ȧ | 3A | : | 5A | Z | 7A | z |
| 1B | ė | 3B | ; | 5B | [ | 7B | § |
| 1C | u̇ | 3C | < | 5C | \ | 7C | £ |
| 1D | Ä | 3D | = | 5D | ] | 7D | ë |
| 1E | Ö | 3E | > | 5E | ↑ | 7E | ς |
| 1F | Ü | 3F | ? | 5F | • | 7F | ɋ |
| 20 | SPACE | 40 | @ | 60 | › | | |
| 21 | ! | 41 | A | 61 | a | | |
| 22 | " | 42 | B | 62 | b | | |
| 23 | = | 43 | C | 63 | c | | |
| 24 | $ | 44 | D | 64 | d | | |
| 25 | % | 45 | E | 65 | e | | |
| 26 | & | 46 | F | 66 | f | | |
| 27 | ' | 47 | G | 67 | g | | |
| 28 | ( | 48 | H | 68 | h | | |
| 29 | ) | 49 | I | 69 | i | | |
| 2A | • | 4A | J | 6A | j | | |
| 2B | + | 4B | K | 6B | k | | |
| 2C | , | 4C | L | 6C | i | | |
| 2D | – | 4D | M | 6D | m | | |
| 2E | . | 4E | N | 6E | n | | |
| 2F | / | 4F | O | 6F | o | | |

**Programming Note:**

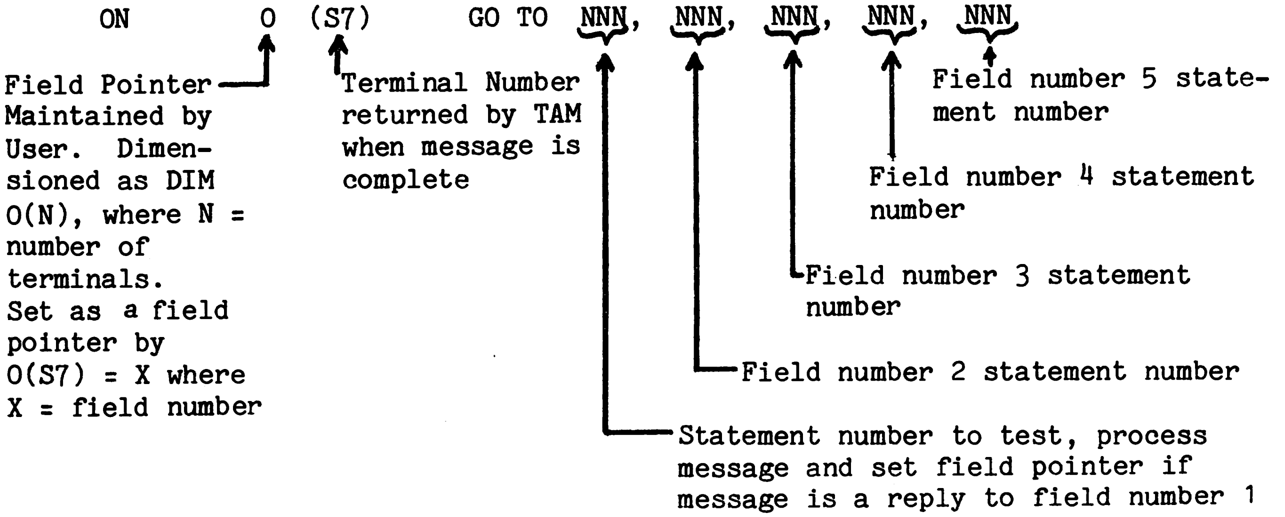
To output underlined characters, use HEX() functions, or add hex 80 to all the characters in a variable. The latter technique is illustrated at line 30, below.

```
10   DIM N$25
20   INPUT "ENTER NAME", N$              :REM KEYBOARD ENTRY
30   ADD(STR(N$, 1, LEN(N$)), 80)        :REM UNDERLINE CHARACTERS
40   PRINT N$                            :REM PRINT UNDERLINED RESPONSE
```

## Multiple Terminal Programming Techniques

Multiple terminal programs handle several terminals usually with one or more identical prompts or screen images needed for their common single task application. One entry field is active per terminal at any one time, but the program may provide for skipping fields or field backspacing, obtaining a different screen image, or other special functions by testing the hex value of the key used to terminate an entry field. The ability to control multiple protected input fields is totally under program control and requires a fairly structured approach to multiple terminal programming. Other functions involved include polling, setting-up prompts and keyboard input fields, and maintaining field pointers for each terminal.

