POLEX USERS MANUAL

27 February 2024

Martin Eberhard

Revision History

Polex Rev.	Manual Rev.	Date	Author	Notes
3.00		16 Dec 2013	M. Eberhard	First released version
3.01		19 Dec 2013	M. Eberhard	Fixed for programs with stack in first 4K of RAM
3.02		20 Dec 2013	M. Eberhard	Add RTC count-down timer
3.03		24 Dec 2013	M. Eberhard	Squeeze code. Delete higher baud rate support (because 8251 fails at high baud rates). Allow selection of cassette or printer via STBAUD & PXBAUD. Add address to each line in EN command. Change timer to count up, with overflow flag. Translate BS to del in CPRINT. Provide external access to ONBRD and OFFBRD subroutines.
	X	11 Jan 2014	M. Eberhard	First manual release
3.04	X	03 Jun 2014	M. Eberhard	Support Rev F CPU boards, with ROM disable at bit 6 in BRG port, as well as bit 5 (for rev 0 CPUs)
3.05	X	26 Apr 2016	M. Eberhard	Support e.g. George Risk keyboard as well as Polymorphic Systems keyboard: Strip keyboard parity. Also accept ESC as command abort. Default EX command to address 0000 (to save code space).
3.06		28 APR 2016	M. Eberhard	Convert lowercase to uppercase for Polex commands.
3.07	X	9 Jun 2016	M. Douglas, M. Eberhard	Improved serial transmit, convert BS to DEL, Simplify HL error checking
3.08	X	14 Jun 2016	M. Eberhard	Don’t echo during HL if the receive queue is full
3.09	X	18 Jun 2016	M. Eberhard	Add echo control to HL command
3.10		22 Jun 2016	M. Eberhard	Fix long-standing bug
3.11		18 Nov 2019	Douglas/ Eberhard	PTimer interrupt calls a fixed location which may be modified by user code, and is initialized to be as return. Change HL command default to “no echo”
3.11		27 Feb 2024	M. Eberhard	Add footnotes saying that Phantom must disable writes as well as reads

TABLE OF CONTENTS

Polex
A Monitor Extension for the Polymorphic Systems Poly-88 Computer

INTRODUCTION

The Polymorphic Systems Poly-88 CPU Board has three 1K ROM sockets and 1K of SRAM onboard. The first ROM socket (at address 0000) contains the Poly-88 resident monitor. This monitor provides a very basic cassette tape loader, as well as an interrupt-driven “front panel” debugging utility.

Because the resident monitor resides in ROM at address 0000, it interferes with operating systems such as CP/M, which require RAM at address 0000.

The designers of the Poly-88 did anticipate this problem, and provided a mechanism for disabling the onboard ROMs and RAM via an output port on the CPU board. However, this feature is disabled in the standard configuration of the Poly-88.

To enable this feature, the Poly-88 manual describes one trace to cut, and two jumpers to install on the CPU board. (Unfortunately, these instructions are not quite correct - see the Modifications section later.)

Polymorphic Systems also provided for extensions to the Poly-88 monitor (version 4.0), by checking for the presence of a ROM in socket 1 during the monitor’s initialization.

Polex is such an extension for the Poly-88 monitor, and resides in ROM socket 1 at address 0400h. It is automatically invoked upon reset or power-on of the Poly-88.

At initialization, Polex relocates itself to high memory, at address F400. It then switches off the onboard memory, and switches low-address RAM on via the S-100 Phantom[1] line.

Once this initialization is completed, Polex prints a prompt on the console, and waits for commands from the keyboard.

Polex provides several important functions to the Poly-88:

- 60K of free RAM space from 0000 through 0EFFF
- Simple (CP/M compatible) I/O and status routines for the console and the serial port, with standardized addresses, control, and handshaking

[1] The memory board’s phantom input must disable writes as well as reads.

- Interrupt-driven serial I/O, with deep enough queues to keep up with reception, even while the Poly-88 monitor scrolls the screen, which it does the hard way[2]
- Enhanced monitor functions: Examine and modify memory, Execute command, Intel hex file load, and Terminal mode
- Access to the Poly-88 tape load routines
- Support for an additional extension in ROM socket 2, which may add commands to Polex

MEMORY REQUIREMENTS

Polex requires RAM from address F400h through F7FFh. If you also plan to use a Polex Extension Module, then you will need RAM from F000h through F7FFh.

The video screen memory must be in the Poly-88 standard address range: F800h through FBFFh.

If you plan to run CP/M or similar operating system (which requires RAM at address 0000), then you will need a RAM board that supports the PHANTOM signal[3] on the S-100 bus.

A good choice for a RAM board is one of the 64K SRAM boards that are built with 2Kx8 SRAM chips. These boards typically allow de-selection of individual 2K blocks. For example the Digital Research Computers (aka Tanner) 64K SRAM board works perfectly, supporting PHANTOM, and allowing de-selection of the top 2K of memory (F800h through FFFFh) with a dip switch.

If you plan to use a memory-mapped disk controller (such as the Micropolis (AKA Vector Graphic) FD Controller B or the Northstar MDS), then you can address this disk controller board at address FC00h through FFFFh. This leaves your system with 62K of RAM. When Polex and an Extension Module are loaded, you still have 60K available for CP/M or other Operating system.

When installed, the system memory map looks like this:

Address Range	Occupant
FC00h-FFFFh	Potential Disk Controller
F800h-FBFFh	Poly-88 Video Board
F400h-F7FFh	Polex RAM image
F000h-F3FFh	Potential Extension Module RAM image
0040h-EFFFh	Available RAM
0000h-003Fh	Interrupt Vectors

[2] 9600 baud requires 2 stop bits or 5 mS line delay, if echo is enabled.
[3] The memory board's phantom input must disable writes as well as reads.

I/O PORTS

Polex requires the I/O devices on Poly-88 CPU board and video board to be at their standard I/O port addresses:

Port Address	Input	Output
00h (CPU)	**8251 USART Data**	**8251 USART Data (Printer & Cassette Tape interfaces[1])**
	See 8251 Data Sheet	
01h (CPU)	**8251 USART Status**	**8251 USART Control(Printer & Cassette Tape interfaces[1])**
	See 8251 Data Sheet	
04h (CPU)	-	**Baud Rate Generator**
		Bits <3:0> select the baud clock
		Bit <4> = 0 selects the cassette tape = 1 selects the printer port
		Rev 0 CPU boards: Bit <5> = 0 selects onboard memory = 1 selects offboard memory
		Rev F CPU Boards: Bit <6> = 0 selects onboard memory = 1 selects offboard memory
		Rev F CPU Boards: Bit <7> Unknown function: writing 1 to this bit hangs Polex
08h (CPU)	-	**Clear Real Time Clock Interrupt**
		Any value written clears the interrupt
0Ch (CPU)	-	**Enable Single Step Interrupt**
		Any value written enables single-step
F8h (Video)	**Keyboard Data**	-

1. The CPU's serial port is used for both the cassette tape and the general purpose ("printer") interfaces. The cassette tape interface should be configured as Device Address 0, and the printer interface should be configured as Device Address 1. Devices Addresses are set via a jumper on the interface boards on the rear panel of the Poly-88.

INTERRUPTS

The Poly-88 CPU board has several I/O devices that can only operate in interrupt mode - they cannot be polled without modifying the Poly-88 hardware.

Polymorphic Systems did eventually provide Errata that described (somewhat extensive) board modifications that would allow some of these devices to be polled, but these modifications are not required for Polex, nor to run CP/M when the Polex ROM is installed.

Vector interrupt hardware on the CPU board directs these interrupts to hardwired addresses:

Vector Address	Interrupt Source
0000h	Reset
0008h	S-100 VI6 signal
0010h	S-100 VI5 signal
0018h	S-100 VI4 signal
0020h	VI3: 8251 UART (on the CPU board)
0028h	VI2: Keyboard (on the Video board)
0030h	VI1: Real-time Clock (on the CPU board)
0038h	VI0: Single-Step Logic (on the CPU board)

REQUIRED POLY-88 MODIFICATIONS

The following modifications to the Poly-88 CPU board enable control of the onboard memory, via bit 5 of the Baud Rate Generator port on the Poly-88 CPU, as well as PHANTOM-control of a RAM board. (These modifications are recommended in the Poly-88 manual, though a mistake in the manual has been corrected here.)

1) Cut trace HH
2) Jumper from the right side of HH to pad H
3) Jumper from S-100 Pin 67 (PHANTOM) to pad J

Steps 1 and 2 connect bit 5 of the Baud Rate Generator port to the onboard memory disable circuit, allowing software to enable or disable onboard memory.

Step 3 (which was incorrectly described in the Poly-88 manual) drives the S-100 PHANTOM signal whenever onboard memory is selected. This allows the lowest 4K of memory to banks-switch between onboard memory (RAM and ROM) and offboard memory (most likely RAM), based on the state of bit 5 in the Baud Rate Generator port.

Polex Commands

Polex commands may be typed at the Polex prompt ‘>’. Commands are executed once you type the Return key. You can correct typing mistakes with the RUB OUT key.

System Commands

PO (Return to Poly-88 Monitor)

Return to Poly-88 monitor’s command loop, to access the Poly-88 tape loader routines. (You must reset the computer to return to Polex.)

Once you issue this command, the Poly-88 will behave exactly as it would without the Polex ROM installed - you will see a blank screen with a cursor, and you can type ‘B’ or ‘P’ to load a cassette tape, or control-Z to enter the “front panel” mode.

Memory Commands

DU <ADDR> [<END>] (Dump Memory)

Dump memory on the console screen in hexadecimal, from <ADDR> through <END>. If no <END> address is specified then just one address is dumped. Press the space bar to pause the dump. Press the ‘ALT MODE’ or ‘ESC’ key to abort the dump.

EN <ADDR> (Enter Memory Data)

Enter hex data into memory starting at <ADDR>, using a space or Return as a separator between bytes. Type Return on a blank line to exit. Press the ‘ALT MODE’ or ‘ESC’ to abort without saving the current line of hex data.

EX [<ADDR>] (Execute)

Execute at <ADDR>. (The address defaults 0000.) Programs can return to Polex by jumping to PXINIT or PXWARM.

