TRS-80® MODEL I/III

DEBUG

Catalog
Number
26-2000

TM

Radio Shack TRS-80 SOFTWARE

CUSTOM MANUFACTURED IN THE USA FOR RADIO SHACK A DIVISION OF TANDY CORP.

TRS-80 Debug © 1980, 1981 Tandy Corporation. All rights reserved.
Debug Manual © 1981 Tandy Corporation. All rights reserved.

Reproduction or use without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information obtained herein.

Please refer to the Software license on the back of this manual for limitations on use and reproduction of this Software package.

Important Note to Model I/III Level I Users of TAPE DEBUG (Catalog # 26-2000)

The following commands and options described in the manual for TAPE DEBUG are *not* available in the Level I BASIC:

1. The **I** and **C** single step commands.
2. The **,bbbb** (breakpoint) option of the **J** command.
3. The **T** (entry point address) and **N** (name) options of the **W** (write a system tape) command.

These options and commands are in the Level II and Model III BASIC revisions *only*.

Thank-You!

Radio Shack®

A DIVISION OF TANDY CORPORATION

8759133

Introduction

DEBUG is an easy-to-use monitor for writing and debugging Z-80 machine-language programs.

With DEBUG, you can:

- Display blocks of memory and the Z-80 registers in two different ways.
- Modify individual Z-80 registers or memory locations or enter an entire machine-language program.
- Jump to a program and begin execution.
- Insert breakpoints in your programs.
- Single-step execution of programs.
- Write programs or data to a tape.
- Load programs or data into memory from a tape.

DEBUG uses the memory area from 4200H to 39FFH and can be used only on programs in the user area from 4A00H to the end of memory.

You can use DEBUG with any Model I or Model III Computer System.

Note to Model I Customers with an Expansion Interface: *You cannot use DEBUG when you have the Expansion Interface on. Use the Power Supply for a power source.*

Table of Contents

1 / Loading DEBUG
Level I BASIC 1
Level II BASIC 2

2 / DEBUG Commands 3

3 / Sample Session 7

Appendices

Appendix A/ Summary of DEBUG Commands 9
Appendix B/ DEBUG Memory Map 10
Appendix C/ Half-Screen Sample Display 11
Appendix D/ Model I Subroutines 12

1 / Loading Debug

The procedures you should use for loading DEBUG depend on whether you are using Level I, Level II, or Model III BASIC.

Level I BASIC

Follow the power-up sequence for a ROM-based system described in your *Model I* or *Model III Operation and Basic Language Reference Manual*.

After power-up, the Computer will display

```
READY >
```

When the tape is rewound in the recorder and the volume is set correctly, press the recorder's PLAY button and type `CLOAD` (ENTER). Blinking asterisks will appear in the upper right of the screen to tell you that the tape is loading. If the asterisks do not appear, consult your computer's Owner's Manual.

After loading, control will jump directly into DEBUG. The DEBUG half-screen display should appear and the machine will now accept DEBUG commands.

Making a backup of DEBUG

After you have loaded DEBUG, it's a good idea to make a backup tape or two in case the original becomes worn out or damaged.

1.) Prepare a blank tape for recording. Make sure that it is past the leader, if it has one. Press the PLAY and RECORD buttons together (put the recorder in "Record" mode).

2.) To record, enter the following sequence exactly and ignore the machine's responses as you go:

```
W41FE (ENTER) 493F (ENTER)
```

DEBUG will now make a system tape of itself. When the recorder stops, you should confirm that DEBUG is on the tape by going through the LOAD DEBUG sequence with this tape.

Level II or Model III BASIC

Follow the power-up sequence for a ROM-based system described in your *Model I* or *Model III Operation and Basic Language Reference Manual*. (If you have a disk system and want to use this program, hold the (BREAK) key and press the RESET button.)

The Model III system starts a little differently from Model I. When your Model III prompts with CASS?, press: (L). Both models load the same now.

1. The Computer will prompt you with the question: MEMORY SIZE?. For now, just press (ENTER).

2. Type SYSTEM (ENTER). The computer will prompt with *? Type DEBUG (ENTER). Rewind the DEBUG tape and press the recorder's PLAY button (be sure the volume level is set properly for your system). One non-blinking and one blinking asterisk in the upper right of the display show that loading is in progress.