With typical 2200 programming using a single CRT/Keyboard Console, a completed entry field is tested and processed by the mainline program and no polling is involved. With multiple terminal programming, a one dimensional array is used to maintain the field pointer for each possible terminal and each possible field. In order to maintain the field pointers, when polling reveals a terminal has a completed field, the program requests the field, and a TAM subroutine* returns the terminal number in variable S7, the field's contents in a different variable, and also allows identification of the terminator key used. The program, through the use of an ON GO TO statement, tests a one dimensional array's contents to determine the statement number to test and process the entry field, and increments the field pointer for that terminal. All processing routines return to a common statement which enables the keyboard for the appropriate field. The last field sets the field pointer to the first field and handles required output to external storage devices. The ON GO TO statement appears below. A flowchart of multiple terminal programming, which assumes fixed screen prompts remain on the screen once established, follows.



```
ON      O  (S7)     GO TO NNN, NNN, NNN, NNN, NNN
```

Field Pointer Maintained by User. Dimensioned as DIM O(N), where N = number of terminals. Set as a field pointer by O(S7) = X where X = field number

Terminal Number returned by TAM when message is complete

Field number 5 statement number

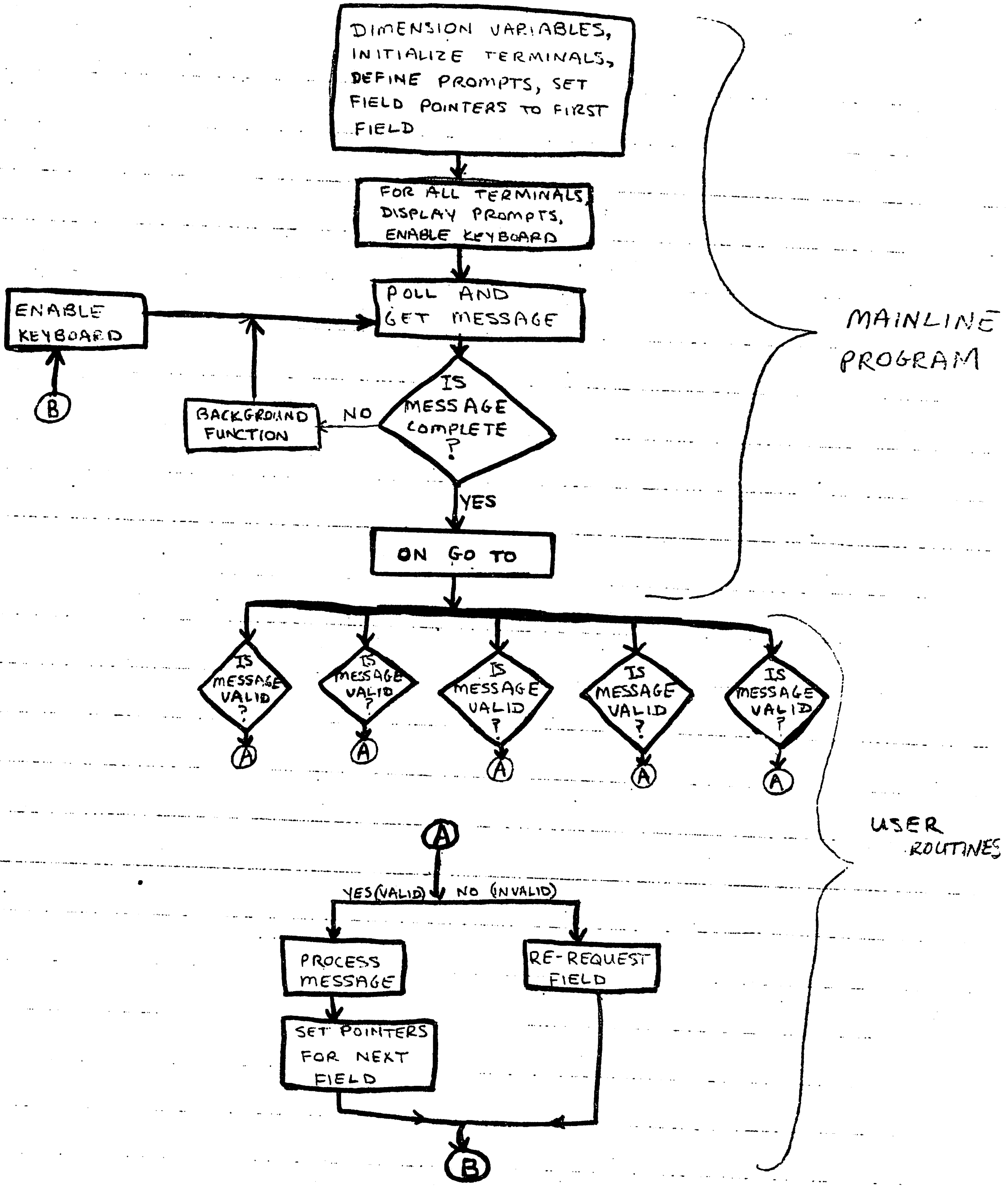
Field number 4 statement number

Field number 3 statement number

Field number 2 statement number

Statement number to test, process message and set field pointer if message is a reply to field number 1

* POLL TERMINALS or POLL TERMINALS AND RETURN MESSAGE. Alternatively, the Receive Controller Status operation may be used as described later in this manual.


DIMENSION VARIABLES, INITIALIZE TERMINALS, DEFINE PROMPTS, SET FIELD POINTERS TO FIRST FIELD
FOR ALL TERMINALS, DISPLAY PROMPTS, ENABLE KEYBOARD
ENABLE KEYBOARD
POLL AND GET MESSAGE
B
BACKGROUND FUNCTION
NO
IS MESSAGE COMPLETE ?
YES
ON GO TO
IS MESSAGE VALID ?
A
IS MESSAGE VALID ?
A
IS MESSAGE VALID ?
A
IS MESSAGE VALID ?
A
IS MESSAGE VALID ?
A
MAINLINE PROGRAM
USER ROUTINES
A
YES (VALID)
NO (INVALID)
PROCESS MESSAGE
RE-REQUEST FIELD
SET POINTERS FOR NEXT FIELD
B

