DN8x-Series Remote Station Macros

September 1980

The DN8x-series remote station macros manual describes an extra option available to the customer who wishes to modify the DN8x system software.

TOPS–10 7.01 Monitor

MACDLX V27A(667)
PAL10 V142A(144)

digital equipment corporation • marlboro, massachusetts

First Printing, October 1980

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DECnet	IAS
DECUS	DECsystem-10	MASSBUS
Digital Logo	DECSYSTEM-20	PDT
PDP	DECwriter	RSTS
UNIBUS	DIBOL	RSX
VAX	EduSystem	VMS
		VT

CONTENTS

FIGURES

PREFACE

The DN8x-series remote station macros are an extra option available to the customer who wishes to modify the DN8x system software by adding special functions, such as polled terminal-handling, screen-formatting, data-field-checking, and input consolidation. The option itself consists of a series of macro calls that allow programs added to the system to communicate with the DN8x system software.

1.0 PROGRAMMING CONVENTIONS

Tasks are installed by assembling them with the DN8x system. Any number of tasks can be included in a DN8x system, subject to the limitations of available core space.

The DN8x system is written with the assumption that any added code is correct. Therefore, programming errors in an added task can corrupt the system software. There is little protection of DN8x code from errant tasks.

In addition, added tasks must:

- not change the processor priority level.
- not modify the trap and device interrupt vectors of devices used by the DN8x system. (You can, however, access other attached devices at will.)

Macro arguments can be in registers, in core, or immediate, but the arguments can not be on a task's stack.

Macros perform their functions and return immediately to the calling task unless BLOCKING is specified. Macro execution does not modify the registers, memory, or stack, unless the results are returned in the registers or in memory.

2.0 TASK SCHEDULING

The DN8x system software provides four levels of task priority. These levels are numbered 0 through 3, with 0 being the highest priority. The priority level of a user task is specified by an argument in the TSKGEN macro call.

The DN8x system software maintains four task queues, one for each priority level. The tasks in each queue are run in a round robin. All tasks in a queue must be marked not runnable before tasks in a lower priority level queue are run.

The DN8x system software assigns each task a unit of run time when the task is scheduled for execution. The unit of run time is an assembly parameter whose name is TK.QTM. and whose default value is 12 clock ticks or two-tenths of a second. Once started a task runs to completion or until one of the following conditions occurs:

- A HIBER macro or SLEEP macro is executed.
- A PUT macro, a GET macro, or an IMGPUT macro is executed in BLOCKING mode.
- A RECEIVE macro is executed in BLOCKING mode when its message queue is empty.
- A user timeout occurs.
- An EXIT macro is executed.
- A higher priority task becomes executable.

When a task executes a HIBER macro, the task is marked not runnable and remains in that state until some I/O activity occurs for the task or until the time specified by the HIBER macro has elapsed.

When a task executes a SLEEP macro, the task is marked not runnable and remains in that state until the time specified by the SLEEP macro has elapsed.

When a task executes a PUT macro, a GET macro, or an IMGPUT macro in BLOCKING mode, the task is marked not runnable and remains in that state until the operation has completed.

When a task executes a RECEIVE macro in BLOCKING mode and the task message queue is empty, the task is marked not runnable. When another task executes a SEND macro addressed to the not runnable task, it is marked runnable, and will resume execution when it reaches the top of its priority queue.

If a task runs longer than its assigned run time, the task is marked not runnable and placed at the end of its priority queue. (This safeguard prevents loops in a user task from locking out other tasks.)

When a task executes an EXIT macro, the task is marked not runnable. The task will run again only if the DN8x system is reloaded or restarted, or if another task executes a TRIGER macro for the inactive task.

When the DN8x system is initialized, all tasks in the system are marked runnable, beginning at the starting addresses specified by their TSKGEN macros. Tasks that are started by a TRIGER macro call also begin at their starting addresses.