3. After 30 seconds or so, the recorder should stop and another *? should appear (if not, see your computer's Owner's Manual.) Press (/) (ENTER) to start the program.

The DEBUG half-screen display should appear and the machine will now accept DEBUG commands.

Making a Backup of DEBUG

After you have loaded DEBUG, it's a good idea to make a backup tape or two just in case the original becomes worn out or damaged.

***Note:** If you are using a TRS-80 Model III (Level II only) and wish to make your backup at 1500 baud, first type the following exactly, ignoring the machine's responses as you go (otherwise a 500 baud tape will be made):*

M4211(SPACEBAR)01 (ENTER) (ENTER)

1.) Prepare a blank tape for recording. Make sure it is past the leader if it has one. Press the PLAY and RECORD buttons together (put recorder in "Record" mode).

2.) To record, enter the following sequence exactly and ignore the machine's responses as you go:

W4332 (ENTER) 493F (ENTER) 4909 (ENTER) DEBUG (ENTER)

DEBUG will now make a system tape of itself. When the recorder stops, you should confirm that DEBUG is on the tape by going through the LOAD DEBUG sequence with this tape. (If you have a Model III, you can simply press (ENTER) in response to CASS?, since the baud rate is already set.)

2 / DEBUG Commands

DEBUG responds immediately when you press a single key command; there's no need to press (ENTER). If a command needs more data, such as an address, the computer will prompt with a request for what's needed.

D (Display Memory Contents)

Press (D) to display a block of memory. DEBUG will prompt with:

```
ADDRESS=
```

You should type in the hexadecimal address of the beginning byte of the block of memory you wish to see. Press (SPACEBAR) to enter the memory address. If you make a mistake in the address, just key in the correct address also and press (SPACEBAR). DEBUG will look only at the last four characters typed. Example: To display memory beginning at 4020H, type

```
D ADDRESS= 4020 (SPACEBAR)
```

The display will be either half- or full-screen, depending on the format currently specified (see next two commands).

X (Half-Screen Display Mode)

Press (X) to put the Display in the half-screen format.

A 128-byte block of memory will be displayed, starting with the next lower address which is an even multiple of 16. The next line on the display starting with PC is a list of the 16 bytes to be found beginning at the address specified by the PC register.

The bottom two lines of the display are the Z-80 registers and their contents.

This is the display mode DEBUG is in when you load it. Appendix C shows a typical half-screen display.

S (Full-Screen Display)

Press (S) to put the display in the full-screen mode. A 256-byte block of memory will be displayed, starting with the next lower address which is an even multiple of 256. This display fills the screen, so commands entered in this mode will overlay the bottom line of display.

; (Increment Display Address)

Press (;) to increment the displayed address by 16 in the half-screen mode or 256 in the full-screen mode.

- (Decrement Display Address)

Press (-) to decrement the displayed address by 16 in the half-screen mode or 256 in the full-screen mode.

M (Modify Memory)

Press (M) to change the contents of a memory location or to enter a machine-language program. DEBUG will respond with the prompt:

```
ADDRESS=
```

You should type in the hexadecimal address of the memory location you wish to modify. As with the D command, press (SPACEBAR) to enter the address.

The present contents of the memory location you specified will be displayed below the AF at the lower left of the screen. Type in the value you wish to change to.

Press (SPACEBAR) to enter the change and increment to the next memory location.

Press (SPACEBAR) with no entry to advance to the next memory location without any change.

Press (ENTER) instead of (SPACEBAR) to make a change and decrement to the previous memory location.

Press (ENTER) with no entry to terminate the modify command and return to DEBUG.

Examples:

To change memory location 7000H to the value 3BH, type

```
M ADDRESS= 7000 (SPACEBAR) 3B (SPACEBAR) (ENTER)
```

To change memory location 65ABH and the following two bytes to 00H, type

```
M ADDRESS= 65AB (SPACEBAR) 00 (ENTER) 00 (ENTER) 00 (ENTER) (ENTER)
```

R (Change Register Contents)

To change the contents of a Z-80 register pair, type:

```
Raa,bbbb (SPACEBAR)
```

where ''aa'' is the name of a register pair (AF, BC, DE, HL, AF', BC', DE', HL',IX, IV, SP, PC) and ''bbbb'' is the new register contents. If fewer than four digits are typed before pressing (SPACEBAR), leading zeros are assumed.