FI <ADDR> [<END> [<VAL>]] (Fill Memory)

Fill memory from <ADDR> through <END> with <VAL>. You must specify a start address, <ADDR>. If no fill value <VAL> is specified, then the range will be filled with 00. If no end address is specified, then only <ADDR> will be filled with 00.

Serial Port Commands

HL [0/1](Load Intel Hex File)

Load Intel Hex file into memory at the addresses specified in the hex file. “HL 0” or just “HL” disables echoing - in which case a single ‘.’ is printed for each received record. With “HL 1”, all received characters are echoed to the screen. Press the ‘ALT MODE’ or ‘ESC’ key to abort. Any hex record data that would overwrite Polex will cause an error. Polex will also abort with an error if any record’s checksum is incorrect.

If the serial port is set for 9600 baud, then echo should be disabled, unless you either transmit with 2 stop bits, or with a minimum of 5 mS delay after each line. (This gives the Poly-88 enough time to scroll the screen.)

SB <RATE> (Set Baud Rate)

Set serial port baud rate and select serial port. The least-significant nibble (bits 3:0) of <RATE> selects the baud rate from table 1:

Bits 3:0 Value	**Baud Rate**	**Bits 3:0 Value**	**Baud Rate**
0	Disabled	8	900
1	50	9	1200
2	75	A	1800
3	110	B	2400
4	134.5	C	3600
5	150	D	4800
6	300	E	7200
7	600	F	9600

Table 1

(Higher baud rates are not supported by Polex primarily because the 8251 (plain) USART does not handle asynchronous communication well with a divide-by-one clock, which is the only way to get higher baud rates with this USART.)

The most-significant nibble (bit 4 in particular) of <RATE> selects the serial port to use. 0 selects the cassette tape port, and 1 selects the printer port.

TE (Terminal Mode)

Terminal mode - keyboard data goes to the serial port, and serial port data goes to the console screen. Press the ‘ALT MODE’ or ‘ESC’ key to exit. (This command is useful for verifying a serial port connection.)

Interfaces

Software Entry Points

Polex provides the following set of fixed-address entry points for software access:

PXINIT (F400) (Restart Polex)

This assumes that Polex is still resident in RAM.

PXCSTA (F403) (Get Console Status)

Call this address to get the console keyboard status.
On Return:
A=0 and Z flag set if no keyboard character waiting
A=FFh and Z flag cleared if a keyboard character is waiting.
All other registers are preserved

PXCIN (F406) (Get Console Input)

Call this address to get one keyboard character.
Waits for a keyboard character, and returns it in A. The Z flag is always cleared. All other registers are preserved.

PXCOUT (F409) (Console Output)

Call this address to send one character to the console video screen.
On Entry:
C=character to print on the video screen
On Return:
A=C
All other registers are preserved.

PXSSTA (F40C) (Get Serial Port Input Status)

Call this address to get the serial input port status.
On Return:
A=0 and Z flag set if not ready (meaning that the receive queue is empty)
A=FFh and Z flag cleared if a ready (the receive queue is not empty)
All other registers are preserved.

PXSIN (F40F) (Get Serial Port Input)

Call this address to get one serial port character.
Waits for a serial port character, and returns it in A. The Z flag is always cleared. All other registers are preserved.

PXSOUT (F412) (Serial Port Output)

Call this address to send one character to the serial port.
On Entry:
   C=character to send to the serial port
On Return:
   A trashed
   All other registers are preserved.

PXSOST (F415) (Get Serial Port Output Status)

Call this address to get the serial output port status.
On Return:
   A=0 and Z flag set if not ready (meaning that the transmit queue is full)
   A=FFh and Z flag cleared if a ready (meaning that the transmit queue is not full)
   All other registers are preserved.

PXBAUD (F418) (Set Serial Port Baud Rate and Port)

Call this address to set the serial port baud rate, and to select the cassette port or the printer port.

See Table 1. All registers except the PSW preserved.

PXIHEX (F41B) (Get hex input from console)

Call this address to get one 16-bit hexadecimal value from the user input buffer.

Leading blanks will be skipped.

On Entry:
   DE=address of text string to parse, which must be terminated with space or FFh
On Return:
   PSW trashed
   BC preserved
   DE=first address after the last hex character
   HL=16-bit hex value
   Z flag set and carry cleared only if no value found
   Abort to PXWARM if bad hex found
   All other registers are preserved.

PXOHEX (F41E) (Print Hexadecimal on the Console)

Call this address to print the value in A in hexadecimal on the console.

All registers are preserved.

PXWARM (F421) (Warm-start re-entry)

This entry point restores the stack and sets the baud rate to its default (9600 baud), before returning to the command processor.

PXCIN2 (F424) (Get Console Character, if Available)

Call this address to get one console keyboard character, only if one is available.
On Return:
   A=0 and Z flag is set only if no keyboard character waiting.
   A=keyboard character, if one was waiting
   The ‘ALT MODE’ key (which produces an ASCII ‘}‘ character) and the ESC key abort to Polex’s command processor.
   All other registers are preserved.

PXOFBD (F427) (Enable Offboard Memory)

Call this address to select offboard memory (and deselect onboard memory) for the first 4K of the address space. Do not call this routine when the stack is in the first 4K of memory.

PXONBD (F42A) (Enable Onboard Memory)

Call this address to select onboard memory (and deselect offboard memory) for the first 4K of the address space. Do not call this routine when the stack is in the first 4K of memory.

Note that the Poly-88 ROMs are accessible at address 0000 when onboard memory is enabled.

PTMRISR (F7FA) (Timeout ISR)

(Initialized to “ret” instruction.)User code may insert up to 3 bytes here, to jump to a timer interrupt service routine when PXTIME overflows. The interrupt service routine must be in memory above FFFh. All registers can be used by the interrupt service routine. Execute a “ret” instruction when done.

PXTIME (F7FD) (60 Hz Count-Up Timer)

(Do not call this address.) Read a 16-bit value at this address to get the current value of the count-up timer. Write a 16-bit value this address to preset the timer.

This timer is incremented by an interrupt service routine, which is triggered by a 60 Hz signal. Interrupts that occur while onboard memory is selected will be missed. When CP/M is running, this will occur occasionally only while calling PXCOUT.

When the timer overflows (wraps to 0000h), the PXOVFL byte will be set to 1.

PXOVFL (F7FF) (TIMER OVERFLOW FLAG)

(Do not call this address.) Write 0 to this address immediately after presetting PXTIME. Read this flag to determine if the timer has overflowed - non-zero means it has overflowed.

Timer Example:

To set the timer for a 10-second timeout:

60 X 10 = 600 timer ticks

1. Write PXTIME = (0 - 600) = 0FDA8h
2. Write PXOVFL = 0
3. Wait for PXOVFL <>0

Alternatively, install a “jmp MY_ISR” at PTMRISR, and set up the timer as above. When the timer overflows, the CPU will be interrupted, and execution will continue at MY_ISR.

EXTENSION MODULE INTERFACE

Polex provides an interface for an Extension Module in ROM socket 2, at address 0800h.

If Polex detects a ROM at 0800h, it will call this address, allowing the extension module to perform any necessary initialization. Normally, this initialization includes copying the extension module’s code into high memory, so that it is available once Polex disables onboard memory.

Upon return, the Extension Module should return the address of its Command Processor in HL. If the Extension Module does not have any user commands, then it should return with H=0.

Polex will call the returned address (if it is not 0) after each command line from the user, before searching its own command list.

On Command Processor entry, DE=the address of an input line, with the first byte being a potential first command character (i.e. leading spaces have already been skipped).

The Extension Module should return with the Z flag set if it does not recognize the command from the user, allowing Polex to search its own command list.

If the Extension module recognizes the command, it should execute the command, and then return with the Z flag cleared.

Alternatively, the Extension Module can return by jumping to PXMAIN, where the stack will get repaired. (This makes a convenient error exit.)

Interrupts and Stack Usage

Polex handles all Poly-88 interrupts, which are required for input and output, as well as for the Real Time Clock. Polex prevents interrupts from nesting while using the interrupted program’s stack. When taking an interrupt, Polex requires four bytes of stack space (2 pushes), including the return address to the interrupted program.

When calling Polex entry points, Polex requires 4 stack bytes (2 pushes), including the return address of the calling program. In addition, Polex may take an interrupt, requiring an additional 6 bytes of stack space, for a total of 10 bytes (5 pushes).

Since keyboard and serial port I/O are interrupt driven, it is not a good idea to mask interrupts for too long, as incoming characters will get dropped. At 9600 baud, characters arrive every millisecond. This is a practical upper limit for the time interrupts should be masked during serial transfer (since the Poly-88 UART works best with baud rates at or below 9600 baud).

User experience suffers when keyboard response is longer than about 15 mS. This is a practical upper limit for the time interrupts should be masked when expecting user input.

The Real-Time Clock interrupt occurs every 16.6 mS. This is the upper limit for the time interrupts should be masked when using the Real-Time Clock to time anything.

Note that many disk subsystems require interrupts to be masked while reading or writing data. As an example, the Micropolis disk subsystem must mask interrupts for about 12.5 mS to read or write a sector.

Cassette Tape Interface

Poly-88 cassette tapes may be accessed via the Polex interface. The Poly-88 supports two cassette tape formats: “Byte-Format” (300 baud, asynchronous) and “Poly-Format” (2400 baud, synchronous).

To set the serial port up for Byte-Format loading, just set the baud rate to 300 baud and select the cassette tape port, by calling PXBAUD with A=06h.

To set up the serial port for Poly-Format loading, perform these operations:

1. Set the baud rate to 2400 baud and select the cassette tape port by calling PXBAUD with A=05h (this is not the normal 2400 baud setting because the UART will be set up in divide-by-one mode.)

