MULTICS TECHNICAL BULLETIN MTB-195

To:	Distribution
From:	J. W. Gintell
Date:	May 8, 1975
Subject:	Technical paper describing MCS

The attached document was developed by Warren Johnson (Multics Marketing in Phoenix) to be given to Multics customers. It gives a technical overview of MCS and broadly describes the present and future plans.

Honeywell

```
**     **          **
***   ***          **   **
** * * **          **  ****  **
**  *  **          **   **
**     **  **  **  **   **   **  ******  ******
**     **  **  **  **   **   **  **      **
**     **  **  **  **   **   **  **      ****
**     **  **  **  **   **   **  **          **
**     **  ******  **   ***  **  ******  ******
```

Multics Communications System

May 1975

Multics Project Office

Large Systems Sales Support

Phoenix, Arizona

BACKGROUND

When Multics first became operational on the Honeywell Model 645, user terminals (as well as other peripherals) were connected to the system through a device called the "Generalized Input/Output Controller" (GIOC). The hard-wired GIOC could respond to connect instructions from a processor, find and interpret data control words (DCWs), transfer data to and from main memory, post status words, and signal termination to a processor via an interrupt; but the GIOC did not possess any capabilities to intelligently handle the wide variety of terminals available. Thus, all terminal support capabilities were imbedded within Multics itself.

This remained the case as Multics migrated from the Honeywell Model 645 hardware to the Model 6180. A simulation of the GIOC was created for the DATANET Front-End Network Processor (FEP), even though it was capable of assuming many of the terminal support duties. Hence, Multics initially treated the FEP as a physical line-control device only and continued to support a limited number of terminal types within itself. This GIOC simulation was unnecessarily slow, inefficient, and cumbersome. A single FEP only was supported, and it was difficult to adapt the code to handle different terminal types.

GOALS

The Multics Communications System (MCS) is designed to take advantage of the processing power of the FEP. By allowing the FEP to handle such functions as line control and line protocol, the central system is free to do other things. Also, the redesign of the communications software allows the new code to take advantage of such hardware features as the Extended Instruction Set (EIS). These features add up to an overall improvement in system performance. Furthermore, MCS is designed to allow for easy implementation of new devices as well as to support multiple FEPs.

All changes are transparent to the Multics user. Terminal types and configurations previously supported continue to be supported. Also, many new terminal types and options are supported under MCS.

In general, MCS does not pretend to be all things to all people. It is intended to take maximum advantage of the current hardware offering and to be flexible enough to allow additions and modifications to be made easily.

MCS TIMETABLE

The following lists outline the features MCS is designed to provide, along with some ideas for future enhancements. The Multics Release Number is indicated because the following sections are written as if MCS development were complete.

MULTICS RELEASE 3.0, SEPTEMBER 1975

NEW FEATURES

1. Correspondence code IBM 2741s
2. Uppercase only terminals at all baud rates
3. Support for GE TermiNet 1200 or similar devices on Bell 202C modem, with ETX type line control
4. Support for dialup synchronous lines, 2400 to 9600 baud
5. Multiple FEPs
6. Login control argument to specify terminal type
7. Block input, i.e., full screen, for video terminals
8. Non-readahead mode for non-interruptible terminals
9. Continuous input mode
10. White space canonicalization for output
11. Support for
    a. 7-bit ASCII character set
    b. 7-bit ASCII uppercase only character set
    c. 6-bit ASCII character set
12. New modes: tabecho, lfecho, echoplex

DEVICE CLASS	EXAMPLES
ASCII	Teletype Models 37, 38, GE TermiNet 300
1050	IBM 1050
2741	IBM 2741, Trendata 1000
ARDS	ARDS, Tektronix
SYNC	Mohawk 2400

BAUD RATE	ASYNCHRONOUS	SYNCHRONOUS
110	LSLA,HSLA	
134.5	LSLA,HSLA	
150	LSLA,HSLA	
300	LSLA,HSLA	
600	HSLA	
1200	HSLA	
1800	HSLA	
2000		HSLA
2400		HSLA
4800		HSLA
7200		HSLA
9600		HSLA

NEW CENTRAL SYSTEM SOFTWARE

1. Old interrupt handler and line-control interpreter, tty_inter is eliminated
2. Line-control table deleted, function moved to FEP
3. New DIA interface module handles FEP interrupts and controls DIA traffic
4. Segment tty_ctl contains only the code conversion tables and is paged
5. Terminal type changeable by the user to any other system-recognized type
5. New buffering strategy with smaller wired buffer area