Example: To change the DE' register to 43H, type

```
RDE',43 (SPACEBAR)
```

J (Jump to a Program)

Press (J) to begin execution of a machine-language program, setting optional breakpoint. DEBUG will respond with the prompt: ADDRESS=

Type in the address(es) in one of three formats:

ADDRESS= *aaaa,bbbb* (ENTER) ADDRESS= *aaaa* (ENTER)
ADDRESS= *,bbbb* (ENTER)

where *aaaa* is the address you wish to jump to and *bbbb* is the address where you want a breakpoint inserted. If *aaaa* is omitted, the displayed PC will be used.

When a breakpoint is set, the contents of the specified address are saved and a hexadecimal F7 (RST 30H) is put there instead. When execution reaches this location, control is returned to DEBUG and the saved value is restored. You can then examine registers or memory at this point in the program's flow and verify that your program is doing what you intended.

Note: *Breakpoints must be set at the beginning of multi-byte Z-80 instructions to operate properly. Also, breakpoints cannot be set in ROM addresses.*

I (Single Step)

Press (I) to execute a single Z-80 instruction. The instruction in the memory location pointed to by PC will be executed. The PC is incremented, the display is updated, and control returns to DEBUG.

The I command will not advance past a call or jump to a ROM address.

This function is especially useful in conjunction with breakpoints to debug a small critical part of a program too long to single-step through in its entirety.

C (Single Step)

If the instruction at the memory location is a call and you wish to complete the entire call/return sequence, press (C). The call is executed and control returns to DEBUG when the subroutine returns. Otherwise, this command is like I.

The C command will step through a called sequence, but not a jump to a ROM address.

U (Update)

Press (U) to cause the display to be updated continuously.

This function will be of use if you have interrupt driven devices attached to your TRS-80, or for looking at the Model III real time clock, address 4040H.

Press any key to exit this mode.

T (Load a System Tape)

Press (T) to begin immediately loading a system tape into memory. Upon completion, the first two bytes displayed in the upper left corner will be the program's entry point.

Note for Model III (Level II only) users: *Make sure the baud switch (location 4211H) is in agreement with the tape being loaded:*

baud switch = 00 for 500 baud tapes
baud switch = 01 for 1500 baud tapes

Use the M command to make this change.

Pressing (BREAK) will cancel the load on Model III, but not on Model I. Pressing the RESET button will cancel the load on both computers.

W (Write a System Tape)

Press (W) to write a system tape. DEBUG will prompt with:

S = Key in the four digit hexadecimal address of the first byte of the block you wish written, followed by (ENTER).

E = Key in the address of the last byte of the block, followed by (ENTER).

T = Key in the entry point address of the program, followed by (ENTER).

N = Key in the name (up to six alpha-numeric characters, the first of which should be a letter) you wish the program to be called, followed by (ENTER). This interchange is displayed on a single line. After writing the tape, DEBUG will clear this line.

If you wish to cancel the command, press (ENTER) with no address for S, E, or T. While entering the name, use (BREAK) to cancel. You cannot cancel while a tape is being written; you must wait until it is done.

Q (Quit)

Press (Q) to exit DEBUG entirely and return to the READY state.

3 / Sample Session

You can follow these simple instructions to do the sample session, but a knowledge of Z-80 assembly language and machine language will be necessary to write your own programs.

Below is a realistic section of code that will read keyboard input and accept only a valid number. The number is converted from ASCII into binary and summed into a location in memory.

This program terminates by returning to DEBUG (entry point: 4909H); this is an easy way to end programs to be used under DEBUG.