2. Output the following sequence to port 01, which is the UART control port:
    a. 0AAh ;fake sync character to terminate any modes
    b. 40h ;select control port
    c. 0Ch ;select synchronous mode, divide-by-one clock
    d. 0E6h ;first sync character
    e. 0E6h ;second sync character

00h ;receiver off for now

Polex Source Code Listing

```
;===============================================================
; POLEX MONITOR EXTENSION FOR THE POLYMORPHIC SYSTEMS POLY-88
;
; POLEX RESIDES IN THE SECOND ROM SOCKET OF A POLY-88 COMPUTER,
; WITH THE STANDARD (REVISION 4.0) POLY-88 MONITOR IN THE FIRST
; ROM SOCKET. THE POLY-88 MONITOR AUTOMATICALLY INVOKES POLEX
; DURING INITIALIZATION.
;
; POLEX RELOCATES ITSELF TO HIGH MEMORY, AND DISABLES THE
; POLY-88 ONBOARD MEMORY, REPLACING IT WITH PHANTOM-ENABLED
; RAM, SO THAT PROGRAMS LIKE CP/M CAN RUN IN THE POLY-88,
; WITHOUT MAJOR MODIFICATIONS TO THE POLY-88.
;
; POLEX PROVIDES BASIC MONITOR COMMANDS FOR EXAMINING AND
; MODIFYING MEMORY, AS WELL AS AN INTEL HEX LOADER.
;
; IMPORTANTLY, POLEX PROVIDES INTERRUPT-DRIVEN CONSOLE AND
; SERIAL PORT ROUTINES FOR E.G. CP/M, WITH STANDARDIZED ENTRY
; POINTS. BOTH TRANSMIT AND RECEIVE SERIAL PORTS HAVE LARGE
; ENOUGH QUEUES TO KEEP UP EVEN WHILE THE POLY-88 CONSOLE IS
; SCROLLING (WHICH IT DOES THE HARD WAY).
;
; POLEX IS EXTENSABLE VIA MODULES (E.G. A DISK MODULE) IN THE
; THIRD ROM SOCKET ON THE CPU BOARD, TO PROVIDE FURTHER
; FUNCTIONALITY.
;
; WHEN POLEX IS RUNNING, THE POLY-88 FRONT PANEL MODE IS
; DISABLED, BECAUSE ITS STACK USAGE BLOWS AWAY PROGRAMS LIKE
; CP/M, WHERE THE STACK MAY BE IN THE LOW 4K OF MEMORY.
;
; THE POLY-88 CASSETTE TAPE INPUT ROUTINES CAN BE ACCESSED
; VIA THE 'PO' COMMAND.
;
; POLEX EXPECTS THE POLY-88 SERIAL PORTS TO BE CONFIGURED IN
; THE STANDARD WAY, WITH THE PRINTER PORT AT "DEVICE ADDRESS"
; 1, AND THE CASSETTE TAPE PORT AT "DEVICE ADDRESS" 0. (DEVICE
; ADDRESSES ARE SET BY JUMPERS ON THE POLY-88'S SERIAL PORT
; BOARDS THAT ATTACH TO THE REAR PANEL OF THE POLY-88 AND
; CONNECT THE ITS CPU BOARD WITH 16-PIN RIBBON CABLES.)
;
; POLEX REQUIRES THE MODS RECOMMENDED IN THE POLY-88 MANUAL,
; IN THE SECTION CALLED 'MEMORY FUNCTIONS OF THE POLY 88 CPU,'
; BUT CORRECTED. SPECIFICALLY, THE FOLLOWING MODS ARE REQUIRED:
;   1) CUT TRACE HH
;   2) JUMPER FROM THE RIGHT SIDE OF HH TO PAD H
;   3) JUMPER FROM S-100 PIN 67 (PHANTOM) TO PAD J
;
; FORMATTED TO ASSEMBLE WITH DIGITAL RESEARCH'S ASM.
;
;===============================================================
; POLEX COMMANDS (ALL VALUES ARE IN HEX):
;
; DU <ADR1> <ADR2>       DUMP MEMORY FROM <ADR1> THROUGH <ADR2>
;
; EN <ADR>               ENTER HEX TO MEMORY AT <ADR>
;                        QUIT EN COMMAND WITH A BLANK LINE
;
; EX <ADR>               EXECUTE AT <ADR>
;
; FI <ADR1> <ADR2> <VAL> FILL MEMORY FROM <ADR1> THROUGH <ADR2>
;                        WITH <VAL>
```

```
;
; HL [0/1]                LOAD INTEL HEX FILE TO MEMORY.
;                         0=NO ECHO (DISPLAY PACIFIER PER LINE).
;                         1=ECHO INCOMING DATA. DEFAULT (NO PARAM)
;                         IS NO ECHO (REQ'D FOR 9600 BAUD).
;
; PO                      RETURN TO POLY-88 MONITOR LOOP
;
; SB <BAUD>               SET SERIAL BAUD RATE AND PORT
;                         LSD OF <BAUD> SELECTS THE BAUD RATE
;                         MSD OF <BAUD> SELECTS THE PORT
;
;                         BAUD RATES:
;                           0: DISABLED    8: 900 BAUD
;                           1: 50 BAUD     9: 1200 BAUD
;                           2: 75 BAUD     A: 1800 BAUD
;                           3: 110 BAUD    B: 2400 BAUD
;                           4: 134.5 BD    C: 3600 BAUD
;                           5: 150 BAUD    D: 4800 BAUD
;                           6: 300 BAUD    E: 7200 BAUD
;                           7: 600 BAUD    F: 9600 BAUD
;
;                         PORTS:
;                           0: CASSETTE TAPE INTERFACE
;                           1: RS232 PORT
;
; TE                      TERMINAL MODE (ALT-MODE TO EXIT)
;
;=================================================================
; INTERFACES
;-------------------------------------------------------------
; POLEX PROVIDES THE FOLLOWING FIXED-LOCATION ENTRY POINTS FOR
; SOFTWARE ACCESS:
;
; PXINIT  (F400)   RESTART POLEX
;
; PXCSTA  (F403)   CONSOLE STATUS
;   A=0 & Z SET IF NOT READY (NO CONSOLE KBD CHR WAITING).
;   A=FF & Z CLEAR IF READY (CONSOLE KBD CHR WAITING)
;   ALL OTHER REGISTERS PRESERVED.
;
; PXCIN   (F406)   CONSOLE INPUT
;   WAIT FOR CONSOLE KBD CHARACTER, AND RETURN IT IN A.
;   ALL OTHER REGISTERS PRESERVED.
;
; PXCOUT  (F409)   CONSOLE OUTPUT
;   PRINT CHARACTER IN C ON CONSOLE. ON RETURN, A=C
;   ALL OTHER REGISTERS PRESERVED.
;
; PXSSTA  (F40C)   SERIAL INPUT PORT STATUS
;   A=0 & Z SET IF NOT READY (THE INPUT QUEUE IS EMPTY).
;   A=FF & Z CLEAR IF READY (THE INPUT QUEUE IS NOT EMPTY)
;   ALL OTHER REGISTERS PRESERVED.
;
; PXSIN   (F40F)   READ FROM SERIAL PORT INPUT
;   WAIT FOR SERIAL PORT CHR, AND RETURN IT IN A.
;   ALL OTHER REGISTERS PRESERVED.
;
; PXSOUT  (F412)   WRITE TO SERIAL PORT OUTPUT
;   SEND CHR IN C TO SERIAL PORT. ON RETURN, A TRASHED
;   ALL OTHER REGISTERS PRESERVED.
;
; PXSOST  (F415)   SERIAL PORT OUTPUT STATUS
;   A=0 & Z SET IF NOT READY (THE OUTPUT QUEUE IS FULL).
;   A=FF & Z CLEAR IF READY (THE OUTUT QUEUE IS NOT FULL)
;   ALL OTHER REGISTERS PRESERVED.
;
; PXBAUD  (F418)   SET SERIAL BAUD RATE AND PORT
;   A=BAUD RATE. (SEE ABOVE TABLE.) TRASHES A.
;
; PXIHEX  (F41B)   GET CONSOLE HEX INPUT
;   SCAN PAST BLANKS, RETURN HEX VALUE IN A
;   ON ENTRY: DE POINTS TO INPUT STRING, TERMINATED BY
;   POLY-88 CURSOR (FFH).
;   ON EXIT: HL=VALUE, DE ADVANCED PAST HEX VALUE, A TRASHED
;   Z  SET, CARRY CLEAR IF NO VALUE FOUND
;   BAD HEX ABORTS TO MAIN.
;
; PXOHEX  (F41E)  PRINT A AS HEX ON THE CONSOLE
;   ALL REGISTERS PRESERVED
;
; PXWARM  (F421)   WARM-START RE-ENTRY
;   MAIN LOOP - RESTORES STACK, PRINTS PROMPT
;
; PXCIN2  (F424)  GET CONSOLE CHR, IF AVAILABLE
;   Z SET IF NO CHR READY, A=CHR IF READY.
;   'ALT-MODE' KEY ABORTS TO MAIN
;
; PXOFBD (F427)  ENABLE OFFBOARD MEMORY. DO NOT CALL THIS WHEN
;   THE STACK IS IN THE 1ST 4K OF MEMORY.
;
; PXONBD (F42A)  ENABLE ONBOARD MEMORY. DO NOT CALL THIS WHEN
;   THE STACK IS IN THE 1ST 4K OF MEMORY.
;
; PTMRISR (F7FA)  THREE BYTE ISR CALLED WHEN THE 60-HZ TIMER
;   OVERFLOWS. INITIALIZED TO "RET". USER CODE CAN MODIFY TO
;   A "JMP USERISR" INSTRUCTION. ISR LOCATION MUST BE >= 1000H.
;   COMPUTER IS "ONBRD", ALL REGISTERS CAN BE USED.
;
; PTIMER  (F7FD)  READ THIS 2-BYTE LOCATION TO GET THE CURRENT
;   VALUE OF THE 60-HZ RTC COUNT-UP TIMER. WRITE THIS
;   LOCATION TO SET THE TIMER.
;
; PTOVFL (F7FF) READ THIS VALUE TO SEE IF THE TIMER OVERFLOWED.
;   <>0 MEANS THAT IT DID OVERFLOW. wRITE 0 TO THIS VALUE TO
;   RESET THE OVERFLOW FLAG.
;
;--------------------------------------------------------------
; POLEX PROVIDES AN INTERFACE FOR AN EXTENSION MODULE LOCATED
; IN ROM AT ADDRESS 0800, WITH COMMAND PROCESSOR EXECUTION AT
; AN ADDRESS PROVIDED BY THE MODULE DURING INITIALIZATION.
;
; INITIALIZATION:
;
; POLEX CHECKS FOR A MODULE ROM AT ADDRESS 0800 DURING ITS
; INITIALIZATION, AND CALLS THIS ADDRESS IF A ROM IS FOUND. THE
; MODULE SHOULD PERFORM ANY REQUIRED INITIALIZATION, (INCLUDING
; RELOCATING ITS COMMAND PROCESSOR INTO RAM) AND RETURN TO
; POLEX WITH THE ADDRESS OF ITS COMMAND PROCESSOR IN HL.
;
; IF A MODULE HAS NO COMMAND PROCESSOR (E.G. A DRIVER MODULE),
; THEN IT SHOULD RETURN H=0. NOTE THAT A MODULE'S COMMAND
; PROCESSOR MUST NOT BE IN THE FIRST 4K OF MEMORY.
;
; COMMAND PROCESSOR:
;
```

```
; IF A MODULE ROM WITH A COMMAND PROCESSOR WAS FOUND DURING
; INITIALIZATION, THEN POLEX WILL CALL THE MODULE'S COMMAND