NEW FEP SOFTWARE

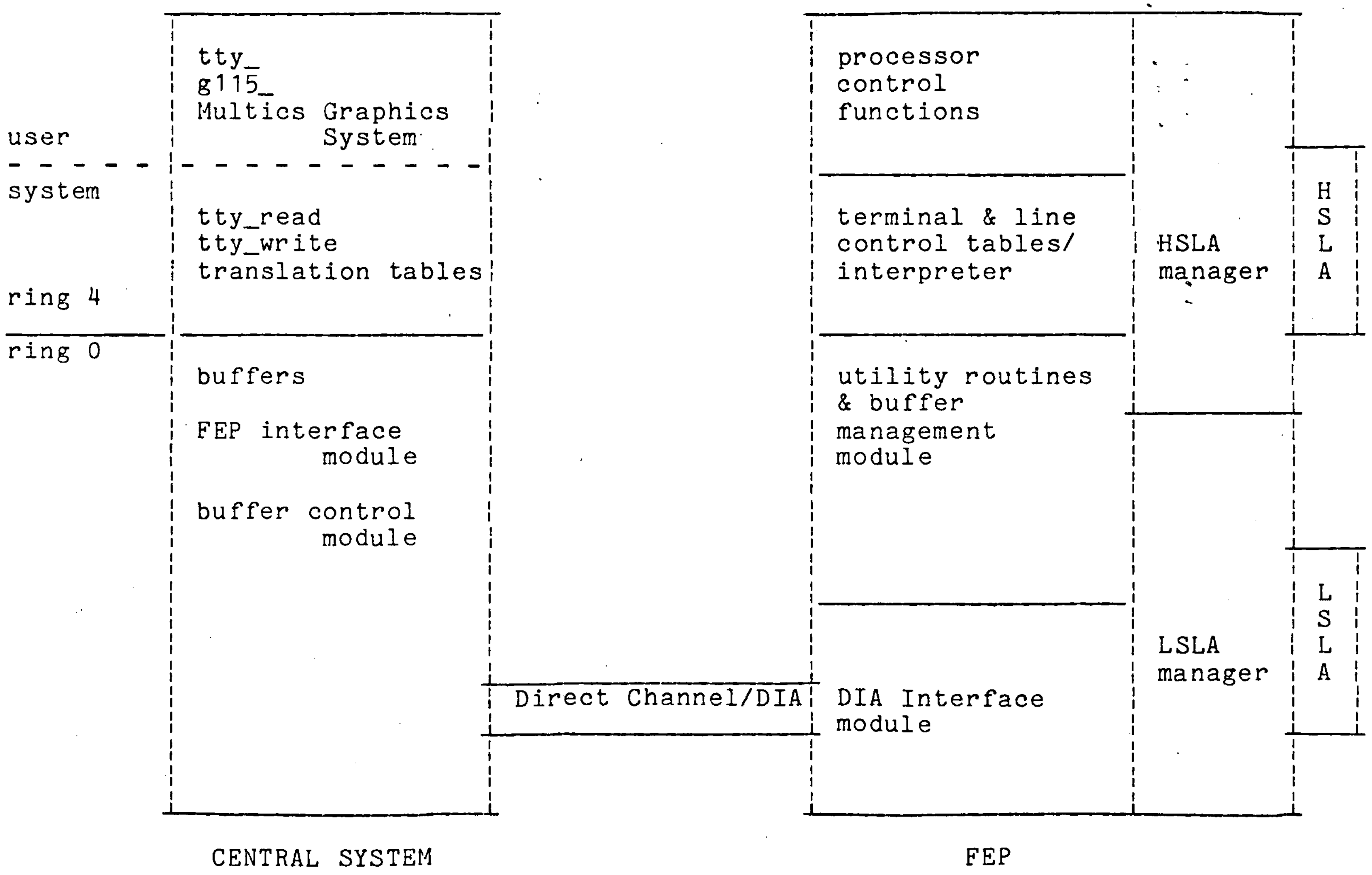
1. New Multics/FEP interface, less DIA I/O
2. Major redesign of FEP software
3. Table-driven line control, easily modified and expanded
4. New LSLA and HSLA device handlers; HSLA handler designed for high-speed input and output
5. Decreased hardware-error sensitivity
    a. DIA parity errors
    b. LSLA sync errors
    c. Reporting of all errors to operator
6. Better crash analysis and data