Sample Program (Level II only)

```
Address Object    Label   OP     Operands    Comment
                          ORG    7000H       ;ENTER PROGRAM AT 7000H
7000    213070    START   LD     HL,7030     ;POINT HL TO SUM
7003    CD4900    GET     CALL   $KBWAIT     ;GET CHARACTER
7006    FE3A              CP     3AH         ;ABOVE NUMBERS?
7008    F20370            JP     POS,GET     ;IF YES, GET ANOTHER
700B    FE30              CP     30H         ;BELOW NUMBERS?
700D    FA0370            JP     NEG,GET     ;IF YES, GET ANOTHER
7010    D630              SUB    A,30H       ;CONVERT ASCII TO NUMBER
7012    86                ADD    A,(HL)      ;ADD SUM
7013    77                LD     (HL),A      ;STORE IN SUM
7014    C30949            JP     4909H       ;DEBUG ENTRY
```

$KBWAIT is a ROM subroutine (entry point 0049H) that waits for a key to be pressed, then returns with its ASCII value in the A register. After you have familiarized yourself with the program and understand what it should do, have DEBUG display memory in half-screen mode beginning at 7000H.

Enter the program using the M command. (If you need help, just enter the following where / means press (SPACEBAR):

M7000/21/30/70/CD/49/00/FE/3A/F2/03/70/FE/30/FA/
03/70/D6/30/86/77/C3/09/49/(ENTER).) When you have finished entering the program, verify that everything is where it should be by comparing it with the half-screen display sample in Appendix C. Remember, you are concerned only with the contents of 7000H to 7016H.

Now, use the J command to jump to 7000. Press (ENTER). The machine is in $KBWAIT waiting for a key to be pressed.

Press (A) and notice that nothing happens, because the A is not a number. The program is back in $KBWAIT waiting for another key to be pressed.

Press [3] and notice that the DEBUG blinking square is back. Also, notice that the value in memory location 7030H is different. The 3 is a valid number so it was converted from ASCII and added to memory location 7030H, and control returned to DEBUG.

Now use the R command to change HL to 5757, and PC to 7000. Press [I] and note the changes in HL and PC. Press [C] to do the call to $KBWAIT. The machine is now waiting for a key to be pressed.

Press [B] and notice the changes in PC and AF. Press [I] to do the compare and notice that the sign bit (bit 1) in the F register is set (a negative number) because B is above the numbers in ASCII.

Press [I] to do the jump back to the call. Press [C] to do the call to $KBWAIT. Press [5] to satisfy $KBWAIT and then use [I] to step all the way through, noting changes in the F register as the compares are done. The jumps will not be taken and the converted number will be added to SUM.

When the PC shows 7014 and you are ready to do the return to DEBUG, press [J] only. If you press [I], you will continue single-stepping into DEBUG.

Breakpoints are of little use with a program this short; therefore, let's assume this is part of a much longer program that you do not wish to single-step through, but want only to see the A register before and after the subtraction.

Use the J command to jump to 7000 with a breakpoint at 7012H.

Press [6] to satisfy $KBWAIT. Notice that the PC is at 7012 ready to do the subtraction, and note the value in A.

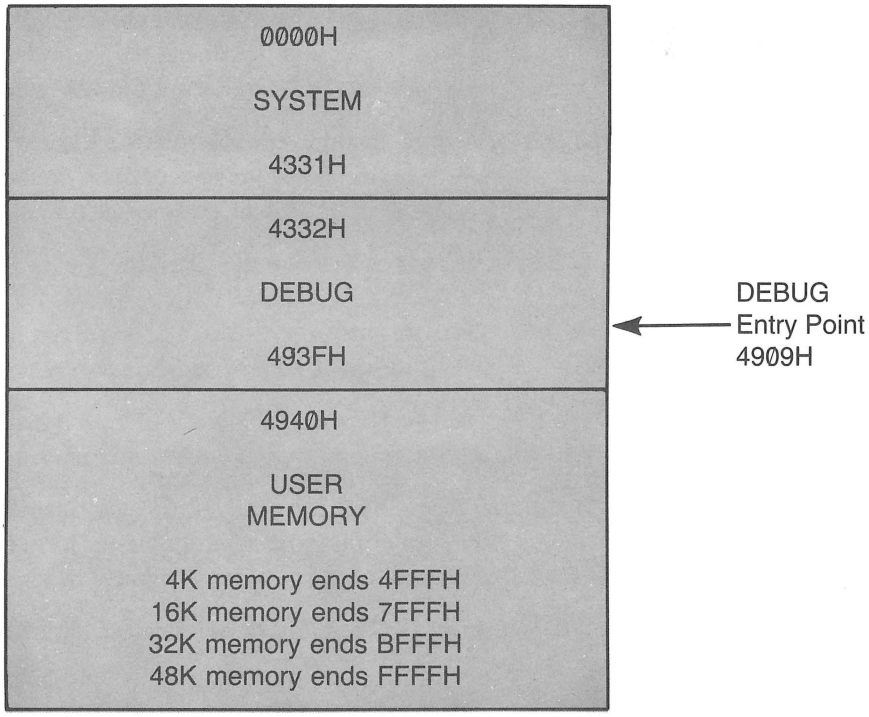
Press [I] to do the subtraction. Note the new value in A. Press [J] to continue at the address in PC.

