Multics Task Report

MTR - 138

From: Robert S. Coren

To: Distribution

Date: 01/26/77

Subject: MCS for Multics Release 6.0

This MTR describes the modifications and extensions desired for the version of MCS to be included in Multics Release 6.0. An attempt has been made to estimate the amount of programmer time needed to accomplish the various tasks. Because the amount of time required to accomplish all these tasks exceeds the amount available, some of the modifications will have to be omitted from MR6.0. A future MTR will specify more accurately exactly what will be included.

PROPOSED EXTENSIONS

Installation-Defined Types

A mechanism will be provided to allow an installation to define its own set of terminal types and line types. The initial modes table will be replaced by a type table (probably part of the CDT) which will specify the names and characteristics of the terminal and line types supported. A standard version of this table should probably be supplied with the system, specifying a subset of the types supported by the Honeywell-supplied software.

For terminal types, the table will specify such things as default conversion, translation, and delay tables, initial modes, permissible modes and orders, tab-setting strings, and compatible line types. The various default tables, currently bound into bound_tty_active, should probably be moved into the user ring; a mechanism will be provided to generate this segment (possibly using cds) and binding the addresses of the tables to terminal types specified in the CDT. Obviously, different types can share one or more tables.

For line types, the table will specify such things as compatible terminal types (and possibly default terminal type), possible baud rates, options and attributes that affect FNP behavior, permissible modes and orders, possible modems, and whether the relevant protocol is synchronous or asynchronous. The existence of a non-standard line type implies that a site is providing either a new control tables module in the FNP or modifications to an existing one. It will be necessary to provide a reasonably easy way to specify what control tables to use for a given line type. The method of converting attributes of a channel described in the CDT into appropriate flag settings, etc., in the FNP will have to be generalized; at present it is difficult to add a line type without modifying control_tables and init in the FNP, and possibly load_fnp_ as well.

Some of the code in both ring zero and the FNP contains assumptions about possible terminal and line types; obviously these will have to be eliminated, and all decisions based on types be made table-driven. I assume that the definition of new terminal types will be more widely used than that of new line types; anyone providing new control tables has to have a more intimate understanding of the workings of MCS than someone who simply wants all the users at his site to have the correct combination of delay timings and conversion tables by default.

When the various mechanisms associated with the feature have been designed in more detail, an MTB will be published describing the proposed format of the new tables and addressing in more detail the issues referred to above.

New System-Defined Types

CISL will probably have to provide support for the IBM 2780. This will entail a new set of control tables, or a modification of the current bisync tables, to allow alternation between transparent and non-transparent transmission. A 2780 I/O module will also be required to format messages and handle device control.

The generalization of terminal and line type support required to allow installation-defined types will also make it easier for us to add new types to the set supported by the standard system.

More Control Over Terminals

The user should be given more control over various aspects of terminal behavior. Some planned improvements in this area are mentioned here.

OUTPUT OPTIMIZATION

We would like to improve the tab-setting mechanism to take into account a terminal's line length. In addition, some users have requested a method for telling the system that hardware tabs are somewhere other than every 10 columns, and tab echo does not work well on terminals that advance some fixed number of spaces (such as 1 on the Tektronix) when a tab is typed.

In this connection, it has also been suggested that delays be taken into account in determining whether to use tabs or blanks to advance the carriage a small number of columns. Some generalization of the delay mechanism is also desirable, possibly involving the addition of more delay values. Finally, we may decide to "decanonicalize" overstruck output for increased efficiency, if some reasonable way to determine the most "efficient" sequence of characters can be found.

BREAK CHARACTERS

It would be nice to allow users to modify their input break characters. Unfortunately, this term means different things at different levels of the system: the FNP recognizes "transmission" characters, ring zero recognizes "wakeup" characters and "end-of-line" characters, and tty_ doesn't know about anything except newline. We will attempt to clarify the break character concept and at least give the user some control over one or more of these functions. To alter the transmission character in the FNP we will have to devise a way of constructing HSLA CCTs on the fly, and maintaining reference counts on the tables thus constructed.

LINE CONTROL

We will add a pair of features that allows information to be passed between FNP control tables and user programs without any of the intermediate levels of software having to understand its content. An "alter parameters" order will allow a user to send orders directly to the control tables without interpretation by tty_index or dia_man; a generalized "signal" from the control tables will be interpreted by dn355 as a request to interrupt the process (whether we will invent an ips signal type or restrict this mechanism to blocked processes remains to be decided).

Deconcentration

We will supply a demultiplexing facility for channels from a remote concentrator. The intention is that the user-ring and outer-level ring-zero software perceive each subchannel exactly as if it were a normal communications channel with a line of its own, and the FNP perceive the concentrator as a single channel; dn355 must recognize concentrator channels and call special routines to perform the mapping between the concentrator channel and the subchannels.

Immediate candidates for a concentrator interface are TymNet and TeleNet. We have had discussions with Mike Ford of TeleNet, and have determined that a new line protocol is required: bisync without acknowledgement. We may be able to modify the existing bisync tables to make acknowledgement optional.

Poll and Select

Currently existing support for the VIP protocol will be extended to support multiple terminals via poll and select. We might want to implement deconcentraion in ring zero for a multiple-terminal VIP channel. Since the VIP "terminals" may include a printer and a tape cassette reader as well as a keyboard/display, if we choose to support these additional devices we will probably need a user-ring I/O module to manage them.

Validation Level

GM has requested the ability to associate the user's validation level with a channel at attach time, so that the channel cannot be used by an outer ring. An order would be provided to "promote" the channel's validation level if an inner ring wishes to make the channel available to an outer ring.