; PROCESSOR (AT THE ADDRESS THAT WAS IN HL WHEN THE MODULE'S
; INITIALIZATION COMPLETED) ON EACH LINE OF USER INPUT, BEFORE
; CHECKING ITS OWN COMMAND LIST.
;
;  ON MODULE COMMAND PROCESSOR ENTRY:
;    HL POINTS TO AN INPUT LINE, WITH THE FIRST BYTE BEING
;    THE FIRST CHARACTER OF A POTENTIAL 2-CHARACTER COMMAND
;
;  ON MODULE COMMAND PROCESSOR EXIT:
;    Z SET IF THE MODULE DID NOT RECOGNIZE THE COMMAND.
;    Z CLEARED IF THE COMMAND WAS TAKEN BY THE MODULE.
;    MODULE COMMANDS MAY ALSO RETURN BY JUMPING TO PXWARM,
;    WHERE THE STACK WILL GET REPAIRED. (THIS MAKES A
;    CONVENIENT ERROR-EXIT.)
;
;=================================================================
; NOTES
;
; AFTER ASSEMBLY, YOU MUST CHECK:
; 1)  KBUFF+ROF < SRQ
; 2)  KBUFF < 0800H
; (THESE WILL CREATE ASSEMBLY ERRORS, DUE TO TESTS AT THE END.)
;
; POLEX IS AUTOMATICALLY INVOKED BY THE POLY-88 MONITOR ROM
; (VERSION 4.0), AFTER ITS INITIALIZATION.
;
; POLEX'S INSTALL CODE RELOCATES MOST OF THIS PROGRAM INTO HIGH
; MEMORY, AND THEN SWITCHES OFF THE ON-BOARD MEMORY (WHICH
; SWITCHES ON THE OFF-BOARD MEMORY). THIS WILL ALLOW OPERATING
; SYSTEMS (SUCH AS CP/M) THAT REQUIRE RAM STARTING AT ADDRESS
; 0000 TO RUN ON THE POLY-88.
;
; ALL ADDRESSES WITHIN THE RELOCATED CODE ARE OFFSET BY 'ROF',
; SO THAT THEY ARE CORRECT ONCE RELOCATED. THE ASSEMBLER'S
; LISTING OUTPUT (POLEX.PRN) SHOWS ADDRESSES FOR THE CODE
; BEFORE IT GETS RELOCATED, SO READING THE LISTING IS TRICKY.
;
; THE POLY-88 REAL-TIME CLOCK, KEYBOARD AND SERIAL PORT ARE
; INTERRUPT DRIVEN. POLEX INSTALLS INTERRUPT VECTORS IN ALL 8
; RESTART ADDRESSES IN OFFBOARD RAM. MOST OF THESE INTERRUPTS
; SHOULD NEVER OCCUR, AND JUST JUMP TO THE POLY-88 MONITOR. THE
; RTC, KEYBOARD, AND UART VECTORS ULTIMATELY JUMP TO SERVICE
; ROUTINES IN THE POLEX ROM.
;
; WHEN RUNNING IN ONBOARD MEMORY, THE KEYBOARD AND UART
; INTERRUPT ROUTINES ARE REPLACED BY SUBROUTINES THAT CALL
; THE POLEX SERVICE ROUTINES. THIS RELACEMENT IS VIA THE POLY-88
; 'WORMHOLES'.
;
; NOTE THAT THE RTC INTERRUPT WILL OCCUR 60 TIMES PER SECOND-
; THERE IS NO WAY TO DISABLE IT IN SOFTWARE. THE INTERRUPT
; MUST BE CLEARED (BY WRITING ANYTHING TO THE RTC PORT)
; BEFORE RETURNING FROM THAT INTERRUPT. HOWEVER, (UNLIKE THE
; KEYBOARD AND UART INTERRUPTS), THERE IS NO 'WORMHOLE' THAT
; ALLOWS REPLACEMENT OF THE RTC SERVICE ROUTINE. THIS MEANS
; THAT RTC INTERRUPTS THAT OCCUR WHILE ONBOARD MEMORY IS ENABLED
; WILL NOT BUMP THE TIMER COUNTERS IN POLEX MEMORY, AND THE
; TIMER WILL APPEAR TO BE RUNNING TOO SLOWLY.
;
; POLEX CALLS THE POLY-88 VIDEO DRIVER, DSPLY (IN ONBOARD
; MEMORY), FOR ITS CONSOLE OUTPUT. ON-BOARD MEMORY IS SWAPPED
; IN AND OUT AS REQUIRED. INTERRUPTS ARE ENABLED DURING DSPLY,
; BECAUSE IT TAKES MANY 9600 BAUD CHARACTER TIMES TO SCROLL
; THE SCREEN.
;
; THE POLEX UART ROUTINES HAVE BIG ENOUGH QUEUES THAT SERIAL
; TRANSFER CAN OCCUR WITHOUT HANDSHAKING, EVEN DURING POLY-88
; VIDEO SCROLLING (WHICH IT DOES THE HARD WAY). (AT 9600 BAUD,
; THE SERIAL TRANSFER CAN ONLY KEEP UP IF THE SENDER SENDS 2
; STOP BITS WITH EACH CHARACTER, OR IF A 5 mS MINIMUM DELAY
; IS INSERTED AFTER EACH LINE.)
;
; NOTE THAT THE UART ON THE CPU CARD IS ONLY READABLE WHILE
; ONBOARD MEMORY IS ENABLED VIA THE BRG PORT. HOWEVER, wRITING
; TO THE UART WORKS WHETHER OR NOT ONBOARD MEMORY IS ENABLED.
;
; POLEX DOES NOT SUPPORT BAUD RATES ABOVE 9600 BAUD, WHICH
; REQUIRE USE OF THE DIVIDE-BY-ONE CLOCK DIVIDER. THE 8251
; (PLAIN) UART DOES NOT DO WELL WITH THESE BAUD RATES.
;
; THE POLY-88 FRONT PANEL MODE IS DISABLED BECAUSE OF ITS
; EXCESSIVE STACK USAGE, AND BECAUSE IT IS INCOMPATIBLE WITH
; PROGRAMS (SUCH AS CP/M) THAT LOCATE THE STACK IN THE FIRST
; 4K OF RAM.
;
; CP/M AND CP/M PROGRAMS MAY HAVE LIMITED STACK SPACE FOR USE BY
; CALLS TO I/O ROUTINES, AND FOR USE BY INTERRUPTS. POLEX IS
; WRITTEN TO MINIMIZE STACK USAGE FOR CP/M COMPATIBILITY. WORST-
; CASE STACK USAGE OCCURS WHEN A UART INTERRUPT COMES WHILE IN
; ONE OF THE I/O ROUTINES THAT MAY BE CALLED BY CP/M:
;
; BYTES   SOURCE
;  2      RETURN ADDRESS FOR CP/M'S I/O ROUTINE CALL
;  2      WORST-CASE POLEX I/O ROUTINE (WHILE INTS ENABLED)
;  2      INTERRUPT RETURN ADDRESS
;  2      PUSH H IN VECTOR
;  2      PUSH PSW IN VECTOR
;
; 10 BYTES TOTAL
;
; NOTE ALSO THAT CALLING PROGRAMS (SUCH AS CP/M) MAY HAVE
; THEIR STACKS IN THE 1ST 4K OF MEMORY, WHICH WILL DISAPPEAR
; WHEN WE SWITCH TO ONBOARD MEMORY. THEREFORE A LOCAL STACK
; IS SWAPPED IN FOR INTERRUPTS AND FOR CALLS TO DSPLY, BOTH
; OF WHICH USE ONBOARD CODE.
;
; NOTE THAT OKINT AND OUINT WILL GET CALLED ONLY WHEN AN
; INTERRUPT OCCURS WHILE ONBOARD MEMORY IS ENABLED. WHILE CP/M
; IS RUNNING, THIS WILL ONLY HAPPEN DURING CALLS TO PXCOUT,
; WHICH USES THE POLY-88 ROM'S DISPLAY DRIVER, DSPLY. PXCOUT
; SWAPS IN A LOCAL STACK BEFORE ENABLING ONBOARD MEMORY.
;=================================================================
; REVISION HISTORY
; VERS. 1.00 BY M. EBERHARD 30 OCT 2013
;   CREATED
;
; VERS. 1.01 BY M. EBERHARD 22 NOV 2013
;   ASSEMBLY OPTION (NOT SELECTED) TO MOVE THE DISK CONTROLLER
;   TO THE STANDARD MICROPOLIS ADDRESS F800 AND MOVE VIDEO
;   MEMORY TO FC00. ADJUST I/O ROUTINES & ENTRY POINTS TO MATCH
;   CP/M NEEDS.
;
; VERS. 1.02 BY M. EBERHARD 23 NOV 2013
;   REPLACE KBD INT IN ONBOARD RAM TO SUBSTITUTE CONTROL-A
;   FOR CONTROL-Z, SINCE CP/M USES CONTROL-Z
;
; VERS. 2.00 M. EBERHARD 23 NOV 2013
;   SPLIT OFF MICMOD ROM, MOVING ALL DISK COMMANDS THERE
;   ADD SERIAL PORT SUPPORT & VECTORS
;
; VERS. 2.01 M. EBERHARD 4 DEC 2013
;   ADD TE COMMAND (SO THE SERIAL PORT CAN BE TESTED)
;   DEBUG SERIAL PORT INTERRUPT CODE
;
; VERS. 2.02 M. EBERHARD 6 DEC 2013
;   USE MODULE EXEC ADDRESS RETURNED DURING INIT (SO A MODULE
;   COULD RELOCATE ITSELF ANYWHERE)
;
; VERS. 3.00 M. EBERHARD 16 DEC 2013
;   ELIMINATE SUPPORT FOR FRONT PANEL MODE, REDUCE STACK USAGE
;   (INCLUDING IN INTERRUPTS) FOR CP/M AND OTHER PROGRAMS WITH
;   SMALL STACKS. ELIMINATE ECHO-CONTROL DURING HEX LOAD.
;
; VERS. 3.01 M. EBERHARD 19 DEC 2013
;   FIX FOR INTERRUPTED PROGRAMS WITH STACKS IN THE 1ST 4K OF
;   MEMORY (WHICH DISAPPEAR WITH ONBRD).
;
; VERS. 3.02 M. EBERHARD 20 DEC 2013
;   ADD RTC COUNT-DOWN TIMER
;
; VERS. 3.03 M. EBERHARD 24 DEC 2013
;   SOME CODE SQUEEZING, DELETE HIGHER BAUD RATE SUPPORT IN
;   STBAUD, ALLOW SERIAL PORT (CASSETTE VS PRINTER) TO BE
;   SELECTED BY STBAUD/PXBAUD. ADD ADDRESS TO EACH LINE IN EN
;   COMMAND. CHANGE TIMER TO COUNT UP, AND ADD OVERFLOW FLAG.
;   TRANSLATE BS TO DEL IN CPRINT. PROVIDE EXTERNAL ACCESS TO
;   ONBRD AND OFFBRD SUBROUTINES.
;
; VERS. 3.04 M. EBERHARD 03 JUN 2014
;   REV F POLY-88 CPU BOARDS HAVE THE ROM DISABLE BIT (BRGOMD)
;   AT BIT 6 OF THE BRG PORT, INSTEAD OF BIT 5 (AS ON REV 0
;   BOARDS). THIS VERSION SUPPORTS BOTH. NOTE THAT BIT 7 OF
;   THIS PORT SEEMS TO DO SOMETHING ON THE REV F BOARD: WRITING
;   1 TO THIS BIT HANGS POLEX.
;
; VERS. 3.05 M. EBERHARD 26 APR 2016
;   STRIP KEYBOARD PARITY, ACCEPT ESC AS COMMAND ABORT KEY TOO.
;   USE ASSEMBLER TO CHECK FOR KBUFF SIZE/PLACEMENT ERRORS.
;   EX COMMAND DEFAULTS TO ADDRESS 0 (TO SAVE CODE SPACE).
;
; VERS. 3.06 M. EBERHARD 27 APR 2016
;   SQUEEZE CODE, CONVERT LOWERCASE ON COMMAND LINE TO UPPER-
;   CASE
;
; VERS. 3.07 M. DOUGLAS & M. EBERHARD 12 JUNE 2016
;   USE THE RTC INTERRUPT TO ENABLE THE UART TRANSMITTER SO
;   THAT MULTI-CHARACTER TRANSMISSIONS HAVE A CHANCE TO QUEUE
;   AND, IN TURN, PREVENT TRANSMITTER SHUTDOWN BETWEEN EACH
;   CHARACTER. MOVE EI FROM THE CHARACTER I/O SUBROUTINES INTO
;   THE CORRRESPONDING xxSTAT SUBROUTINES. THIS ENSURES EXTERNAL
;   CALLS TO xxSTAT BEHAVE AS EXPECTED. HEX LOAD MODIFIED TO
;   SQUEEZE CODE. RECORD TYPE AND ASCII TO HEX CONVERSION ARE
;   NO LONGER CHECKED. RECORD CHECKSUM REMAINS. UPDATE CODE