Much longer sections of code could have been before or after this breakpoint and checking this subtraction would have been just as easy.

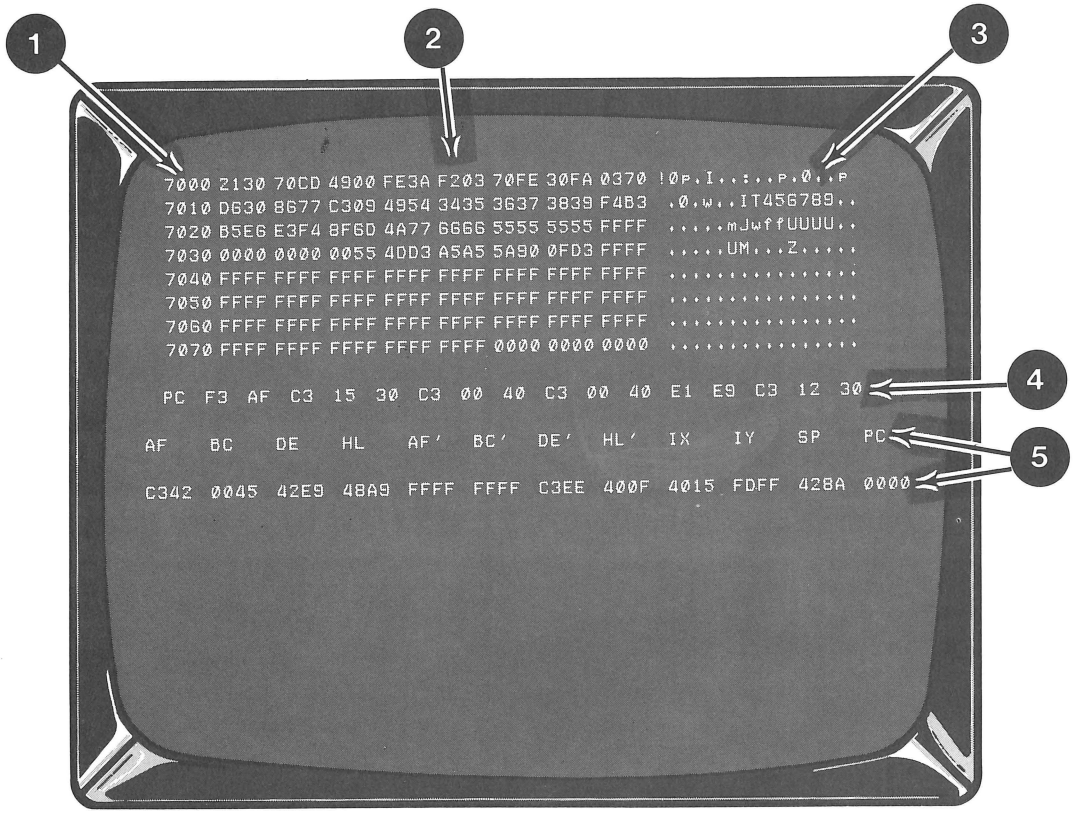
Appendix A / Summary of DEBUG Commands

; Increment display address
- Decrement display address
C Single-Step through single call/return sequence
D Display memory contents
D ADDRESS=.49B3 (SPACEBAR)
I Single-Step execution of program
J Jump to program with optional breakpoint
J ADDRESS= 7000,705A (ENTER)
M Modify memory
M ADDRESS= 7200 (SPACEBAR) 53 (ENTER)(ENTER)
Q Return to BASIC READY state
R Change register contents
RHL ,6302 (SPACEBAR)
S Full-Screen display mode
T Load a system tape
U Update display continuously
Press any key to cancel
W Write a system tape
S= starting byte address
E= ending byte address
T= entry point address
N= name of program
X Half-Screen display mode