When tasks are restarted after a power failure or other system restart condition, the tasks begin at the restart addresses specified by their TSKGEN macros.

NOTE

The DN8X system software reclaims all assigned buffer space before performing the task restart. Tasks cannot, therefore, make references to buffer space assigned to them prior to restart.

3.0 CORE MANAGEMENT

The DN8x system software provides buffer management services for added tasks. These services are invoked with the CNKGET macro and CNKFRE macro, which dynamically allocate and deallocate buffer areas.

The buffers, called chunks, are fixed in length and are specified at assembly time by the system parameter CNKSIZ. Chunks not allocated to a task are stored on a linked list. This linked list is generated during the system initialization process and occupies all the free core space.

NOTE

The DN8x system software reclaims all assigned buffers before performing the task restart. Tasks cannot, therefore, make references to buffer space assigned to them prior to the system restart.

4.0 TASK CONTROL MACROS

4.1 TSKGEN Macro Call

Function:

The TSKGEN macro defines a specific task within the DN8x system and provides task scheduling information. The TSKGEN macro must be the first statement in a task. The TSKGEN macro is not an executable statement.

Command Format:

```
TSKGEN name, stadr, rstadr, priority, pdlsiz
```

name — A 3-character task name used as an argument in the SEND, WAKE, and TRIGER macro calls.

stadr — The task starting address, used when the DN8x system is first loaded and when the TRIGER macro references the task.

rstadr — The task restart address, used when the DN8x system is restarted after a power failure or other system restart condition.

priority — The task priority in the range 0 to 3, where 0 is the highest priority.

pdlsiz — The number of words (octal) the task requires on the stack.

Example:

```
TSKGEN FOO,FOOBEG,FOORST,3,20

          ;task name is FOO

          ;starting address is FOOBEG

          ;restart address is FOORST

          ;priority level is 3

          ;stack length is 20 (octal) words
```

4.2 EXIT Macro Call

Function:

The EXIT macro marks the task executing the call not runnable. The task can be marked runnable by a TRIGER macro call from another task or by a system restart.

Command Format:

```
EXIT
```

5.0 CORE MANAGEMENT MACROS

5.1 CNKGET Macro Call

Function:

The CNKGET macro call requests the DN8x system to allocate a chunk to the calling task. If a chunk is available, the chunk address is placed into the specified location and the Z bit in the PS word is cleared upon return to the calling task. If a chunk is not available, the specified location is cleared and the Z bit in the PS word is set upon return to the calling task.

Command Format:

```
CNKGET adr
```

adr The location where the chunk address is to be stored.

Example:

```
CNKGET R3  ;get a chunk and return its address
            ;in register 3
```

NOTE

The CNKGET macro does not clear the allocated buffer.

5.2 CNKFRE Macro Call

Function:

The CNKFRE macro call returns a specified chunk to the linked list controlled by the DN8x system.

Command Format:

```
CNKFRE adr
```

adr The location containing the address of the chunk to be deallocated.

Example:

```
CNKFRE FRED  ;deallocate the chunk whose address is stored at
              FRED
```

NOTE

The CNKFRE macro does not clear the deallocated buffer.

6.0 SCHEDULING MACROS

6.1 HIBER Macro Call

Function:

The HIBER macro call marks a task not runnable until a specified period of time elapses, until I/O activity occurs, or until another task issues a WAKE macro to the not runnable task. The optional argument, if present, specifies the elapsed time in clock ticks for the task to be marked not runnable.

Command Format:

```
HIBER arg
```

arg — (optional) The location containing the number of clock ticks to wait before the task is marked runnable.

Examples:

```
HIBTIM:3

HIBER HIBTIM  ;suspend for a maximum of 3 clock ticks

HIBER         ;suspend until I/O done
```

6.2 SLEEP Macro Call

Function:

The SLEEP macro marks a task not runnable for a specified time period.