## ADDRESSING

The 2236 MXC controller can be enabled and accessed by a set of seven different device addresses. Typically HEX(01), HEX(02), HEX(03), HEX(04), HEX(05), HEX(06) and HEX(07). Since however, the high order two bit address of the controller board are switch setable, each controller can be set to addresses HEX(01) thru HEX(07), HEX(41) thru HEX(47), HEX(81) thru HEX(87), or HEX(C1) thru HEX(C7).

When the different device addresses are used to enable and access the controller, they specify different functions or modes of operation to the controller as follows:

| Addresses | Operation |
|---|---|
| HEX(01) and HEX(05) | Console Input, Console Output operation to 2236 terminal number 1 only. (Power on, Initial Program loading, etc). |
| HEX(06) | Control and Status operations to the controller. (Send commands to the controller to select a terminal, request lines to be entered for terminals, etc). |
| HEX(03) | Receive the data entered in a terminated line. |
| HEX(07) | Send Display Data to the "currently selected terminals" CRT. |
| HEX(04) | Send a Print line to the slave printer of the "currently selected" terminal. |
| HEX(02) | Receive complete status of the terminal control (CRT, print buffer empty full, entered lines ready, etc). |

## OPERATION MODES

A. Console Input/Output (Output to address 05, Input from address 01)

When a 2200 CPU is first powered on or is not currently executing a program, it allows system commands to be entered and displayed via a connected CRT terminal, always assuming address HEX(01) for the keyboard and HEX(05) for the CRT. When enabled by these addresses the 2236MXC locks solely onto 2236 terminal number 1 and supports system console input, console output operation.

Whenever the two high order 2236MXC address switches are set to 0, the #1 Interactive Terminal will act like a 2226 in every manner. (i.e., the RESET key works as one expects, and editing is displayed and works in the usual manner).

Any keystrokes made on Interactive terminals #2 to #8 are ignored. However, the terminal will clear it s own screen whenever RESET is keyed.

Unlike the 2226, the only BASIC verbs supported by the keyboard are CONTINUE, CLEAR, LOAD and RUN.

B. Program Control Mode

Program Control Mode is a special mode in which communication between all terminals is supported. This mode becomes activated whenever the 2236MXC is enabled by address 06. In this mode, output may be sent to and input received from any of the 4 or 8 possible terminals. This I/O capability is performed by using either one of the following addresses: 02, 03, 04, 06, and 07. Each address will allow the 2200 program to communicate with a terminal in a specific manner.