FEATURES PLANNED FOR RELEASES AFTER MR3.0

1. Improved code conversion tables, designed for use with EIS
2. Improved tty_read and tty_write to make full use of new code conversion tables
3. User-supplied code-conversion tables
4. User-specified erase and kill characters
5. User-specified delay-timing tables
6. G115 remote computer interface to be handled by FEP
7. Automatic baud rate detection for HSLAs at 110-600 baud
8. Improved handling of hangups by answering service, including optional reconnection to process
9. Fail-soft operation of FEP, i.e., FEP failures will not result in Multics crashes; hangups will be sent for all users of that FEP, and it will be dumped and rebooted
10. Improved handling of multiple terminals per process, with better detection of which terminal caused a QUIT, etc.

FEATURES UNDER CONSIDERATION

1. Support for bisync line protocol, and terminals that use it, e.g., IBM 2780, IBM 3270, etc.
2. Support for EBCDIC code terminals at all baud rates, e.g., IBM 3767
3. Support for HDLC line protocol
4. Support for remote terminal concentrators, e.g., RNP700 and others
5. Support for "transparent data" mode where 8-bit bytes are sent and received without parity
6. Online T&D for FEP and communications channels


user
system
ring 4
ring 0
tty_
g115_
Multics Graphics System
tty_read
tty_write
translation tables
buffers
FEP interface module
buffer control module
Direct Channel/DIA
processor control functions
terminal & line control tables/ interpreter
utility routines & buffer management module
DIA Interface module
HSLA manager
LSLA manager
HSLA
LSLA
CENTRAL SYSTEM
FEP

MCS DESIGN

There were several explicit functions performed by the software under the GIOC simulation. While these functions still exist in MCS, they are performed differently. The FEP still manages control of its communications adaptors: the Low Speed Line Adaptor (LSLA) and High Speed Line Adaptor (HSLA). In addition, it now contains the software necessary to drive each of the types of terminals that can be connected to Multics. This function was previously performed in the central system by tty_inter and its associated set of terminal control tables, tty_ctl.

Another functional entity is buffer control within the central system. This includes input buffering of type-ahead data and output buffering of write-behind data. Previously, a fixed size (at bootload time) buffer pool, consisting of several small buffers, was used. These buffers were usually filled with data and the necessary GIOC DCWs to transmit the data or receive it into the buffer. While the buffering function stays in the central system and in ring 0, the strategy for performing the I/O buffering has been changed. For example, the buffers only contain data and not DCWs. Furthermore, the previous strategy of allocating a new small input buffer while the current small input buffer was being filled by a transmitting terminal has been changed since it precluded the use of high-speed terminals on Multics. (These terminals transmitted faster than the software could answer interrupts and allocate new buffers.) This type of device buffering has been moved to the FEP. Buffering between the FEP and the central system consists of buffers of data. Provisions have been made for the case when either machine is unable to accept a buffer of data.

Another function that was altered is output data formatting, which was previously handled in ring 0 by tty_write. This function has been moved out of ring 0 with only a few minor changes. In a similar vein, the formatting and canonicalization of input data previously performed by tty_read and tty_con has been moved out of ring 0, again with only a few minor changes. Basically, these small changes consisted of the removal of DCWs from buffers and the addition of non-ring 0 device-code translation tables that can be easily changed by the individual user process.

Finally, the large block of code in both tty_read and tty_write concerned with the ARDS graphics capabilities has been totally removed from the standard central system software and replaced by the new Multics Graphics System.

NEW CENTRAL SYSTEM SOFTWARE

As indicated earlier, the central system software no longer concerns itself with terminal control, but instead has responsibility only for the conversion of data from a user's process into device-specific code and for the inverse conversion of data received from the various terminals. The central system software is no longer exclusively in the supervisor ring, graphics is no longer incorporated into the main flow, and various new features have been added. In spite of all this, the user interface remains essentially unchanged.

Some of the new features are briefly described below:

1) A control function to change erase, escape, and kill characters.

2) A control function to change the break character (currently the newline character - 012). A break character is defined as a character that sends a wakeup to a process that is blocked on input, limits the effect of kill and canonicalization, and limits a read. The user is allowed to specify a new break character or a list of break characters. Users may also specify that the input line should be terminated one character after the specified break character. (This is for devices that send an end of data character like EOT or ETX followed immediately by a longitudinal parity check character.) Finally, users are

allowed to specify breaking on every character (for full duplex echoing from a process) or no breaking at all (for devices that are not really typewriters but perhaps small computers that plan to send data in 8-bit bytes).