Command Format:

```
SLEEP arg
```

arg — The location containing the number of clock ticks to elapse before the task is marked runnable.

Examples:

```
SLPTIM:3

SLEEP SLPTM  ;suspend the task execution for three clock ticks
```

6.3 WAKE Macro Call

Function:

The WAKE macro call marks runnable a task previously suspended by a HIBER macro call. If the task named by the WAKE macro call is not marked not runnable, the WAKE macro call has no effect.

Command Format:

```
WAKE name
```

name The name of the task to be marked runnable.

Example:

```
WAKE FOO  ;mark FOO runnable
```

6.4 TRIGER Macro Call

Function:

The TRIGER macro call marks runnable a task whose execution has been terminated by an EXIT macro call.

Command Format:

```
TRIGER name
```

name The name of the task to be marked runnable.

Example:

```
TRIGER ODT  ;run the task ODT
```

7.0 INTERTASK COMMUNICATION MACROS

The DN8x system provides intertask communication with the SEND and RECEIVE macro calls. These two macro calls access the message queue associated with each task. The length of the message queue is a system assembly parameter whose name is TKSQSZ and whose default value is 10 octal.

The SEND macro call places a 1-word message on the message queue of the specified task, and then returns to the calling task.

The RECEIVE macro call retrieves a 1-word message from the message queue of the task that executes the macro. Messages are retrieved from the queue in a first in, first out sequence.

A form of task synchronization is provided by the BLOCKING and NONBLOCKING options for the RECEIVE macro call. The RECEIVE BLOCKING macro call marks the task not runnable when the message queue is empty. The RECEIVE NONBLOCKING macro call proceeds with task execution regardless of the message queue status.

7.1 SEND Macro Call

Function:

The SEND macro call places a 1-word message into the message queue of the receiving task.

If the receiving task is hibernating or has previously executed a RECEIVE BLOCKING, the receiving task is marked runnable.

If the SEND macro call is successful, the N bit in the PS word is cleared. If the SEND macro call is unsuccessful, the N bit in the PS word is set. A SEND macro call is unsuccessful if the receiving message queue is full.

Command Format:

SEND name,adr

name — The name of the receiving task as specified by its TSKGEN macro.

adr — The location of the 1-word message that will be sent to the receiving task.

Examples:

```
SEND ODT,R0          ;send ODT the contents
                     ;of register 0

BMI DIE              ;go to DIE if SEND unsuccessful

SEND FOO,FOOBAR      ;send FOO the contents of FOOBAR

BMI DIE
```

7.2 RECEIVE Macro Call

Function:

The RECEIVE macro call extracts the first 1-word message from the task message queue. If a message is returned, the N bit in the PS word is cleared. If the message queue is empty when the RECEIVE NONBLOCKING macro call is executed, no message is returned and the N bit in the PS word is set. If the RECEIVE BLOCKING macro call is executed when the message queue is empty, the task is marked not runnable. The task remains not runnable until some other task puts a message into the message queue of the receiving task, which marks the task runnable.

Command Format:

```
RECEIVE adr,mode
```

adr — The address where the 1-word message is placed.

mode — Either the word BLOCKING or the word NONBLOCKING.

Examples:

```
RECEIVE R4,NONBLOCKING    ;test for a message

BPL WIN                   ;go to WIN if a message
                          ;is received
```

8.0 CLOCK REFERENCE MACRO

8.1 TIMER Macro Call

Function:

The TIMER macro call places the 2-word system up-time, measured in clock ticks, into the specified address and address+2. Each clock tick represents 1/60 of a second (1/50 of a second for 50 hertz power). The clock ticks are derived from the DN8x system line frequency clock.

Command Format:

```
TIMER adr
```

adr — The destination for the 2-word system up-time.

Example:

```
TIMER TLOC                ;put the up-time
                          ;into TLOC and TLOC+2
```

9.0 TERMINAL I/O MACROS

The DN8x system includes five terminal I/O macros: OPEN, RELEASE, PUT, GET, and IMGPUT. These macro calls enable the user task to intercept and manipulate characters moving between the terminals and the network host.