In order to more fully understand the operation of Program Control Mode, one should have a little insight as to the setup of the MXC in respect to each Interactive Terminal.

The MXC sets aside a portion of it s memory for each of the possible 4 or 8 terminals. This memory is then used to provide space for the line request characters, a buffer for all CRT output and a buffer for all PRINTER output. Additional memory is used for pointers, flags and etc. by the MXC.

## 3.1 Address HEX(06) Control Commands

Address 06 allows the 2200 program to define which terminal is to be communicated with and what tasks it is expected to perform.

In particular, address 06 can

a) cause the flow of data to be directed to and from a particular terminal (SELECT terminal).
b) cause cursor positioning to be performed. (POSITION CURSOR).
c) define a line request.
d) cause one or all terminals to be initialized.

The functions and command codes of address 06 are defined in more detail below. Although the address 06 commands are described individually below, they can be transmitted to the controller in various combinations in a single BASIC statement.

In the following explanations of the address 06 commands, there are generally three means of communicating with the 2236 MXC. The three methods are:

```
        $GIO
        PRINT HEX(
and     PRINT alpha-variable
                or
        PRINT USING alpha-variable
```

$GIO is the recommended method and must be used if control information is sent out via alpha-variable.

PRINT alpha-variable will cause a problem if variable values such as row, column, number of characters in line are output.

The potential problem is this:

With the PRINT command if the alpha-variable contains a byte with the value of 13, HEX(0D) the system interprets that character as carriage return, an extra character is generated after it.

The extra character generated is dependent upon the device type. For example, a code of $0A_{16}$ is generated for a device type 0 and a code of $00_{16}$ is generated for a device type 2.

It should be noted that if device type 7 is used on the 2200VP no extra character is generated. In general however, the $GIO command with an A000 microcommand should be used to output information from an alpha variable.

```
Example,  100 $GIO /006 (A000, B$) C$
          200 $GIO /006 (A000, B$) C$ <1, 10>
```

A second problem of some concern is the automatic carriage (return)/linefeed which is generated if the BASIC interpreter believes it is at the end of a print line. This problem may be solved by using the SELECT PRINT XYY statement before the PRINT HEX( or PRINT(USING) alpha-variable statement which resets the line character count.

On the 2200VP a device type 7 and a line width of 0 would alleviate both problems.

It is because of these two problems that the $GIO statement is the recommended communication statement.

<u>SELECT TERMINAL HEX(F4XX)</u>

Whenever a command code of F4 is received, the next byte will determine to which terminal communication is to be directed to or received from. The data byte must be a hexadecimal representation of the desired terminal (i.e., 01 = terminal #1, 02 = terminal #2, ..., 08 = terminal #8). For the remainder of this memo the current terminal is the last SELECTed terminal.

Example, select terminal #1

```
A)      10 SELECT PRINT 006
        20 PRINT HEX(F401);

        or

B)      10 $GIO /006 (40F4 4001, Q6$)
```

<u>POSITION CURSOR HEX(F7XXYY)</u>

A command code of F7 will cause the CRT of the current terminal to be positioned at row XX and column YY. XX and YY must be hexadecimal representation of the desired row or column. The 2236 Interactive Terminal has 24 rows; numbered 0 to 23, and 80 columns; numbered 0 to 79.

Example, position cursor of current terminal at row 10 column 32.

```
        10 SELECT PRINT 006
        20 PRINT HEX(F70A20);
        or
        10 DIM R$3
        20 R$ = HEX(F7)
        30 BIN(STR(R$,2,1)) = 10
        40 BIN(STR(R$,3,1)) = 32
        50 $GIO /006 (A000, B$) R$
        or
        10 $GIO /006 (40F7 400A 4020, Q6$)
```

## REQUEST LINE HEX(03XXAABB)

A command code of 03 will cause the 2236MXC to setup to receive a field of up to XX characters (a hexadecimal representation of the count, not to exceed 216) starting from the current CRT cursor position for the currently selected terminal. All field entries will be forced to stay within the field limits. A line request is active until either a carriage return or a special function key is entered. Edit mode may be initiated (BB = 01) or suppressed (BB = 00). The characters previously stored in the keyboard soft buffer may (AA = 01) or may not (AA = 00) be allowed to be treated as entered characters for the line. (In other words, keystrokes received prior to a line request being set, can be either received as part of the line or deleted). If deleted they are never echoed back to be displayed on the CRT.