3) An expanded "set_table" control function that allows users to really supply their own translation tables in the user ring. These translation tables have approximately the same format as before, but they correctly translate any output from Multics that is played back as input. This means that line-control characters (EOA, STX, etc.) and tab or carriage return delay characters are properly stripped off on input.

4) A control function to allow users to change their terminal type (also a login line option as described later). In order to describe the meaning of this control function, it must be pointed out that there are two notions of "terminal type." From the point of view of the FEP, terminals are divided not into types but into "device classes" based upon the fact that special control functions are required for that device class. Hence, all slow-speed ASCII terminals (e.g., GE TermiNet 300, Teletype Models 37, 38, and 35) currently supported by Multics are in one device class. Devices resembling an IBM 2741 are in another device class. From the Multics view however, terminals are divided into types based upon different criteria. Thus, a Trendata 1000, Dura 1021, IBM 2741 can all be considered different types.

The various terminal types are installation definable. It is this terminal type rather than the device class recognized by the FEP that the user specifies in the control call.

5) A control function that allows a user to switch his terminal into what might be called "no-control mode." This mode is provided for the user who wants the FEP to do no line control or device control whatsoever. For example, assume that a user has dialed into a 134.5-baud port on a minicomputer whose line-control procedures are different enough from an IBM 2741 that the standard line-control procedures employed by the FEP cause the device to behave incorrectly. The user could enter no-control mode to cause the FEP to stop any control of his terminal. The FEP acts only as a transmitter and receiver of characters. It will be up to the user´s process to do device control by sending special commands to the FEP.

6) A control function to change carriage return and tab delay formulae (the coefficients for the formulae are currently kept on a device type/baud rate basis).

7) A mode call that allows users to enter echoplex mode so that characters typed on the user´s terminal are echoed by the FEP rather than by local echoing at the terminal. Of course, this mode is only honored for those devices possessing full duplex capabilities.

8) A control function to place terminals into "polite mode." This means that output sent to a terminal is not transmitted by the FEP until the carriage is at the left margin. If a user in polite mode wants to send a message to his terminal regardless of carriage position, he must bracket the write call by control calls to reset and then set polite mode. While the user is in polite mode, a timer is set each time the carriage leaves the left margin. If the carriage does not return to the left margin within a specified time, any waiting output is sent to the terminal, regardless of current carriage position.

9) A control function that allows the user to "sense carriage position and lock keyboard." This control function requests the FEP to return (to the central system) any input from the terminal (even if the break character has not yet been typed) and to lock the keyboard if possible. This call allows the user to gather up any input, lock the keyboard, interrupt the input with an output message, and then "replay" the input on the terminal so the typist whose typing activity was interrupted is able to continue. (Of course, this call is not necessary when in echoplex mode.) Sense carriage position may also be use to compute how much room is left on the carriage for the the first line of output (i.e., the length of the first line of output equals the line length minus the number of characters input).

10) A new mode, lfecho, that causes the FEP to echo a line-feed character upon receipt of a carriage-return character.

11) A mode, tabecho, that causes the FEP to echo spaces upon receipt of a tab character. This mode is useful for terminals that have no hardware tab capability.

12) A mode call to allow users to specify that keyboard locking is to take place prior to sending output. The default for this mode is no locking for ASCII-type terminals and locking for IBM-type terminals. This allows users of ASCII-type terminals that do have a locking feature to turn on this facility.

Foreign terminals present some problems on Multics. Since Multics only recognizes a limited number of terminal types, it is desirable for the user to be able to specify as soon as possible after dialing up that he has a particular terminal type. If a terminal is so different that the user cannot even go through the standard handshaking procedure with Multics, he is out of luck. But if he can manage to get the answering service to issue the login prompting lines, a new login line argument is provided to allow him to specify his terminal type (e.g., login Grady -tt TN300).

Another potential problem is the initializer typing messages on a user's terminal after the user has switched translation tables (e.g., the user may have switched tables when he changed his typing element). When the initializer types a message on the user´s terminal, the message will be garbage since the initializer cannot know about the translation table change. To solve this problem, the new send_message facility is used and the message is typed by the user´s process using the correct translation tables. If the user does not have the send_message segment, the initializer is forced to send a message to the terminal using a default translation table.

