PERMANENT MEMORANDUM NO. M-1101

SUBJECT: Typewriter Control (TYC) DATE: September, 1961

TO: PDP Distribution List FROM: Bolt, Beranek and Newman, Inc.

Purpose:

The purpose of TYC is to help in debugging other programs and to allow the control of other programs from the on-line typewriter. TYC can be commanded from the typewriter to do the following: Type-out the contents of selected registers, change the contents of selected registers, execute instructions (including jmp instructions which give control to other routines), list those registers having given contents in a given field and define new TYC commands.

Properties:

TYC has been programmed in both the DECAL and FRAP Systems and is, therefore, relocatable. The DECAL version occupies 415 octal registers and is identical to the FRAP version. If new TYC commands are to be defined, additional space should be left at the end for the definition table to grow: two words per command defined are required.

There are four usable subroutines within TYC which can be called by any program. Calling sequences and properties of these will be discussed later.

Usage:

TYC becomes operative after starting at the TYC origin. After some preliminary housekeeping, PDP-1 enters a waiting loop and waits for a typewriter key to be struck.

Typewriter functions used to type in commands are abbreviated as follows:

Space	s/p
Upper case	u/c
Lower Case	l/c
Tab	t/b
Carriage return	c/r

The commands given below can be executed by TYC with results as indicated. Commands must be typed exactly as noted. Any time the operator types "carriage return", TYC abandons any incomplete command and returns to the waiting loop. TYC returns to the waiting loop after executing each command below:

TYC Commands

1. Set Address (code a). Type

$$c/ra x_1 x_2 x_3 x_4 s/p$$

where $x_1x_2x_3x_4$ are 4 octal digits representing an address in which the operator is currently interested in interrogating. Subsequent TYC commands refer to this address. (e.g., a 1732.)

2. What Address (code w). Type w. This causes TYC to type-out what address is of current interest to the operator. After typing w, a carriage return is executed followed by the address type-out followed by a tab.

3. Increment Address (code i). Type i. This causes TYC to increment the address of interest by 1. After typing i, a carriage return is executed followed by the incremented address type-out followed by a tab.

4. Type Contents of Address (code t). Type t. This causes TYC to type-out the contents of the address of interest in octal and return to the waiting loop. Leading zeros are suppressed (but not spaced-out) and if the entire contents is zero, a single zero is typed. If Sense Switch 1 is up, the program will list the contents of successive registers, each on a separate line preceded by the register address until the switch is turned off (down). The address of interest is automatically incremented in the process.

5. Change Contents of Address (code c). Type

   $cx_1x_2x_3x_4x_5x_6{}^{s}/p$

   where the x's are octal digits. Any number of octal digits from one to six followed by a "space" may be typed. Leading zeros need not be typed.

   This causes the contents of the address of interest to be changed to $x_1x_2x_3x_4x_5x_6$. If Sense Switch 1 is up, the program will, after making the requested change, increment the address, carriage return, type the incremented address, tab and expect 1 to 6 octal digits to change the next register, etc.

6. Find Registers Having Given Contents (code f). Type

   $fx_1x_2x_3x_4{}^{s}/py_1y_2y_3y_4y_5y_6{}^{s}/p$

   TYC will start at register $x_1x_2x_3x_4$ and examine the contents of succissive registers until register 7777 is reached, then return to the waiting loop for the next command.

   Each register will be "anded" with the mask in register (origin $+263_8$) and compared with $y_1y_2y_3y_4y_5y_6$. Each time the computer finds a register whose contents agree in the masked positions with $y_1y_2y_3y_4y_5y_6$ it carriage returns, types the address, tabs, and types the contents of the register. At the end of the listing, the computer types "c" followed by the number (in octal) of registers found (count). If Sense Switch 2 is up, the computer suppresses typing the contents of the registers. If Sense Switch 1 is up, the computer types only the number of registers found.

7. Execute Given Instruction (code e). Type

   $ex_1x_2x_3x_4x_5x_6{}^{s}/p$

   The computer then executes the number $x_1x_2x_3x_4x_5x_6$ as an instruction.

When the program is entered at its origin, the AC and IO registers are saved in registers 261 and 262 respectively (octal). When the program is waiting for a character to be typed, these numbers are restored to the AC and IO and so are visible in the console lights. When an e command is given, the AC and IO are restored before the instruction entered is executed in register 21_8, and if the instruction is not a jump, the new values of the AC and IO resulting from the execution of the instruction are returned to 261_8 and 262_8 and displayed when the computer returns to the waiting loop. If the instruction was a skip instruction and a skip occurred, the letter s is typed.

Of course, the execution of a jump instruction usually causes TYC to lose control. TYC does not use any of the sense flags or other indicators except for Sense Flag 1 to detect typed-in characters.

If a jump is executed to another program and it is desired that this program should return control to TYC, it should jump to the TYC origin if the values of the AC and IO it produces are to become the new "official" values. On the other hand, if it jumps to origin+2, TYC will use the values it had last.

If Sense Switch 1 is up, the TYC will stay in e mode after executing $x_1x_2x_3x_4x_5x_6$ and expect another e and 6 digits to be typed and executed, etc., until the switch is lowered.

8. Define Given Instruction (code d). Type

$$d\gamma x_1x_2x_3x_4x_5x_6$$

where γ is a single character and the x's are octal digits. This causes a new TYC command to be defined as the execution of $x_1x_2x_3x_4x_5x_6$ as an instruction. γ must not be one of the letters a, w, i, t, c, f, e, or d, or any letter previously used in a d operation; otherwise, the new definition will be ineffective. (e.g., dp730003 s/p will cause p to execute the TYO instruction).

The d feature has two main uses. First, if TYC is being used as a debugging aid, frequently occurring jumps and other instructions can be made into cliches and executed with less typing.

Second, TYC can be used to control other programs which require typewriter control. In order to make this convenient, we shall describe some of the subroutines of TYC that may be useful.

Subroutines:

getl, located at 171_8.

If a program executes jsp, getl the program will return with the next character typed in the right 8 bits of the IO unless this character is a carriage return. If a carriage return is typed, the program will return to the waiting loop.

getn, located at 153_8.

jsp getn with n<6 in the IO, this routine will make a computer word out of the next n octal digits typed, justifying them to the right and will return with this word in the IO. Characters other than octal digits will cause errors except that carriage return will return control to TYC.

type, located 206_8.

jsp type causes the word in the IO to be typed in octal. Leading 0's are suppressed. tcr located at 145_8 and tab located at 151_8 cause carriage return and tab respectively to be typed.

The table of commands starts at 301_8 and each entry consists of a character in one register followed by a computer instruction (usually a jump in the other).

NOTE: TYC has been modified to include the following new features:

1. Typing (xα-β-γ-)

   where α, β, and γ denote octal numbers, sets the parameters

   beg = C(1540) = α
   end = C(1541) = β
   new = C(1542) = γ

2. Typing "y" then puts the contents of the registers from α to β inclusive into the registers γ to γ+β-α.

3. Typing "z" causes the contents of the registers from α to β inclusive to be compared to the contents of the registers from γ to γ+β-α inclusive and any registers in α to β which differ are printed out.

   The purpose of the feature is to enable comparison of the contents of a block before a program has acted with subsequent contents.