Example, from the current position of the current terminal setup a line request of 20 characters, currently buffered characters may be treated as valid keystrokes and suppress edit mode.

```
10 SELECT PRINT 006
20 PRINT HEX(03140100);
or
10 $GIO /006 (4003 4014 4001 4000, Q6$)
or
10 C$ = HEX(03)
20 BIN(STR(C$,2,1)) = 20
30 STR(C$,3,2) = HEX(0100)
40 $GIO /006 (A000, B$) C$ <1,4 >
```

There are 4 variations of the line request command. These are programmed in the above manner with only 1 change.

A) Command 03 -- Set up line request echo characters only. (i.e., no underline)

B) Command 04 -- Set up line request echo characters with underline.

C) Command 05 -- Set up line request echo characters only and initialize field with spaces on CRT

D) Command 06 -- Set up line request echo characters with underline and initialize the field with underlined spaces on CRT.

## INITIALIZE LINE REQUEST  HEX(07XXXX...FFOD)

A command code of 07 is used after a line request command of HEX(03) or HEX(04) to initialize the desired line on the CRT with the supplied characters XXX... starting with the left most position in the field. Any non-space characters received are treated as protected characters and are automatically skipped over in entry mode. The string of characters is terminated by a carriage return (OD) or a HEX(FF) code. The cursor is positioned at the left most non-protected character.

Example, setup a line request to receive today's date in the form of MM/DD/YY.

```
        90 SELECT PRINT 006
        100 PRINT HEX(04080000);
        110 PRINT HEX(07); "  /  /  "; HEX(OD);

or      110 PRINT HEX(07); "  /  /  "; HEX(FF);

or      110 A$ = "  /  /  ":STR(A$, 9, 1) = HEX(FF)
        120 $GIO /006 (4007 A000, Q6$) A$
```

## PREFILL REQUEST LINE  HEX(08XXXX...OD)

A command code of 08 can be sent either after a line request command 03 or 04 or immediately after an Initialize Line Request Command 07 to prefill the desired line with the supplied characters XXX... starting with the left most position. The characters are treated as keystrokes and will skip over protected character, if any exist. The cursor is left at the left most non-protected character. The string of characters is terminated by a carriage return, HEX(OD).

Example, initialize today's date as 06/03/77

assuming line request has been made and initialized

```
100 SELECT PRINT 006
110 PRINT HEX(08); "060377"; HEX(OD);
or
110 A$ = "060377":STR(A$, 7, 1) = HEX(OD)
120 $GIO /006 (4008 A000, Q6$) A$
```

## END OF LINE REQUEST SEQUENCE HEX(F5)

A special command must be supplied to signal the end of a line request sequence which consists of the setup, any initializes and prefilling desired. Thus a line request, plus any initialization command may be sent out in several statements or as one string of characters in one statement. The last command sent however, must be a HEX(F5), to signal the microcode to invoke the lire request.

Example,

assuming setup, initializes and prefill are complete.

```
100 SELECT PRINT 006
110 PRINT HEX(F5);
or
110 $GIO /006 (40F5, Q6$)
```

## INITIALIZE ALL TERMINALS HEX(F8)

This command will cause the screens of all terminals to be cleared, and pending requests and input buffer data to be cleared.

Example, clear the screens and buffers of all terminals.

```
10 SELECT PRINT 006
20 PRINT HEX(F8);
or
10 $GIO /006 (40F8, Q6$)
```

## INITIALIZE CURRENT TERMINAL HEX(F9)

This command will clear the CRT screen, pending request and input buffer of current terminal.

Example,

```
100 SELECT PRINT 006
110 PRINT HEX(F9);
or
110 $GIO /006 (40F9, Q6$)
```

## DELETE CURRENT LINE REQUEST HEX(0C)

This command is similar to the initialize current terminal except the CRT screen is not cleared.

Example,

```
100 SELECT PRINT 006
110 PRINT HEX(0C);
or
110 $GIO /006 (400C, Q6$)
```

## B.2 Address HEX(03) Receive terminated line

Address 03 is used to get the data associated with the terminated line request of the currently selected terminal into the 2200. This will be done by a program after a status check indicated a line has been received and terminated. The alphanumeric variable or array setup to receive the line should be sufficiently large to receive the entire line. It it is not the additional characters will be truncated.

```
Example, 100 $GIO /003 (C620, Q6$) W$ (line terminated with SF Key)
         or
         100 Q6$ = HEX(OD) (line terminated with SF key or CR)
         110 $GIO /003 (C630, Q6$) W$
```

Termination by either a special function key or a special character, HEX(OD), is determined by the 8th byte of the Arg-2 variable, Q6$.