;   SIZE CHECKS AT THE END OF THIS FILE. CONVERT KBD BS TO DEL.
;   TRANSLATE BS TO DEL FOR TE COMMAND TOO.
;
; VERS. 3.08 M. EBERHARD 15 JUNE 2016
```

```
                ;   SPEED UP HL A TAD. DON'T ECHO SERIAL DATA DURING HL IF THE
                ;   RX QUEUE IS NEARLY FULL. (MAKES 9600 BAUD MOSTLY WORK.)
                ;
                ; VERS. 3.09 M. EBERHARD 18 JUNE 2016
                ;   BACK OUT ECHO LOGIC FROM 3.08. ADD ECHO PARAMETER TO HL, SO
                ;   THAT "HL 0" WILL PRINT A PACIFIER PER RECORD, RATHER THAN
                ;   ECHOING THE RECEIVED DATA. (REQUIRED FOR 9600 BAUD.)
                ;
                ; VERS. 3.10  M. EBERHARD 22 JUNE 2016
                ;   INITIALIZE POLY-88'S POS VARIABLE DURING INIT.
                ;
                ; VERS. 3.11  M. DOUGLAS & M. EBERHARD 11 NOV 2019
                ;   CALL PTMRISR WHEN PTIMER OVERFLOWS. PTMRISR IS A THREE BYTE
                ;   FIXED LOCATION VARIABLE. THE FIRST BYTE IS INITIALIZED TO
                ;   A RET INSTRUCTION. USER CODE CAN MODIFY PTMRISR TO INSTEAD
                ;   JUMP TO A USER ISR. ISR LOCATION MUST BE >= 1000H. COMPUTER
                ;   IS "ONBRD" UPON ISR ENTRY. ALL REGISTERS CAN BE USED.
                ;   CHANGED THE HL COMMAND TO DEFAULT TO NO ECHO.
                ;===============================================================
                ;**********
                ;DEBUG FLAG
                ;**********
0000 =          DEBUG   EQU     0       ;1: "PROM" IS IN RAM AT E000

                ;*****
                ;ASCII
                ;*****
0008 =          BS      EQU     08H     ;BASKSPACE
0009 =          TAB     EQU     09H     ;TAB. MOVES CURSOR TO EVEN/8 POS
000B =          VT      EQU     0BH     ;VERTICAL TAB. JUST HOMES CURSOR.
000C =          FMFD    EQU     0CH     ;FORM FEED. CLEARS SCRN, HOMES CURSOR
000D =          CR      EQU     0DH
0018 =          CTLX    EQU     18H     ;ASCII "CAN" CLEARS CUR. LINE
001A =          CTLZ    EQU     1AH     ;CONTROL-Z ENTERS FRONT PANEL MODE
001B =          ESC     EQU     1BH     ;ESCAPE
007F =          DEL     EQU     7FH     ;DELETE
00FF =          CURSOR  EQU     0FFH    ;POLY-88 CURSOR

                ;***************
                ;PROGRAM EQUATES
                ;***************
003E =          PROMPT  EQU     '>'     ;PROMPT CHARACTER
007D =          CABKEY  EQU     '}'     ;COMMAND ABORT CHARACTER. NOTE: CODE
                                        ;..ASSUMES THIS IS > ASCII'z'.
                                        ;(POLY-88 ALT-MODE KEY)
0020 =          PAUKEY  EQU     ' '     ;PAUSES DUMPING IN DU COMMAND
002E =          PCFIER  EQU     '.'     ;HL PACIFIER FOR WHEN ECHO DISABLED

001C =          MAXBD   EQU     1CH     ;MAX ALLOWED BAUD RATE

001F =          DEBAUD  EQU     1FH     ;DEFAULT 9600 BAUD, PRINTER PORT

                ;********************************************
                ;CODE LOCATIONS BEFORE & AFTER CODE IS MOVED
                ;********************************************
                 IF DEBUG
                ROMLOC  EQU     0E000H  ;PROM IN RAM FOR DEBUG
                 ENDIF

                 IF NOT DEBUG
0400 =          ROMLOC  EQU     0400H   ;LOCATION IN ROM
                 ENDIF
```

```
F400 =          RAMLOC  EQU     0F400H  ;LOCATION AFTER MOVED
0800 =          MODROM  EQU     00800H  ;ADDR OF OPTIONAL ROM MODULE

                ;******
                ;STACKS
                ;******
001E =          VSSIZE  EQU     30      ;LOCAL STACK SPACE FOR PRINTC
                                        ;ROOM FOR AT LEAST 28 BYTES

0FF8 =          ISTACK  EQU     0FF8H   ;INTERRUPT STACK IN ONBOARD RAM

                ;************************
                ;FIXED-LOCATION VARIABLES
                ;************************
F7F7 =          FXDRAM  EQU     RAMLOC+3F7H

                ;*******************************************************
                ;UART QUEUES AND POINTERS
                ;
                ;NOTE THAT THE CODE MAKES SIGNIFICANT ASSUMPTIONS ABOUT
                ;THE SIZE AND ALIGNMENT OF THE SERIAL PORT QUEUES IN RAM.
                ;WITH SRQSIZ=32, RQLOC MUST BE XXXX XXXX XX00 0000B
                ;WITH STQSIZ=8, TQLOC MUST BE XXXX XXXX XXXX 0000B
                ;*******************************************************
0020 =          SRQSIZ  EQU     32              ;RECEIVE QUEUE SIZE
                                                ;MUST BE POWER OF 2
0004 =          STQSIZ  EQU     4               ;TRANSMIT QUEUE SIZE
                                                ;MUST BE POWER OF 2

F780 =          RQLOC   EQU     RAMLOC+380H     ;SERIAL RX QUEUE LOCATION
F7A0 =          TQLOC   EQU     RAMLOC+3A0H     ;SERIAL TX QUEUE LOCATION

                ;**************************************
                ;POLY-88 CPU BOARD I/O PORT ASSIGNMENTS
                ;**************************************
0000 =          UARTD   EQU     00H     ;SERIAL DATA
0001 =          UARTS   EQU     01H     ;SERIAL STATUS PORT
0001 =          UTXRDY    EQU     01H     ;TRANSMITTER READY
0002 =          URXRDY    EQU     02H     ;RECEIVER READY
0004 =          UTXMTY    EQU     04H     ;TRANSMITTER EMPTY
004E =          UMOD16    EQU     4EH     ;8 BITS, 1 STOP, NO PARITY, X16
004D =          UMOD01    EQU     4DH     ;8 BITS, 1 STOP, NO PARITY, X1
0040 =          UCMDRE    EQU     40H     ;RESET UART
0026 =          UCRXEN    EQU     26H     ;ENABLE RX, DTR, RTS ON
0027 =          UCMDEN    EQU     27H     ;ENABLE TX & RX, DTR, RTS ON

0004 =          BRG     EQU     04h     ;BAUD RATE GENERATOR PORT
000F =          BRGBD     EQU     0Fh     ;BAUD RATE MASK
0010 =          BRGDEV    EQU     10h     ;0=CASSETTE, 1=RS232
0060 =          BRGOMD    EQU     60h     ;1=DISABLE ONBOARD MEMORY
                                          ;REV 0 BOARDS USE BIT 5
                                          ;REV F BOARDS USE BIT 6

0008 =          RTC     EQU     08H     ;OUTPUT ANYTHING TO THIS PORT
                                        ;..TO RESET RTC INTERRUPT

                ;**********************************
                ;POLYMORPHIC VIDEO BOARD PARAMETERS
                ;**********************************
F800 =          VBASE   EQU     0F800H  ;STANDARD VIDEO MEM ADDR
F800 =          VDMEM   EQU     VBASE   ;BEGINNING OF VIDEO MEMORY
0040 =          VDROW   EQU     40H     ;64 BYTES/ROW
0400 =          VDSIZ   EQU     0400H   ;1K SCREEN BUFFER
```

```
00F8 =          KBD     EQU     0F8H    ;KEYBOARD DATA

                ;************************
                ;POLY-88 MONITOR LOCATIONS
                ;************************

                ;ROM ROUTINES
0054 =          USRTSR  EQU     0054H   ;USART INTERRUPT SERVICE
010D =          KSR     EQU     010DH   ;KEYBOARD INTERRUPT SERVICE
0064 =          IORET   EQU     0064H   ;SHARED EXIT FOR INT INTS
007F =          DSPLY   EQU     007FH   ;FAMOUS DISPLAY ROUTINE

                ;RAM LOCATIONS

0C00 =          TIMER   EQU     0C00H   ;4-BYTE RTC COUNTER
0C0E =          POS     EQU     0C0EH   ;CURSOR POSITION
0C16 =          SRA4    EQU     0C16H   ;UART IN VECTOR
0C18 =          SRA5    EQU     0C18H   ;KBD INT VECTOR
0FFE =          POLYSP  EQU     0FFEH   ;STACK POINTER FOR RETURN TO
                                        ;POLY-88 ROM. ASSUMES RETURN ADDRESS
                                        ;IS ON THE POLY-88 STACK

                ;============================================================
                ; START-UP ROM CODE
                ; THE POLY-88 MONITOR CALLS INSTAL WHEN ITS OWN INITIALIZATION
                ; IS COMPLETE. THIS ROUTINE INSTALLS POLEX IN RAM AT RAMLOC AND