Another issue is the specification of terminal types by the answering service. Currently there is code in the answering service to declare a terminal to be a particular type based upon its answerback coding (which is read by the answering service rather than by ring 0 as with the former ttydim). This makes it difficult to add new terminal types at the local installation. To solve this problem and to allow easy addition of new terminal types, the terminal identification procedure and specification of the various characteristics for each terminal type is tabularized within the answering service. There are two tables: the first table specifies the way in which the initial terminal type is to be set up; the other table specifies the device characteristics for each terminal type. These tables can be replaced while the system is running.

NEW FEP SOFTWARE

Each FEP attached to the central system has an area in main memory called a mailbox that it uses to communicate with the central system. This mailbox is divided into a header area followed by 16 submailboxes. Each submailbox is used to communicate information between the two machines. When the central system wants to send some type of command to the FEP, it selects a submailbox, places the command and any associated data in the submailbox, and then tells the FEP to read that submailbox by interrupting the FEP on one of 16 interrupt levels (the one corresponding to the submailbox number). Similarly, when the FEP wants to send some information to the central system, it requests the central system to select one of the 16 submailboxes. The central system will do so and then place a command in that submailbox that will indicate to the FEP that it may use this submailbox. Then the central system sends an interrupt whose level number corresponds to the number of the selected submailbox as described above. The FEP may then fill that submailbox with the information it wanted to send and write the submailbox back into the central system.

The FEP interface module is called from two sources. The user ring software calls it (through a gate) to pass several types of control function and mode calls on to the FEP for implementation. The buffer management programs call the interface module to cause output data to be sent to the FEP for transmission. Of course, the FEP calls it indirectly by generating interrupts to cause the interface module to look at a submailbox for the data passed with the "call."

The FEP software is functionally divided into four groups of modules dealing with: the FEP hardware (HSLA, LSLA, etc), the modules responsible for supervisory control of the FEP itself, actual terminal control, and utility functions.

There are three hardware managers: one for the HSLAs, one for the LSLAs, and one for the Direct Interface Adaptor (DIA). The HSLA and LSLA managers are responsible for assembling messages to be passed to the terminal control modules and for transmitting messages from the terminal control modules. The DIA module is responsible for supporting the FEP side of the FEP/central system interface as described above.

The terminal control modules consist of a macro-built table, for device control state/transition data similar to the tty_ctl module in the former ttydim, and a program to "execute" (interpret) these tables. Terminal control was performed by tty_inter in the previous ttydim. The FEP device control tables contain control information for the following seven device classes:

ASCII	Teletype Models 37, 38 and GE TermiNet 300
1050	IBM 1050
2741	IBM 2741, Trendata 1000
NO-CONTROL	as discussed previously
ARDS	ARDS, Tektronix

SYNC	Mohawk 2400
BISYNC	Various IBM devices

The table-interpreting program also contains the algorithms for determining the device speed of terminals dialing up on asynchronous HSLA lines at up to 600 baud and for the dynamic configuring of these HSLA subchannels to match the baud rate of the incoming terminal.

The supervisory control modules are responsible for scheduling program executions, dispatching interrupts, managing buffers, and the scheduling of delayed (timer initiated) program executions.

The utility modules take care of post-bootload initialization, use of the attached FEP control console, dumping the FEP, printer tracing of system activity, etc.

The FEP was previously bootloaded by a BOS program prior to every Multics bootload. This created two problems. First, bootloading the FEP and its subsequent initialization caused all lines to be hung up. Second, every time the FEP software changed, a new BOS tape had to be created. Both of the problems are solved by having the FEP software on the Multics tape and bootloading the FEP from Multics during a regular bootload. However, before bootloading the FEP, Multics interrogates it to see if it is up; if up, the FEP purges any software queues and sends a message to the central system indicating that it is ready to run. (Of course it is always possible to force a FEP bootload even though the FEP claims it is up.) The FEP will also be informed when Multics crashes to prevent the FEP from trying to talk to Multics. It is possible for the Multics operator to load the FEP with a precanned message, in all the popular device codes, which the FEP sends to any terminal dialing up. The message could be enough to inform the user that Multics is currently unavailable.

Another operational issue is FEP crashes. In the event of an FEP failure, all users dialed in through that FEP when it crashes are logged out, the FEP is rebooted, and dialups are allowed. This is especially desirable when there are multiple FEPs. If the initializer happens to be one of the consoles dialed up through that FEP, the message coordinator leaves all messages in a file until the initializer reconnects. Also, the FEP can be dumped before it is reloaded, so that analysis of the FEP failure can be performed.