If the 20-bit is on, termination was by special function key.

If the 40-bit is on, termination was by the special character, HEX(OD).

```
100 AND(STR(Q6$, 8, 1), 60)
110 ON VAL(STR(Q6$, 8, 1)) /32 GOTO 200, 300
.
.
.
200 REM TERMINATED BY SPECIAL FUNCTION KEY
.
.
.
300 REM TERMINATED BY SPECIAL CHARACTER
.
.
.
```

### B.3 Address HEX(07) Send Data to CRT

Address 07 is used to transmit characters to be displayed onto the CRT of the current terminal. Since the CRT output buffer in the controller is limited to 512 characters it is generally most efficient to send CRT output of blocks 512 bytes or less, waiting for ready (buffer empty) prior to sending the next block. (Or else the CPU will be hung up awaiting the buffer to empty).

Example,

```
        100 SELECT PRINT 007
        110 PRINT HEX(030A0A0A0A); TAB(10); "NAME", TAB(30);
            "ADDRESS" etc.
```

### B.4 Address HEX(04) Send Line to Slave Printer

Address 04 is similar to address 07 except that the characters are directed to the printer of the`current terminal. Print data is sent a line at a time (up to 160 characters). A test for ready (printer buffer empty) should be made prior to sending out the next print line for efficient operation. (Or else the CPU will be hung up awaiting the buffer to empty).

Example,

```
        100 SELECT PRINT 204
        110 PRINT HEX(0C); "NAME"; N$
        120 PRINT "ADDRESS"; A$
```

## B.5 Address HEX(02) Receive Controller Status

Address 02 is used to report the statuses of the various buffers to the 2200. When enabled by address 02 the 2236 MXC will send 32 bytes of data and 1 ENDI data byte to be used as a terminator for the input sequence.

| Bytes | Buffer | Explanation (HEX Value) |
|---|---|---|
| 1-8 | Line Request | 30 - no terminated line request this terminal.<br>31 - terminated line request this terminal. |
| 9-16 | Terminal On/Off Status | 30 - this terminal not powered on.<br>31 - this terminal powered on. |
| 17-24 | CRT | 30 - buffer empty this terminal.<br>31 - buffer not empty this terminal. |
| 25-32 | PRINTER | 30 - buffer empty this terminal.<br>31 - buffer not empty this terminal. |
| 33 | ENDI Terminator | |

Example

Is the CRT buffer of terminal #4 empty?

```
5 DIM W$ 33
10 $GIO /002 (C620, Q6$) W$
20 IF STR (W$,20,1) = "0" THEN 40
30 GOTO 10
40 REM
```

Summary, the following example shows how one can setup a line request at row 2 column 0 of terminal #3, except the entry and print the response.

```
10 $GIO /006 (40F4 4003, Q6$)  Select Terminal #3
20 SELECT PRINT 007      Display Prompt
30 PRINT HEX(030A); "ENTER NAME (default = END)"
35 SELECT PRINT 006
40 $GIO /006 (40F7 4002 4000, Q6$) position cursor
50 $GIO /006 (4004 4020 4001 4000, Q6$) setup line request,
      echo with underline, use soft buffer as character, non
      edit mode
60 PRINT HEX(08); "END"; HEX(0D)  prefill request
70 $GIO /006 (40F5, Q6$)  end of line request sequence
80 $GIO /002 (C620, Q6$) W$  Is line terminated
   :IF STR (W$,3,1) = "0" THEN 80
90 INIT (20) N$: $GIO /003 (C620, Q6$) N$  get name
95 $GIO /002 (C620, Q6$) W$:IF STR(W$, 27, 1)<> "0" THEN 95
100 SELECT PRINT 204
110 PRINT "NAME="; N$
120 GOTO 20
```

## B.6 Additional Features of Program Control Mode

EDIT — editing is represented by a blinking cursor. When in edit mode only SF' 11 and 27, 12 and 28, 13 and 29 and 14 and 30 are operations and mean move right 5 positions, move right 1 position, move left 1 position and move left 5 positions respectively. Remember protective characters are skipped.

