To: Distribution

From: James R. Davis

Date: 03/02/81

Subject: Further Work for Multics Video Support

1. INTRODUCTION

Existing video support on Multics was designed to meet the needs of the new menu system, and meets (or is close to meeting) them. But eventually the video system should be the default, used by everybody, without any explicit action. There is much that must be done (to it and to the rest of Multics) before this can happen. This MTB lists some of the work to be done. Where possible, I have given estimates for the amount of work required. The items appear in approximate order of (my) priority.

The reader should be familiar with the MTBs on the video system: 458, 461, and 462.

2. IMMEDIATE AND CONTINUING

In the short term, the task is to get the video system installed in a stable form.

The video system is not free from bugs. A bug list exists, and all known bugs are on it. Some of these bugs are documented features that have not been implemented. At this writing, the single largest missing feature is that MORE processing does not used the ring zero "marked io" synchronization facility.

The tty_ dim (and vtty_) need a way to inform callers of the state of the terminal print function (i.e. local echo, set by the printer_on and printer_off orders). Emacs always turns the printer on (local echo) when done, because it cannot tell what state the printer was in when it entered. This is not the right

default for the video system. Rather than guess, Emacs should be able to ask the I/O system what state the printer is in. This has already lead to many complaints. It should not take more than a week to change.

An info file should be created from the MPM Comm I/O writeup telling how to add terminals to the video TTF. This will increase the exposure of the video system. Many people will try the video system if their terminal is defined for it. They shouldn't have to wait for the book to be published, or have to ask the maintainer. This should take less than a day.

3. NEAR TERM

The performance and efficiency of the video system should be studied. Other than making all internal blocks quick, I have done little for efficiency. If the video system is to become the standard for Multics, it must become highly efficient. One week should give some improvement, and I suspect a months effort would not be wasted.

Multics request loops (the command listener, the subsystem listener) should be changed to issue a "reset_more" control order just before printing their ready (or promting) messages. This allows the user to discard output without losing the (desirable) prompt. (The error code should be ignored, since the tty_ module doesn't understand this order.) This work should take a week.

4. IMPROVED TERMINAL SUPPORT

Terminal support of the video system should be extended. There are several aspects to this. The problems described here do not have easy solutions. I do not know what priority should be set for them.

First, the TTF representation for command sequences is not flexible enough to describe all encoding schemes used in terminals. In general only a procedural representation will be flexible enough. I have no design for this. It is at least a two person-month project. In the meantime, the video system could be extended to allow the use of user-written terminal drivers (analogous to Emacs CTLs), so the terminal could be used without the TTF encoding.

Second, there are some desirable terminal features we cannot access because we have no model for them. These include window operations (as on the Concept 100) and scroll operations (as on the DEC VT-100). In most cases a terminal can be used even if not all its features are supported although less efficiently.

It is also desirable to be able to use the video system on a printing terminal. This requires changes to the video system only. For the most part a printing terminal is a subset of a video terminal, so the work involved is checking for absence of features.

5. LONG TERM

In my opinion, the work in this section is more important than adding support for more terminals.

The video system should be consolidated with the rest of Multics Communications. There are several aspects to this. The Terminal Control layer (TC, implemented by vtty_) should be merged with the Typewriter DIM (tty_). Currently the video system calls past the tty_ module, directly to ring zero. This is a violation of modularity.

The video system uses the tty_ modules (and ring zero) in raw mode. There is no need for the conversion and canonicalization functions of MCS if implemented by the video system. They should be removed from ring zero.

Emacs should use the video system entirely, where possible. (Currently Emacs runs under the video system using a CTL that calls the window operations.) Until the video system supports all missing terminals, the Emacs system of CTLs will be retained. Until the video system supports the ARPANET (see below), Emacs will continue to make hardcore calls itself. But in the normal case, Emacs should be able to avoid hardcore calls itself. It is important to do this, because there must be only one caller of the ring zero echo negotiate entries per process if asynchronous output is to be managed. The reason for this involves the complicated protocol inherent in an echo negotiated scheme.

The message facility must be changed such that a caller can save and restore all message handling modes. This would allow Emacs to avoid undesired output. Without this, Emacs and the video system will be fighting each other for control of messages.

The cursor positioning algorithm used by vtty_ is not

optimal. To improve it requires a function to compute the costs of various possible cursor positioning sequences.

6. DISTANT FUTURE

The first video system MTB listed some unsolved or ill-defined issues for video systems. These issues are still ill-defined, and still need to be addressed. These problems are at the leading edge of window video systems, the subject of research and experimentation at laboratories across the world. It will take a serious effort to solve some of them. They include:

Screen Images: A screen image is a record of the contents of a window or screen. Keeping a screen image makes possible some kinds of output optimizations, emulation of missing features, and allows recovery from unexpected screen garbling.

Redisplay: It is desirable for programs to be able to modify displays by simply providing the video system with a desired screen/window image, and letting the video system worry about what changes to the existing screen are necessary. Redisplay can provide significant cuts in time to put up a desired image. Redisplay requires a screen image.

Middle of Line Editing: The current input editor can only effect changes at the end of the line. Given redisplay, it should be possible to move the cursor into the middle (or beginning) of the input line, for more powerful editing.

Pieces of Paper: It is also desirable for program to not have to worry about fitting their output in a limited window. Pieces of Paper allow programs to output to a simulated large buffer, some portion of which may be on display in one (or more?) windows at a time. The program/user can scroll the window over the entire buffer, looking at earlier output. Pieces of Paper require redisplay.

Desk Management: Desk Management involves keeping track of the various windows (and pieces of paper) visible and invisible as they change. This is important because any change to the size or location of windows probably affects adjoining windows. If one grows, another must shrink. The system should do this for the user/program.

Extended Echo Negotiation: The existing protocol provides

for rubout processing by ring zero (or the FNP?). This protocol should be implemented (if it will be a significant performance improvement). The protocol could also be extended to support middle of the line text insertion. This would help Emacs (if changed to use it), and might help the video system if it implemented middle of the line editing.

Echo Ahead: An unsolved problem of video is how to echo characters typed by the user before any program has read them. There is no way to tell what window they will ultimately go to, or even how they should be echoed. One strategy to be investigated is to assume that the characters may be echoed in the current window, and be able to "unecho" them should that assumption prove false.

The video system should work over the ARPANET, support the SUPDUP-OUTPUT TELNET option. A User SUPDUP program should be written, allowing Multics users to use video programs runnning on remote hosts. Ideally, the Answering Service should support SUPDUP logins, as well.

The video system should support visual attributes terminal features such as inverse video, low intensity. This allows a terminal independent forms package to be built on top of the video system.

One interesting suggestion (due to Carl Hoffman) is to provide descriptions of terminal keyboards in the TTF, giving the legend and location of such keys as the delete key, the control key, the break key. This would allow on-line documentation to say "hold down the control key (labelled CTRL and located to the left of the CAPS LOCK key)". A character string entry could describe how the key is labelled ("CTRL") and where to find it on the keyboard ("to the left of the CAPS LOCK key"). This would be particularly useful for consoles like the IBM3101, where the control key is labelled "ALT" and located right of the space bar. This could be useful for terminals which use unusual sequences of key strikes to generate characters. For example, on a VT-100, typing ^@ does not generate a ^@.

No doubt before all or even any of these issues are addressed other ones now unanticipated will arise.