Appendix B / Memory Map While Running Under DEBUG



Appendix C / Half-Screen Sample Display



1. **Memory address of the first byte of a 16 byte row**
2. **Hexadecimal memory contents**
3. **ASCII translation of 16 byte row, a period (.) indicates a non-displayable character**
4. **The 16 bytes to be found beginning at the address in PC**
5. **Z-8Ø registers and their contents**

Appendix D / Model I Subroutines

This appendix lists subroutines which are a part of the Model I's Read Only Memory (ROM). You can access these subroutines with DEBUG.

The Model III ROM subroutines are listed in the *Technical Information Section* of the Model III Owner's Manual.

Model I Level I Subroutines

Here are the Model I Level I BASIC subroutines arranged in alphabetical order.

DISPLAY BYTE
KEYBOARD SCAN
LOAD FROM CASSETTE
RETURN TO BASIC READY
RETURN TO DEBUG
SAVE MEMORY TO CASSETTE
TURN ON CASSETTE

DISPLAY BYTE AT CURSOR
RST 10H

This subroutine displays a character at the current cursor.

Entry Conditions:
A = ASCII Character

Exit Conditions:
DE and IY are altered

Sample Programming:

```
                LD      A,(HL)          ;put character in A
reg.
                RST     10H             ;display byte
```

KEYBOARD SCAN
0B40H

This subroutine scans the keyboard for a character. The character is displayed at the current cursor.

Entry Conditions:
none

Exit Conditions:
A = ASCII Character
Z = 1 if no key is pressed

Sample Programming:

```
WAIT    CALL        0B40H       ;scan Keyboard
        JR          Z,WAIT      ;A=character if
                                ;fall through
```

LOAD MEMORY FROM CASSETTE
0EF4H

Turns on Cassette Recorder and searches for header; then reads in a block of data and turns Cassette Recorder off.

Entry Conditions:
none

Exit Conditions:
HL = address of last byte read + 1
Z = 0 if checksum error
Z = 1 if no error

Sample Programming:

```
        CALL        0EF4H       ;read tape
```

RETURN TO BASIC "READY" STATE
01C9H

This is the entry-point for BASIC. JUMP (do not CALL) to the address 01C9H.

Entry Conditions:
none

Sample Programming:

```
        JP          01C9H       ;return to READY
```

RETURN TO DEBUG
4909H

Return control to DEBUG monitor with a jump (not a call) to the address 4909H.

Entry Conditions:
none

Sample Programming:

```
        JP          4909H       ;return to DEBUG
```

SAVE MEMORY TO CASSETTE
0F4BH

Write a block of memory to the Cassette recorder. The Cassette is turned off before return.

Entry Conditions:
HL = starting byte address
DE = last byte address + 1

Exit Conditions:
none

Sample Programming:

```
        CALL        0FE9H       ;Turn on cassette
        LD          HL,START    ;start address
        LD          DE,LAST1    ;End address +1
        CALL        0F4BH       ;output and turn
                                ;off cassette motor
```

TURN ON CASSETTE
0FE9H

Turn on Cassette drive motor.

Entry Conditions:
none

Exit Conditions:
none

Sample Programming:
See SAVE MEMORY TO CASSETTE

Model I Level II Subroutines

Here are the Model I Level II BASIC subroutines listed alphabetically.

DEFINE DRIVE
DISPLAY BYTE
INPUT FROM CASSETTE
KEYBOARD SCAN
OUTPUT TO CASSETTE
OUTPUT TO LINE PRINTER
RETURN TO BASIC "READY"
RETURN TO DEBUG

DEFINE DRIVE
0212H

This routine will specify which drive the Cassette input/output routines will use.

Entry Conditions:
A = 0 for onboard Cassette
A = 1 for external Cassette

Exit Conditions:
none

Sample Programming:
See OUTPUT TO CASSETTE

DISPLAY BYTE
0033H

See $VDCHAR in Model III section.