The OPEN macro call reserves a terminal for the exclusive use of the calling task. When a task has exclusive use of a terminal as an input device, characters from the terminal go to the task rather than to the network host. When a task has exclusive use of a terminal as an output device, no other task can send characters to the terminal. Data paths with and without exclusive terminal use are illustrated in Figures 1 and 2.

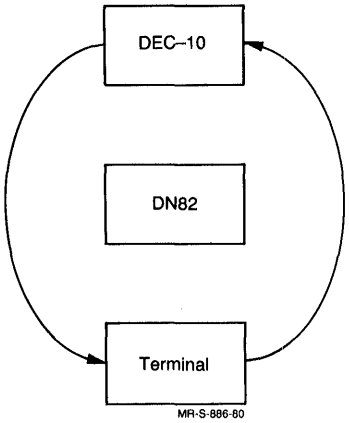

Figure 1 Data Paths without Exclusive Use of Terminal

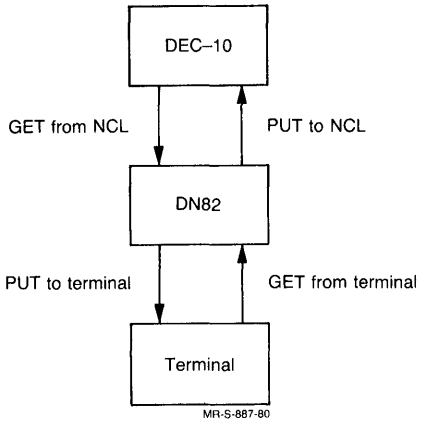

Figure 2 Data Paths with Exclusive Use of Terminal

The RELEASE macro call cancels the exclusive use of a terminal. If no task has exclusive use of a terminal, any task in the system can issue PUT and GET macro calls to the terminal. The CTY is usually left free so that any task in the system can send error messages.

The PUT macro call sends a single character to either the network host or to a terminal.

The GET macro call requests a single character from either the network host or from a terminal.

The IMGPUT macro call sends a character string to a terminal.

The PUT, GET, and IMGPUT macro calls can be executed as either BLOCKING or NONBLOCKING calls. BLOCKING macro calls mark the calling task not runnable until the I/O operation is complete. NONBLOCKING macro calls return immediately to the calling task.

9.1 OPEN Macro Call

Function:

The OPEN macro call reserves a terminal keyboard or printer for the exclusive use of the calling task. When a task has exclusive use of a terminal, no other task can access that terminal until the OPEN macro call has been canceled by a RELEASE macro call.

If the OPEN macro call is successful, the N bit in the PS word is cleared. If the OPEN macro call is unsuccessful, the N bit in the PS word is set.

Command Format:

```
OPEN line,dev
```

line — The octal physical line number of the terminal or -1, indicating the CTY.

dev — Either the word KEYBOARD or the word PRINTER.

Examples:

```
OPEN #2,PRINTER     ;get exclusive use of a TTY's printer

MOV #0,R5           ;load a line designator into register 5

OPEN R5,KEYBOARD    ;get exclusive use of the keyboard attached
                     to channel 0
```

NOTE

Two OPEN macro calls must be used to obtain exclusive use of both the keyboard and the printer on a given terminal: one specifying KEYBOARD and another specifying PRINTER.

9.2 RELEASE Macro Call

Function:

The RELEASE macro call cancels the exclusive use of a terminal keyboard or printer by a task.

If the RELEASE macro call is successful, the N bit in the PS word is cleared. If the RELEASE macro call is unsuccessful, the N bit in the PS word is set.

Command Format:

```
RELEASE line,dev
```

line — The octal physical line number of the terminal or -1, indicating the CTY.

dev — Either the word KEYBOARD or the word PRINTER.