$IF ON addr (Ready/Busy Status for Control Addresses)

for addr = 02, a ready means one of the terminals has a terminated line.

for addr = 03, a ready means the current terminal has a terminated line.

for addr = 04, a ready means the printer buffer of the current terminal is empty.

for addr = 07, a ready means the CRT buffer of the current terminal is not full.

## CONTROLLING A NON-WANG TERMINAL

The multiple terminal program can control non-Wang, RS-232-C compatible, Teletype® - equivalent terminals. The program can select the Teletype mode and define the nature of a non-Wang terminal during program initialization. A variety of Teletype-equivalent CRT terminals can be defined by the following parameters: screen or carriage width up to 255; asynchronous character formats including odd, even, or no parity; 1, 1.5, or 2 stop bits; and 5, 6, 7 or 8 data bits; type of destructive backspace; and a printer terminal's ASCII keycodes for destructive backspace and cursor up. If a CRT terminal (not a printer terminal) is in use, its keycodes for cursor home, up, down, left, and right as well as clear screen and clear line keys may be defined. Screen/printer line roll is automatic. Program-controlled cursor positioning is supported only for Wang terminals, but is possible for non-Wang CRT output terminals.

A non-Wang terminal must be equipped to operate at the transmission rate set for it at Multiplex Controller (300, 600, 1200, 2400, 4800, or 9600 baud).

Teletype mode is similar to Console Input/Output mode in that entered lines when terminated by a receiver carriage return will cause a CR and LF character to be echoed back to the CRT to cause a line ROLL. It differs internally however, in that as opposed to the terminal controller interacting directly with the 2200 CPU on a character by character basis, it will buffer the line being received as in the fixed screen mode. In addition,teletype mode will be limited to a single line edit capability, destructive backspace. It will support the entry of up to 216 characters on multi-CRT lines. In the teletype mode, only the standard ASCII codes are recognized.

The following new control commands will be available when the controller is enabled by Device Address HEX(06) to support teletype mode:

HEX(09 BB XX YY ZZ AA) Enable currently selected terminal in teletype mode

where BB = output mode for communication of MXC and terminals

format = SS PP WW 10

SS — # stop bits

00 — illegal
01 — 1 stop bit
10 — 1.5 stop bits
11 — 2 stop bits

PP — Parity Selection

00 -- none
01 -- odd
10 -- none
11 -- even

WW — # bits in data word

00 -- 5
01 -- 6
10 -- 7
11 -- 8

10 -- is required

XX = screen/carriage width
YY = Type of destructive backspace
01 = echo received code back only
02 = receive code is cursor left, must produce destructive backspace by echoing (cursor back, space, cursor back)
03 = receive code indicates destructive backspace but no echo to be done (i.e., teletype)

ZZ = code received from terminal to indicate destructive backspace.
AA = cursor up code (if zero, line limited to carriage width unless YY = 03)

HEX(0A BB) Enable currently selected terminal in fixed line mode.

BB = Output mode for communication of MXC and terminals. (See previous section for description of BB).

In teletype mode the only additional control commands that can be used are:

HEX(03 XX) Request a line
HEX(F9) Initialize currently selected terminal (i.e., clear any previously received characters).

If it is desired that foreign terminals be useable in fixed screen mode (for applications such as Mask Screen Data entry) the following additional command is needed to provide sufficient information to the controller to do additional screen control:

HEX(0B AA BB CC DD EE FF GG)

Initialized terminal control codes

AA = Terminal Cursor Home Code (up)
BB = Terminal Up Cursor Code
CC = Terminal Down Cursor Code
DD = Terminal Left Cursor Code
EE = Terminal Right Cursor Code
FF = Clear Screen Code (if any)
GG = Terminal Clear Line Code (if any)

Chris [illegible]

WANG LABORATORIES, INC.

MEMORANDUM

TO: Bob Kolk, Ed Demeo, Tech Center
FROM: Roger Droz, Dave Angel
DATE: February 27, 1978
SUBJECT: The 2236MXC vs the 2236MXD

There have been a lot of questions and confusion about the 2236MXC used in non-TAM applications, such as providing remote terminal capability on a 2200T or VP system. Most of the problems arise when users attempt to use the slave printer jack on the back of the 2236 terminal when running a single 2236 terminal as a remote console. This problem was addressed specifically in a memo on December 13, 1977. This memo shall present some guidelines as to when it is appropriate to configure a 2236MXC and when a 2236MXD is more appropriate.

The 2236MXC was designed for use with a single user 2200T or VP CPU. The Terminal Access Method subroutines (TAM) were written to provide a means for a single BASIC program to talk to up to 8 terminals. The $GIO sequences necessary for the user to write his own "TAM" are documented.