INPUT FROM CASSETTE
Ø235H

Input a byte from Cassette. Before you use this routine to input data, you must turn on Cassette motor and find the leader and sync byte (use a call to Ø296H to accomplish this). The CALL to Ø235H must be often enough to keep up with 5ØØ baud. You must turn off the Cassette motor when you are done with input (use a call to Ø1F8H).

Entry Conditions:
none

Exit Conditions:
A = byte from Cassette

Sample Programming:

```
                LD          A,Ø          ;onboard cassette
                CALL        Ø212H        ;define drive
                CALL        Ø296H        ;turn on motor and
                                          find leader
  Set up loop to
collect data and /  CALL      Ø235H        ;input byte
check for end of /  LD        (HL),A       ;collect data
data
                CALL        Ø1F8H        ;turn off cassette
                                          motor
```

KEYBOARD SCAN
ØØ2BH

See $KBCHAR in Model III section

OUTPUT TO CASSETTE
Ø264H

Output byte to Cassette recorder.
In order to read back in what you write with this routine, you must have turned on the Cassette motor and written the leader and sync byte (use a call to Ø287H which has no entry or exit conditions) just prior to calling Ø264H. You must also turn off the Cassette motor after you have output the last byte (use a call to Ø1F8H). Be sure to call Ø264H often enough to keep up with 5ØØ baud.

Entry Conditions:
A = byte to output

Exit Conditions:
none

Sample Programming:

```
                LD          A,Ø         ;onboard cassette
                CALL        Ø212H       ;define drive
                CALL        Ø287H       ;turn on and write
                                         leader
   Set up loop to
put characters in A LD          A,(HL)      ;get next char,
and check for last  CALL        Ø264H       ;output char,
character.
                CALL        Ø1F8H       ;Turn off motor
```

OUTPUT TO LINE PRINTER

The Printer is ''memory mapped'' on the Model I. This means that to output a byte on the printer, you simply load the byte into memory position 37E8H. To find out if the printer is busy, check if bit seven of memory location 37E8H is a ''1''.

Sample Programming:

```
                LD          HL,37E8H    ;HL=printer mapped
                                        location
        PTR     LD          D,(HL)      ;load status word
                BIT         7,D         ;printer busy?
                JP          NZ,PTR      ;if bit 7=1 look again
                                        ;if fall through then
                                         printer is ready to
                                         output byte
                LD          (HL),A      ;output byte
```

RETURN TO BASIC "READY"
1A19H

Return to Basic "READY" with a jump (not a call) to 1A19H.

Entry Conditions:
none

Sample Programming:

```
        JP          1A19H       ;go to READY
```

RETURN TO DEBUG
4909H

Return to DEBUG with a jump (not a call) to 4909H.

Entry Conditions:
none

Sample Programming:

```
        JP          4909H       ;return to DEBUG
```

IMPORTANT NOTICE

ALL RADIO SHACK COMPUTER PROGRAMS ARE LICENSED ON AN "AS IS" BASIS WITHOUT WARRANTY.

Radio Shack shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by computer equipment or programs sold by Radio Shack, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer or computer programs.

NOTE: Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

RADIO SHACK SOFTWARE LICENSE

A. Radio Shack grants to CUSTOMER a non-exclusive, paid up license to use on CUSTOMER'S computer the Radio Shack computer software received. Title to the media on which the software is recorded (cassette and/or disk) or stored (ROM) is transferred to the CUSTOMER, but not title to the software.

B. In consideration for this license, CUSTOMER shall not reproduce copies of Radio Shack software except to reproduce the number of copies required for use on CUSTOMER'S computer (if the software allows a back-up copy to be made), and shall include Radio Shack's copyright notice on all copies of software reproduced in whole or in part.

C. CUSTOMER may resell Radio Shack's system and applications software (modified or not, in whole or in part), provided CUSTOMER has purchased one copy of the software for each one resold. The provisions of this software License (paragraphs A, B, and C) shall also be applicable to third parties purchasing such software from CUSTOMER.

RADIO SHACK TC **A DIVISION OF TANDY CORPORATION**

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
280-316 VICTORIA ROAD RYDALMERE, N.S.W. 2116	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

Printed in U.S.A.