Examples:

```
MOV #-1,R3              ;load the CTY designator

RELEASE R3,PRINTER      ;release the CTY printer

RELEASE R3,KEYBOARD     ;release the CTY keyboard
```

NOTE

Two RELEASE macro calls must be used to relinquish both the keyboard and the printer of a given terminal: one specifying KEYBOARD and another specifying PRINTER.

9.3 PUT Macro Call

Function:

The PUT macro call sends a single character to a terminal printer or to the network host. The terminal printer is specified by the argument PRINTER; the network host is specified by the argument KEYBOARD.

The PUT macro call can be either BLOCKING or NONBLOCKING. The BLOCKING call marks the calling task as not runnable until the operation is complete; the NONBLOCKING call returns immediately to the calling task.

If the PUT macro call is successful, the N bit in the PS word is cleared. If the PUT macro call is unsuccessful, the N bit in the PS word is set.

Command Format:

```
PUT line,dev,mode,adr
```

line — The physical channel number of the terminal or -1, indicating the CTY.

dev — The word PRINTER, indicating the terminal, or the word KEYBOARD, indicating the network host, as the destination.

mode — The word BLOCKING or the word NONBLOCKING.

adr — The address of the single character to be sent.

Examples:

```
PUT #-1,PRINTER,BLOCKING,IDLOC ;print the char in location

CLR R2                         ;IDLOC on the CTY

PUT #2,PRINTER,NONBLOCKING,R4  ;print the char in reg 4 on line 0

BMI LOST                       ;go to LOST if unsuccessful
```

9.4 GET Macro Call

Function:

The GET macro requests a single character either from a terminal or from the network host. The terminal is specified by the argument KEYBOARD; the network host is specified by the argument PRINTER.

The GET macro call can be either BLOCKING or NONBLOCKING. The BLOCKING call marks the calling task not runnable until the operation is complete; the NONBLOCKING call returns immediately to the calling task.

If the GET macro call is successful, the N bit in the PS word is cleared. If the GET macro call is unsuccessful, the N bit in the PS word is set.

Command Format:

```
GET line,dev,mode,adr
```

line — The physical octal channel number of the terminal or -1, indicating the CTY.

dev — The word KEYBOARD, indicating the terminal, or the word PRINTER, indicating the network host as a character source.

mode — The word BLOCKING or the word NONBLOCKING.

adr — The destination address for the requested character.

Examples:

```
MOV #11,R0                          ;setup for line 9
GET R0,KEYBOARD,BLOCKING,R1         ;get the next char
BMI ONO                             ;check for no char
MOV #7,FRED                         ;change to line 7
GET FRED,PRINTER,NONBLOCKING,R3     ;get a char from the network
                                    host
BMI ONO                             ;check for no char
```

9.5 IMGPUT Macro Call

Function:

The IMGPUT (IMaGePUT) macro call sends a string of characters to a terminal.

The IMGPUT macro call can be either BLOCKING or NONBLOCKING. The BLOCKING call marks the calling task not runnable until the operation is complete; the NONBLOCKING call returns immediately to the calling task.

If the IMGPUT macro call is successful, the N bit in the PS word is cleared. If the IMGPUT macro call is unsuccessful, the N bit in the PS word is set.

Command Format:

```
IMGPUT line,PRINTER,mode,count,adr
```

line — The physical octal channel number of the terminal.

mode — The word BLOCKING or the word NONBLOCKING.

count — The octal number of characters in the string.

adr — The address of the leftmost character of the character string.

Examples:

```
C12:    12

FOO:    .ASCII/DECSYSTEMS/        ;10 char print string

IMGPUT DEST,PRINTER,BLOCKING,C12,FOO
                                  ;type DECSYSTEM on tty.
```

10.0 DM11-BB INTERFACE MACROS

The DM11-BB is a 16-line modem control multiplexer for the terminal lines on the DN8x system that provides an interface to the Bell 103 and 202 type modems or their equivalents.