Modem Management

The advent of new Bell modems in the 202 series has highlighted the lack of generality in the control tables' use of dataset leads. In particular, various modems have different responses to the request-to-send lead, and present carrier and clear-to-send under a variety of circumstances. It is possible that we should have a way of specifying some details of modem management within a line type. In particular, it might be advisable to implement a half-duplex protocol that would be generalization of the current 202C6 protocols (ARDS and TN300).

INTERNAL IMPROVEMENTS

Error Reporting

There is currently no reliable mechanism for reporting configuration errors detected during FNP initialization; moreover, errors resulting from an attempt to attach an unconfigured or improperly configured channel are not easy to interpret. Better error-reporting mechanisms for both these situations will be included in the modifications required for installation-defined types.

Buffer Space in the CS

The management of tty_buf requires some extension and generalization. The inability to allocate blocks of any size other than 16 words has to go: for one thing, if we are to support line speeds greater than 9600 baud, larger data buffers will be needed; for another, the addition of new features (both proposed and as yet unimagined) will require additional fields in

the dynamically-allocated CTL, which is presently filled to capacity (except for a dangerously small number of mode and flag bits); finally, the limit of 16 output DCWs in a block unacceptably restricts the size of an output message.

It would also be useful to allocate FCTLs -- the blocks describing all defined channels -- dynamically, so that new channels could be made available simply by installing a new CDT and reloading the FNP, rather than having to wait until the next Multics bootload as at present. On the other hand, the FCTL address must be readily derivable from the line number (used by the FNP) and the device index (used by tty_). The best solution may be to keep a fixed-size (or installation-settable) array of relative pointers in the header of tty_buf (where the FCTLs themselves are kept now).

What allocation scheme is most suitable has not been determined. The most likely choices are: 1) treat the dynamic portion of tty_buf as an area, or 2) modify tty_free to be able to allocate blocks in multiples of some fixed size (probably 8 or 16) rather than one size only. The second solution requires the introduction of a free-block-combining mechanism such as presently exists in the FNP.

In any case, the new buffering scheme will almost certainly require recompiling all the programs in ring-zero MCS.

Buffer Space in the FNP

Using larger buffer sizes for faster channels obviously affects the FNP as well as the CS. Since the FNP buffer allocation mechanism already allows the allocation of blocks of any multiple of 32 18-bit words (up to 1024), it would be easiest to continue to restrict the size of data buffers to be multiples of 32 words (16 words in the CS). A buffer size could be assigned to each channel at dialup time.

TIBs will probably have to be more than 32 words sometime soon, but need not be as large as 64. This does not present a problem, as TIBs are allocated during FNP initialization. They could be 48 words long, as HSLA software communications regions already are.

The FNP must be made less liable to complete exhaustion of buffer space. Certain applications (notably per-channel input and output buffers) must be prevented from taking the last buffer (by imposing a threshold of, say, 1024 words) so that there will always be buffers for those modules that need them immediately (e. g., the scheduler and the LSLA interrupt handler). These non-critical applications would not crash the FNP when they could not get enough buffers.

An associated idea is the limit imposed on the size of an input chain. This should be made dynamic (say, a function of the available space as in the CS), rather than shutting off the channel when it reaches an absolute limit (10 buffers per channel in the present implementation).

Input Processing

This may be the time to eliminate the common circular buffer in tty_buf used for all input, and implement a DCW mechanism to copy input data from the FNP directly into allocated buffers.

Other Improvements

Some new op blocks may be added to make control table writing easier; some FNP debugging aids are also planned. It has been suggested that the line speed in characters per second be reported along with (or instead of) the baud rate, and be used for delay calculations. The implementation of these features must depend on the amount of time available.

TIME ESTIMATES

The remainder of this document is a specification of the tasks necessitated by the modifications described above, along with estimates of the time required to accomplish each task. Those items which seem likely candidates for elimination or postponement are marked with asterisks. All times are given in programmer-weeks.

Installation-defined Types		
Design CDT changes and write MTB	8	
Design modifications to load_fno_, FNP	2	
Desigr new cefault tables addressing	1	
Code additiors tc cv_cmf	3	
Modify loac_fno_, FNP	2	
Modify tty_index for new table stuff	1	
Debuggirg	10	
Total		27
Space Management -- CS		
Design	3	
Coding	5	
Debugging	1	
Total		9
Space Management -- FNP		
Design TIB extensions	3	
Design buffer threshold mechanism	1	
Coding	4	
Debuggirg	2	
Total		10
*Deconcentration		
Design new data bases, dn355 mechanism, MTB?	6	
Design & code specific deconcentration routines	7	

Add no-ack option to bisync	1	
Debugging	10	
Total		24

User Signal and "Alter Parameters" Order

Desigr	2	
Code	1	
Debugging	1	
Total		4

*Break Characters

Design, write MTB	5	
Code (includes new CCT mechanism)	6	
Debugging	3	
Total		14

2780s

Design control table modifications	2	
Code new control tables	2	
2780 I/O module	3	
Debugging	5	
Total		12

*Poll and Select

Design control tables modificatiors	3	
Code control tables modificatiors	2	
Debugging	2	
Total		7

*Output Optimization		
Design & code	3	
Debugging	1	
Total	--	4
*Modem Management (Most of this is included in CDT design)		
Additional code in control tables	2	
Total	--	2
Other		
Validation Level	1	
*Input processing	1	
Miscellaneous (overhead)	4	
Total	--	6

Grand Total		119
Grand Total (minimum)		67