                ; THEN JUMPS TO THE NEWLY-INSTALLED CODE.
                ;============================================================
0400                    ORG     ROMLOC

0400 F3         INSTAL: DI

                ;INSTALL ONBOARD INTERRUPT VECTORS FOR NEW KEYBOARD AND UART
                ;INTS, IN CASE WE GET AN INTERRUPT WHILE ONBOARD

                 IF DEBUG
                        CALL    ONBRD                     ;REAL PROM IS ONBRD HERE
                 ENDIF

0401 212304             LXI     H,OKINT
0404 22180C             SHLD    SRA5
0407 2E29               MVI     L,OUINT AND 0FFH          ;IN SAME PAGE
0409 22160C             SHLD    SRA4

                ;THE POLY88 MONITOR DOES NOT INITIALIZE POS, ITS CURSOR POSITION.
                ;THE FIRST CALL TO DSPLY WILL CAUSE 7FH TO BE WRITTEN TO WHATEVER
                ;RANDOM ADDRESS POS POINTS TO. SO WE POINT POS TO A HARMLESS
                ;ADDRESS HERE.

040C 2170F7             LXI     H,RAMEND-RAMSRC+RAMLOC  ;END OF DEST
040F 220E0C             SHLD    POS                       ;POINT POS PAST CODE

                ;MOVE CODE INTO RAM, BACKWARDS, SO HL IS RIGHT AT THE END

0412 017003             LXI     B,RAMEND-RAMSRC           ;BYTE COUNT
0415 11FE07             LXI     D,RAMEND                  ;END OF RAM SOURCE

0418 1B         MOVER:  DCX     D
0419 2B                 DCX     H

041A 1A                 LDAX    D                         ;GET A BYTE
041B 77                 MOV     M,A                       ;PUT A BYTE
```

```
041C 0B                 DCX     B                       ;BUMP COUNT
041D 78                 MOV     A,B
041E B1                 ORA     C                       ;DONE?
041F C21804             JNZ     MOVER

               ;HL = EXECUTION ADDRESS
               ;GO EXECUTE MOVED CODE

0422 E9                 PCHL                    ;GO TO LOADED CODE

               ;=============================================================
               ; ROM SUBROUTINES THAT DON'T GET RELOCATED INTO RAM
               ; THESE ROUTINES RUN IN ROM, AFTER ONBRD HAS BEEN CALLED
               ;=============================================================

               ;***ROM INTERRUPT SERVICE ROUTINE**************
               ; REPLACEMENT CODE FOR KEYBOARD INTERRUPT
               ; WHILE ONBOARD MEMORY IS ENABLED, KEYBOARD
               ; INTERUPTS EXECUTE HERE.
               ; ON ENTRY:
               ;   ALL REGISTERS ARE ON THE STACK
               ; ON EXIT:
               ;   INTERRUPTS ENABLED
               ;**********************************************
0423 CD4C04    OKINT:   CALL    KINT            ;MUST CALL, SO WE CAN
0426 C36400             JMP     IORET           ;RESTORE REGS, ENABLE
                                                ;..INTS. RET

               ;***ROM INTERRUPT SERVICE ROUTINE*********************
               ; REPLACEMENT CODE FOR SERIAL INTERRUPT
               ; WHILE ONBOARD MEMORY IS ENABLED, UART INTERRUPTS
               ; EXECUTE HERE. INTERRUPTS ARE ENABLED ON EXIT, IN
               ; CASE WE GET MORE THAN ONE INTERRUPT WHILE ONBOARD,
               ; WHICH HAPPENS DURING DSPLY SCROLLING.
               ; ON ENTRY:
               ;   ALL REGISTERS ARE ON THE STACK
               ; ON EXIT:
               ;   INTERRUPTS ENABLED
               ;*****************************************************
0429 CD5204    OUINT:   CALL    UINT            ;MUST CALL, SO WE CAN
042C C36400             JMP     IORET           ;RESTORE REGS, ENABLE
                                                ;..INTS. RET

               ;***ROM INTERRUPT SERVICE ROUTINE********************
               ; RTC INTERRUPT SERVICE
               ; INCREMENTS 2-BYTE 60-HZ TIMER IN RAM. UPON OVERFLOW,
               ; PTOVFL IS SET TO NON-ZERO AND PTMRISR IS CALLED.
               ; INTERRUPTS STILL DISABLED ON EXIT
               ;
               ; ENABLES UART TRANSMITTER IF CHARACTERS ARE QUEUED
               ; AND THE TRANSMITTER IS PRESENTLY DISABLED.
               ;
               ; NOTE THAT RTC INTERRUPTS WHEN ONBRD WILL BE MISSED.
               ; WHEN RUNNING AN OS (E.G. CP/M), THIS WILL OCCUR
               ; OCCASIONALLY WHEN CALLING PXCOUT.
               ;****************************************************
042F D308      RINT:    OUT     RTC             ;CLEAR INTERRUPT

               ;ENABLE UART XMIT IF TX QUEUE NOT EMPTY, BUT UART XMIT DISABLED

0431 2165F7             LXI     H,STQCNT+ROF    ;TRANSMIT QUEUE SIZE
0434 7E                 MOV     A,M             ;ANY XMIT DATA QUEUED?
```

```
0435 2B                 DCX     H               ;(URTDIS) XMIT DISABLED?
0436 A6                 ANA     M               ;FF MEANS DISABLED BUT ENQUEUED
0437 CA3F04             JZ      DOTIMR          ;EMPTY OR NOT DISABLED?

043A 3E27               MVI     A,UCMDEN        ;ENABLE XMIT
043C D301               OUT     UARTS

043E 34                 INR     M               ;FF -> 0: REMEMBER XMIT ENABLED

               ;INCREMENT 60HZ COUNTER/TIMER

043F 21FDF7    DOTIMR:  LXI     H,PTIMER        ;BUMP 16-BIT COUNTER
0442 34                 INR     M
0443 C0                 RNZ

0444 23                 INX     H               ;MS BYTE
0445 34                 INR     M
0446 C0                 RNZ

0447 23                 INX     H               ;POINT TO PTOVFL
0448 75                 MOV     M,L             ;L=FF: INDICATE OVERFLOW

0449 C3FAF7             JMP     PTMRISR         ;CALL USER ISR AND EXIT

               ;***ROM INTERRUPT SERVICE ROUTINE****
               ; KEYBOARD INTERRUPT SERVICE
               ; ELIMINATES FRONT PANEL ACCESS
               ; INTERRUPTS STILL DISABLED ON EXIT
               ; TRASHES PSW
               ;*************************************
044C DBF8      KINT:    IN      KBD
044E 326FF7             STA     KBUFF+ROF       ;SAVE KBD CHR
0451 C9                 RET

               ;***ROM INTERRUPT SERVICE ROUTINE*****
               ; COMBINED UART INTERRUPT SERVICE
               ; INTERRUPTS STILL DISABLED ON EXIT
               ; TRASHES PSW, HL
               ;**************************************
0452 2166F7    UINT:    LXI     H,SRQCNT+ROF    ;RECEIVE QUEUE SIZE

0455 DB01               IN      UARTS           ;GET UART STATUS
0457 E602               ANI     URXRDY          ;RECEIVE INTERRUPT?
0459 C27B04             JNZ     URXINT          ;Y: GO RECEIVE

               ;NOT A RECIEVE INTERRUPT, SO MUST BE TRANSMIT

               ;FALL INTO UTXINT

               ;***ROM ROUTINE****************************
               ; UART TRANSMIT INTERRUPT SERVICE
               ; SEND A QUEUE CHR. IF THE QUEUE IS EMPTY,
               ; THEN DISABLE THE TRANSMITTER TO DISABLE
               ; FURTHER TRANSMIT INTERRUPTS.
               ; ON ENTRY:
               ;   HL=ADDRESS OF SRQCNT, WHICH IS STQCNT+1
               ;******************************************
045C 2B        UTXINT:  DCX     H               ;POINT TO STQCNT
045D 7E                 MOV     A,M             ;ANYTHING IN THE QUEUE?
045E B7                 ORA     A

045F CA7104             JZ      WTXMTY          ;N:SHUT IT DOWN
```

```
0462 35                 DCR     M               ;ONE FEWER BYTE IN QUEUE

               ;GET A QUEUE CHR AND SEND IT

0463 2A6DF7             LHLD    STQOPT+ROF
0466 7E                 MOV     A,M             ;GET QUEUE CHR
0467 D300               OUT     UARTD           ;AND TRANSMIT IT

               ;CIRCULAR-INC STQOPT. ASSUME SUPER-NICE QUEUE ALIGNMENT

0469 2C                 INR     L
046A 3EFB               MVI     A,STQSIZ XOR 0FFH
046C A5                 ANA     L
046D 326DF7             STA     STQOPT+ROF      ;H NEVER CHANGES