The DN8x system includes three macros that reference the DM11-BB: GETBIT, SETBIT, and CLRBIT. These macro calls enable a task to access and to manipulate control bits within the DM11-BB LINE STATUS REGISTER.

The GETBIT macro call retrieves seven bits from the LINE STATUS REGISTER, as well as other bits supplied by the DN8x system (see Section 10.1 for the names and positions of the status bits).

The SETBIT and CLRBIT macro calls set and clear the DATA TERMINAL READY, REQUEST TO SEND, and SECONDARY TRANSMIT status bits in the DM11-BB LINE STATUS REGISTER. These two macro calls also return status information to the calling task.

10.1 GETBIT Macro Call

Function:

The GETBIT macro call retrieves seven bits from the DM11-BB modem control LINE STATUS REGISTER and four status bits from the DN8x system, and then stores the bits into a specified location. The names of the status bits and their bit positions within the word are shown below:

BIT POSITION	NAME OR FUNCTION
15	Set to 1 if either bit 8 or bit 9 is set to 1.
14	Set to 0.
13	Set to 0.
12	Set to 0.
11	Set to 0.
10	Set to 0.
9	Set to 0 if the network host is alive (DS.DIE and DS.DSC are both 0). Otherwise, set to 1.
8	Set to 0 if the line is connected (DS.CON is set to 1). Otherwise, set to 1.
7	RING bit from the LINE STATUS REGISTER.
6	CARRIER bit from the LINE STATUS REGISTER.
5	CLEAR TO SEND bit from the LINE STATUS REGISTER.
4	SECONDARY RECEIVE bit from the LINE STATUS REGISTER.
3	SECONDARY TRANSMIT bit from the LINE STATUS REGISTER.
2	REQUEST TO SEND bit from the LINE STATUS REGISTER.
1	DATA TERMINAL READY bit from the LINE STATUS REGISTER.
0	Set to 1 if the device is assembled as a data set line.

Command Format:

```
GETBIT line,adr
```

line The octal physical line number.

adr The destination of the status bits.

Examples:

```
GETBIT R2,R1  ;place the status bits of the line
              ;specified in Register 2 into Register 1
```

10.2 SETBIT Macro Call

Function:

The SETBIT macro call selectively sets three status bits in the DM11-BB LINE STATUS REGISTER. The bits set are specified by placing a 1 in the corresponding bit position of the mask argument in the macro call. The bits set can be any combination of the DATA TERMINAL READY, REQUEST TO SEND, and SECONDARY TRANSMIT bits (see Section 10.1 for the names and positions of the status bits).

The SETBIT macro call returns the DM11-BB status bits as described in Section 10.1, GETBIT Macro Call.

Command Format:

```
SETBIT line,result,mask
```

line The octal physical line number.

result The location that contains the status bits upon return.

mask The location that specifies the bits to be set to 1.

Examples:

```
MOV #6,R2           ;Set up the bit mask for REQUEST TO SEND and
                    DATA TERMINAL READY

SETBIT #1,R3,R2     ;for line 1

                    ;and return the result in Register 3
```

10.3 CLRBIT Macro Call

Function:

The CLRBIT macro call selectively clears three status bits in the DM11-BB LINE STATUS REGISTER. The bits cleared are specified by placing a 1 in the corresponding bit position of the mask argument in the macro call. The bits cleared can be any combination of the DATA TERMINAL READY, REQUEST TO SEND, and SECONDARY TRANSMIT bits. (See Section 10.1 for the names and positions of the status bits.) The CLRBIT macro call returns the DM11-BB status bits as described in Section 10.1.

Command Format:

```
CLRBIT line,result,mask
```

line The octal physical line number.

result The location that contains the status bits upon return.

mask The location that specifies the bits to clear.