The 2236MXD was designed for use with the 2200 MVP system. TAM routines could be written for the MXD but have not yet been. The $GIO sequences necessary for user written multi-terminal operations have only been documented for internal use.

Neither controller was originally intended for use as a secondary console for a 2200 system. There has, however, been interest in RS-232 compatible consoles for remote or dial-up use. Because of some problems encountered in attempting to use the 2236MXC and 2236 terminal in non-TAM applications, some features were incorporated into the 2236MXD and 2236D to make them suitable for use with single user systems. The MXD performs quite well on a single user T or VP. Most of the MXD features, such as keystroke buffering and character compression, also function on a single user system. Most important, a local printer or plotter plugged into the 2236D terminal functions properly. In short, the 2236MXD - 2236D combination works as well on a single user system as it does on an MVP. In fact, the 30 byte keystroke buffer may solve end of field speed problems in some high speed data entry applications.

Without TAM like software, only port number 1 of the MXD may be used. The terminal is addressed the same on a single user system as it is on an MVP: the keyboard is address 0X1, the CRT is address 0X5, and the printer is 0X4; where x is 0, 4, 8 or C depending on how the address switches on the MXD are set.

It would not be difficult to write a version of TAM for use with the 2236 MXD. The MVP system is, however, recommended for all multi-terminal applications, because the MVP is faster and easier to program. It is false economy to buy a less expensive single user system and perform the software juggling necessary to support multiple terminals. The savings in software development time by letting the MVP operating system select the proper terminal to talk to rapidly makes up any difference in initial hardware cost.

not decision [illegible] not just now! but X [illegible] software.

The 2236 terminal and 2236MXC controller should probably be phased out in favor of the 2236D/2236MXD. As mentioned above, it is possible to write the necessary software to allow the 2236MXD to support multiple terminals on a single user system. The desirability of offering such a software product is somewhat questionable, considering that we have an MVP system that does the job so much better. The only feature missing from the MXD is the ability to support foreign terminals. If this is an important feature, a new version of the MXD with more PROM space to allow both Wang and foreign terminals or a special MXD for foreign terminals could be offered. what about current on-line applications in an interrupt environment?

The 2236MXD is thus to be preferred over the 2236MXC except when:

1) The user wishes to support multiple terminals from a single user application program using the currently offered TAM subroutines.

2) Foreign terminals must be supported.

What foreign terminals ~~off~~ supported with MXC?

~~re~~ N.B. Dr. Harris' DECwriter

To: Distribution
From: Eric Wilson
Subject: MXE code release for BASIC-2 Release 2.4
Date: 10/08/82

The following operational limitations exist in the current release of @MXEO (the 2236MXE operating software):

1) The system can be powered up from the first terminal ONLY. (This is compatable with the MXD terminal controller) The capability to power up the system from other terminals will be added in the next release of the MXE operating code.

2) The LED on the MXE is not valid after the system has been configured. The LED should be checked after power up but before selecting an operating system. When the 2200 is loaded with an operating system the LED will be turned off REGARDLESS of errors on the board!

3) As the timing is different between the MXE and the MXD a problem exists which may prevent the IDEAS (except Rel 2.0 - see below) and WP packages from determining if a terminal printer is available. This problem will only exhibit itself when the BASIC program is trying to print to the terminal printer. The program may think the terminal printer is available when it is not, causing the program to hang until the printer becomes available or the program is reset. This is not a catastrophic error and will not cause damage to the users document.

4) IDEAS 2.0 may not be able to print to the terminal printer through the MXE for similar reasons to above. Users currently running IDEAS 2.0 should obtain the forthcoming IDEAS maintenence release for use with the MXE terminal controller.

NOTE: The above printer problem is not a bug in the MXE. It is due to a faulty implementation in the WP and IDEAS software packages and should be rectified in upcoming releases of both. SYSTEM PRINTERS are not affected!

Distribution:

| | |
|---|---|
| J. Belanger | MS 8236A |
| J. Proulx | MS 8236A |
| G. Mantoni | MS 1309B |
| J. Thibault | MS 1309B |
| [redacted] | [redacted] |
| F. Sullivan | MS 1213 |
| P. Seymour | MS 1383 |
| E. Kelley | MS 1383 |
| R. Jones | MS 1329 |
| D. Morelle | MS 2752 |