0470 C9                 RET

               ;WAIT FOR THE 8251 TRANSMITTER TO EMPTY
               ;OTHERWISE THE LAST CHR WILL BE CHOPPED OFF

0471 DB01      WTXMTY:  IN      UARTS
0473 E604               ANI     UTXMTY
0475 CA7104             JZ      WTXMTY

0478 C3A4F5             JMP     UTXDIS+ROF      ;GO DISABLE THE INTERRUPT

               ;***ROM ROUTINE*****************************
               ; UART RECEIVE INTERRUPT SERVICE
               ; RECEIVE AND ENQUEUE A UART CHR.
               ; IF THE QUEUE IS FULL, CHUCK THE CHR.
               ; ON ENTRY:
               ;    HL=ADDRESS OF SRQCNT
               ;*******************************************
047B 7E        URXINT:  MOV     A,M             ;CHECK SRQCNT
047C FE20               CPI     SRQSIZ          ;ROOM IN QUEUE?

047E DB00               IN      UARTD           ;RECEIVE CHR

0480 C8                 RZ                      ;NO ROOM: DONE

0481 34                 INR     M               ;BUMP Q CHR COUNT

0482 2A67F7             LHLD    SRQIPT+ROF      ;RX Q INPOINTER
0485 77                 MOV     M,A             ;ENQUEUE NEW CHR

               ;CIRCULAR-INC SRQIPT. ASSUME SUPER-NICE QUEUE ALIGNMENT

0486 2C                 INR     L
0487 3EDF               MVI     A,SRQSIZ XOR 0FFH
0489 A5                 ANA     L
048A 3267F7             STA     SRQIPT+ROF      ;H NEVER CHANGES

048D C9                 RET

               ;===============================================================
               ; RAM CODE, WHICH GETS MOVED INTO RAM BY THE START-UP CODE.
               ; "ROF" IS THE RAM ADDRESS OFFSET, WHICH IS THE DIFFERENCE
               ; BETWEEN ITS ADDRESS WHILE IN ROM AND ITS ADDRESS WHEN
               ; INSTALLED IN RAM.
               ;===============================================================

               RAMSRC:
EF72 =         ROF      EQU     RAMLOC-RAMSRC   ;RAM OFFSET
```

```
                ;*********************
                ; POLEX ENTRY POINTS *
                ;*********************

048E C360F4     PXINIT: JMP     INIT+ROF        ;COLD-START SYSTEM ENTRY POINT

                ;THE FOLLOWING 7 ENTRY POINTS ARE CALLED BY CP/M,
                ;AND HAVE BEEN DESIGNED WITH MINIMAL STACK USAGE.

0491 C314F7     PXCSTA: JMP     KSTAT+ROF       ;FOR CP/M: TEST CONSOLE KBD
                                                ;Z SET, 0 MEANS NO DATA
                                                ;Z CLEAR, A=FF IF DATA AVAIL
0494 C31AF7     PXCIN:  JMP     KDATA+ROF       ;FOR CP/M: GET CONSOLE CHR IN A
0497 C342F6     PXCOUT: JMP     PRINTC+ROF      ;FOR CP/M: SEND C TO CONSOLE
                                                ;A=C=CHR ON RET

049A C3ADF6     PXSSTA: JMP     SISTAT+ROF      ;FOR CP/M: TEST UART INPUT
                                                ;Z SET, A=0 IF NO DATA
                                                ;Z CLEAR, A=FF IF DATA
049D C3B6F6     PXSIN:  JMP     SIDATA+ROF      ;FOR CP/M: GET UART CHR IN A
04A0 C3D6F6     PXSOUT: JMP     SODATA+ROF      ;FOR CP/M: SEND C TO UART
04A3 C3CEF6     PXSOST: JMP     SOSTAT+ROF      ;FOR CP/M: GET UART OUTPUT STAT
                                                ;A=0 & Z SET IF NOT READY
                                                ;A=FF & Z CLEAR IF READY

04A6 C38FF5     PXBAUD: JMP     STBAUD+ROF      ;SET UART BAUD RATE & PORT

04A9 C3EFF4     PXIHEX: JMP     FNDHEX+ROF      ;SCAN PAST BLANKS, GET HEX VAL
                                                ;HL=VALUE
                                                ;DE ADVANCED PAST CHR
                                                ;Z SET, CARRY CLEAR IF VALUE
                                                ;CARRY SET IF NO VALUE FOUND
04AC C32BF6     PXOHEX: JMP     PAHEX+ROF       ;PRINT A AS HEX ON THE CONSOLE
04AF C3B3F4     PXWARM: JMP     MAIN+ROF        ;WARM-START RE-ENTRY
04B2 C3EEF6     PXCIN2: JMP     CHKKBD+ROF      ;GET KBD CHR IF ONE IS THERE
04B5 C36BF6     PXOFBD: JMP     OFFBRD+ROF      ;SELECT OFFBOARD MEMORY

                PXONBD:
                ;FALL INTO ONBRD

                ;***EXTERNAL SUBROUTINE********************************
                ; SELECT ONBOARD MEMORY
                ; NOTE: YOU CAN'T CALL THIS SUBROUTINE IF THE STACK IS
                ; (OR MIGHT BE) IN THE 1ST 4K OF MEMORY, AS THE STACK
                ; WILL DISAPPEAR WHEN SWITCHING ONBOARD.
                ;*****************************************************
04B8 F5         ONBRD:  PUSH    PSW
04B9 3AF9F7             LDA     BRGSAV
04BC E69F               ANI     BRGOMD XOR 0FFH ;ONBOARD MEMORY

04BE D304               OUT     BRG
04C0 F1                 POP     PSW
04C1 C9                 RET

                ;***INTERRUPT SERVICE ROUTINE***********************************
                ; DISPATCH TO SPECIFIC INTERRUPT ROUTINE
                ; ON ENTRY FROM A VECTOR:
                ;   INTS ARE DISABLED VIA THE INTERRUPT
                ;     (SP) = ORIGINAL HL
                ;   (SP+2) = ORIGINAL PSW
                ;   (SP+4) = RETURN ADDRESS FOR INTERRUPTED PROGRAM
                ;
                ; THIS SWITCHES TO A PRIVATE STACK, BECAUSE THE INCOMING STACK
                ; MIGHT BE IN THE 1ST 4K OF RAM, WHICH DISAPPEARS WITH ONBRD.
                ;
                ; UPON 'RET' TO THE INTERRUPT SERVICE ROUTINE:
                ;  STACK IS NOW ISTACK, IN ONBOARD RAM
                ;  (SP) = ISRRET+ROF FOR RETURN FROM THE ISR
                ;  (SP+2) = ORIGINAL SP VALUE
                ;  HL, PSW SAVED ON INCOMING STACK
                ;  ALL OTHER REGISTERS UNCHANGED
                ;  INTERRUPTS STILL DISABLED FROM THE INTERRUPT
                ;  ONBOARD MEMORY ENABLED
                ;
                ; INTERRUPTS REMAIN DISABLED UNTIL ISRRET, SO THAT WE WON'T GET
                ; ANY INTERRUPTS WHILE ONBOARD SERVICING ANOTHER INTERRUPT.
                ;***************************************************************
04C2 3AF9F7     DSPTCH: LDA     BRGSAV
04C5 E69F               ANI     BRGOMD XOR 0FFH ;ONBOARD MEMORY
04C7 D304               OUT     BRG             ;..NOW

04C9 22F20F             SHLD    ISTACK-6        ;ISR ADDR ONTO INT STACK
                                                ;AT (ISTACK-5) & (ISTACK-6)

04CC 210000             LXI     H,0             ;GET INCOMING SP
04CF 39                 DAD     SP              ;INTO HL

04D0 31F80F             LXI     SP,ISTACK       ;SWITCH TO INTERRUPT STACK

04D3 E5                 PUSH    H               ;SAVE INCOMING SP
                                                ;AT (ISTACK-1) & (ISTACK-2)

04D4 214DF4             LXI     H,ISRRET+ROF    ;CREATE RET ADDRESS
04D7 E5                 PUSH    H               ;AT (ISTACK-3) & (ISTACK-4)

04D8 3B                 DCX     SP              ;POINT TO ISR ADDRESS
04D9 3B                 DCX     SP
04DA C9                 RET                     ;GO TO ISR

                ;***INTERRUPT EXIT******************
                ; INTERRUPT SERVICE ROUTINE RETURN
                ; RESTORES OFFBOARD INTERRUPTED
                ; PROGRAM'S CONTEXT AND RETURNS THERE
                ;***********************************
04DB E1         ISRRET: POP     H               ;GET ORIGINAL SP

04DC 3AF9F7             LDA     BRGSAV          ;OFFBOARD MEMORY
04DF D304               OUT     BRG             ;..NOW

04E1 F9                 SPHL                    ;RESTORE ORIGINAL SP
04E2 E1                 POP     H               ;ORIGINAL HL
04E3 F1                 POP     PSW             ;ORIGINAL PSW
04E4 FB                 EI                      ;FINALLY ENABLE INTS
04E5 C9                 RET                     ;AND RETURN FROM INTERRUPT