Examples:

```
MOVE #2,R2

CLRBIT R1,R2,R2 ;clear the DATA TERMINAL READY bit

                ;on the line specified in Register 1

                ;and put the status bits into
                ;Register 2
```

11.0 SAMPLE TASK

The task shown below is a simple routine to read addresses from the console terminal and print the contents of those addresses. The user types in enough digits to specify the address and then types a slash (/). The task then prints the contents of the address. this simple task demonstrates the use of many DN8X system macro calls.

```
TSKGEN  ODT,20

ODT:    OPEN    CTY,KEYBOARD
          BMI   ODT.99              ;IN CASE ALREADY IN USE
        OPEN    CTY,PRINTER
          BMI   ODT.99              ;IN CASE ALREADY IN USE
ODT.10: CLR     ODTLOC              ;CLEAR ADDRESS WE ARE
                                    ;EXAMINING
ODT.12: GET     CTY,KEYBOARD,BLOCKING,R0
                                    ;GET A CHARACTER
                                    ;FROM THE CTY
          BMI   ODT.99              ;SHOULD NOT LOSE HERE
        PUT     CTY,PRINTER,NUMBLOCKING,R0
                                    ;ECHO THE CHARACTER
          BMI   ODT.99              ;IN CASE WE LOSE
        BIC     #^C177,R0           ;STRIP THE PARITY
        CMP     #57,R0              ;"/" OPENING CURRENT
                                    ;LOCATION
        BEQ     ODTOPN
        CMP     R0,#70              ;CHECK FOR NOT NUMERIC
        BPL     18$
        CMP     R0,#60              ;CHECK FOR NOT NUMERIC
        BMI     18$
        ANSL    ODTLOC
        ASL     ODTLOC
        ASL     ODTLOC
        BIC     #70,R0              ;MAKE CHARACTER A BINARY
                                    ;NUMBER
        ADD     R0,ODTLOC           ;ACCUMULATE LOCATION
        BR      ODT.12              ;GET RESET OF NUMBER
18$:    MOV     #QESTXT,R1          ;I DON'T UNDERSTAND
        JSR     PC,ODTSTR
        BR      ODT.10
ODTOPN: MOV     #TABTXT,R1
        JSR     PC,ODYSTR
        MOV     @ODTLOC,R0          ;EXAMINE MEMORY
        JSR     PC,ODTOTY           ;TYPE THEM
        MOV     #CRLTXT,R1          ;CARRIAGE RETURN/
                                    ;LINE FEED
        JSR     PC,ODTSTR           ;TYPE THEM
        BR      ODT.10
```

```
;HERE TO TYPE AN OCTAL NUMBER
ODTOTY: MOV     R0,-(P)             ;SAVE DATA
        ROR     R0
        ROR     R0
        ROR     R0
        BIC     #160000             ;STRIP PHANTOM BITS
        BEQ     20$
        JSR     PC,ODTOTY           ;KEEP GOING FOR
                                    ;RESET OF WORD
20$:    MOV     (P)+,R0             ;GET NEXT MOST
                                    ;SIGNIFICANT PART
        BIC     #^C7,R0             ;STRIP EXTRA BITS
        BIS     #60,R0              ;CONVERT TO ASCII
        PUT     CTY,PRINTER,NUNBLOCKING,R0
        RTS     PC
ODTSTR: MOVB    (R1)+,R0            ;GET NEXT BYTE TO TYPE
        BNE     10$
        RTS     PC
10$     PUT     CTY,PRINTER,NUNBLOCKING,R0
        BR      ODTSTR
ODT.99: EXIT                        ;FATAL ERROR
                                    ;SO STOP RUNNING

ODTLOC: .BLKW   1                   ;CURRENT LOCATION
                                    ;TO EXAMINE
HLPTXT: .ASCIZ  / RUNNING ODT/

QESTXT: .ASCIZ  /?      /
TABTXT: .ASCIZ  /   /
CRLTXT: .BYTE   15,12,0
        .EVEN
```

INDEX