                ;===========================================================
                ; PROTOTYPE 8-BYTE INTERRUPT VECTOR
                ; 8 OF THESE GET INSTALLED BY INIT, STARTING AT ADDRESS 0
                ; IN OFFBOARD MEMORY. THE DEFAULT INTERRUPT SERVICE
                ; ROUTINE IS GOPOLY, WHICH WILL RETURN TO THE POLY-88 MAIN
                ; LOOP. (YOU CAN THEN TYPE ^Z TO BRING UP THE FRONT-PANEL
                ; MONITOR TO DEBUG.) WE SHOULD ONLY EVER GET THE FOLLOWING
                ; INTERRUPTS: UART, KEYBOARD, AND RTC. INIT REPLACES GOPOLY
                ; WITH THE APPROPRIATE ROUTINE ADDRESSES FOR THESE THREE.
                ; SINCE WE GOT HERE FROM AN INTERRUPT, INTERRUPTS ARE
                ; ALREADY DISABLED.
                ;==========================================================
04E6 F5         PROTOV: PUSH    PSW             ;INCOMING PSW
04E7 E5                 PUSH    H               ;INCOMING HL VALUE

04E8 217BF5             LXI     H,GOPOLY+ROF    ;DEFAULT VECTOR

04EB C334F4             JMP     DSPTCH+ROF

                ;=============================
                ;= COLD-START INITIALIZATION =
                ;=============================
04EE F3         INIT:   DI                      ;INTS OFF WHILE WE WORK

                ;CREATE LOCAL STACK IN MEMORY THAT IS UNAFFECTED BY BRG

04EF 31F7F7             LXI     SP,SYSTP

                ;SET UP THE SERIAL PORT: INITIALIZE BRGSAV, DISABLE ONBOARD
                ;MEMORY & ENABLE OFFBOARD MEMORY

04F2 3E1F               MVI     A,DEBAUD        ;DEFAULT BAUD RATE
04F4 CD8FF5             CALL    STBAUD+ROF      ;ALSO ENABLES OFFBOARD RAM

                ;INSTALL INTERRUPT VECTORS IN OFFBOARD PAGE 0 MEMORY BEFORE
                ;ENABLING INTERRUPTS WITH OFFBOARD MEMORY SELECTED. THESE
                ;ALL JUST GO TO GOPOLY FOR NOW,SO THAT UNEXPECTED INTERRUPTS
                ;WILL RETURN TO THE POLY-88 MONITOR'S MAIN LOOP.

04F7 214000             LXI     H,8*8           ;EIGHT 8-BYTE VECTORS

04FA 1160F4     VLOOP1: LXI     D,PROTOV+ROF+8  ;PROTOTYPE VECTOR

04FD 2D         VLOOP2: DCR     L
04FE 1B                 DCX     D
04FF 1A                 LDAX    D               ;GET A VECTOR BYTE
0500 77                 MOV     M,A             ;PUT IT IN PLACE
0501 7D                 MOV     A,L             ;DONE WITH A VECTOR?
0502 E607               ANI     7

0504 C26FF4             JNZ     VLOOP2+ROF      ;N: KEEP GOING

0507 B5                 ORA     L               ;DONE WITH ALL VECTORS?
0508 C26CF4             JNZ     VLOOP1+ROF      ;N:KEEP GOING

                ;NOW PASTE IN SPECIFIC VECTORS THAT WE ACTUALLY PROCESS

050B 215204             LXI     H,UINT          ;UART INTERRUPT
050E 222300             SHLD    0023H           ;VI3

0511 2E4C               MVI     L,KINT AND 0FFH ;KEYBOARD INTERRUPT
0513 222B00             SHLD    002BH           ;VI2

0516 2E2F               MVI     L,RINT AND 0FFH ;RTC INTERRUPT
0518 223300             SHLD    0033H           ;VI2

051B 3EC9               MVI     A,RET           ;INITIALIZE TIMER ISR TO RET
051D 32FAF7             STA     PTMRISR

                ;FLUSH ANY GARBAGE FROM THE KEYBOARD AS LATE AS POSSIBLE

0520 DBF8               IN      KBD             ;ALSO CLEARS KBD INT
```

```
               ;INITIALIZE THE VIDEO DISPLAY, & PRINT BANNER (ENABLES INTS)

0522 CD35F7            CALL    ILPRNT+ROF      ;(ENABLES INTS TOO)
0525 0C                DB      FMFD            ;FORM FEED, CLEARS SCREEN
0526 506F6C6578        DB      'Polex 3.1','1'+80H

               ;TEST FOR MODULE ROM, CALL ITS INIT CODE IF ONE IS THERE
               ;SAVE MODULE'S COMMAND PROCESSOR ADDRESS, RETURNED IN HL

0530 CD2AF4            CALL    ONBRD+ROF       ;ENABLE ROMS
0533 3A0008            LDA     MODROM          ;IS THERE A MODULE ROM?
0536 3C                INR     A               ;FF MEANS NO ROM
0537 67                MOV     H,A             ;IN CASE NO MODULE
0538 C40008            CNZ     MODROM          ;INITIALIZE MODULE
053B 22F7F7            SHLD    MODGO           ;REMEMBER CMD PROC ADDR
053E CD6BF6            CALL    OFFBRD+ROF      ;DISABLE ROMS

               ;FALL INTO MAIN

               ;**************************************************
               ; COMMAND PROCESSOR MAIN ENTRY POINT
               ; GETS AND PROCESS COMMANDS
               ; SP GETS FIXED, IN CASE OF RE-ENTRY FROM AN ERROR
               ;**************************************************
0541 31F7F7    MAIN:   LXI     SP,SYSTP        ;FIX STACK POINTER

               ;PRINT THE PROMPT, AND GET A LINE OF KBD INPUT

0544 CD35F7            CALL    ILPRNT+ROF      ;PRINT CR, PROMPT
0547 0DBE              DB      CR,PROMPT+80H   ;WILL ALSO ENABLE INTS

0549 21B3F4            LXI     H,MAIN+ROF      ;CREATE RETURN ADDRESS
054C E5                PUSH    H               ;..ON THE STACK

054D CD7FF6            CALL    GETLIN+ROF      ;GET USER INPUT LINE
                                               ;DE=BEGINNING OF LINE
                                               ;Z SET IF NO CHR FOUND
                                               ;CURSOR AT END OF LINE

0550 C8                RZ                      ;NO COMMAND? NO ERROR.

               ;IF A MODULE WAS DETECTED DURING INIT, THEN GO THERE
               ;SO THE MODULE CAN CHECK ITS COMMAND LIST FIRST

0551 21CEF4            LXI     H,MODRET+ROF    ;CREATE MODULE
0554 E5                PUSH    H               ;..RETURN ADDRESS
0555 2AF7F7            LHLD    MODGO           ;GET THE MODULE'S ADDR
0558 7C                MOV     A,H             ;MODULE DETECTED?
0559 B7                ORA     A
055A C8                RZ                      ;N: RET TO MODRET

055B E9                PCHL                    ;Y: GO TO MODULE CMD PROC

055C C0        MODRET: RNZ                     ;DONE IF MOD TOOK THE CMD

               ;THE COMMAND WAS NOT RECOGNIZED BY THE MODULE. CHECK
               ;POLEX'S COMMAND LIST, AND EXECUTE THE COMMAND IF FOUND

055D EB                XCHG                    ;COMMAND TO HL
055E 114AF7            LXI     D,COMTAB+ROF-1  ;POINT TO COMMAND TABLE

               ;SEARCH THROUGH TABLE AT DE FOR A 2-CHR MATCH OF (HL)
```

```
0561 4E                 MOV     C,M             ;C=1ST COMMAND CHR
0562 23                 INX     H               ;M=2ND CMD CHR

0563 13         NXTCOM: INX     D               ;SKIP OVER ADDRESS OFFSET

0564 1A                 LDAX    D
0565 B7                 ORA     A               ;TEST FOR TABLE END
0566 CA43F7             JZ      ERROR+ROF       ;NOT IN TABLE

0569 A9                 XRA     C               ;TEST FIRST CHR
056A 47                 MOV     B,A             ;TEMP SAVE RESULT
056B 13                 INX     D               ;2ND TABLE CHR
056C 1A                 LDAX    D
056D AE                 XRA     M               ;TEST 2ND CHR

056E 13                 INX     D               ;POINT TO ADDRESS OFFSET
056F B0                 ORA     B               ;BOTH CHRS MATCH?
0570 C2D5F4             JNZ     NXTCOM+ROF      ;NO MATCH: KEEP LOOKING

                ;GOT A MATCH. COMPUTE ROUTINE ADDRESS AND PUT IT ON STACK
                ;A=B=0 HERE

0573 EB                 XCHG                    ;(HL)=OFFSET ADDR OF ROUTINE
                                                ;DE=POINTS TO 2ND INPUT CHR

0574 4E                 MOV     C,M             ;BC=ADDRESS OFFSET
0575 2110F5             LXI     H,CMDBAS+ROF    ;BASE OF COMMAND ROUTINES
0578 09                 DAD     B               ;HL=ADDRESS OF ROUTINE

0579 13                 INX     D               ;SKIP PAST 2-LETTER COMMAND
057A E3                 XTHL                    ;UNDO UPCOMING XTHL


                ;PUT ADDRESS OF ROUTINE ON THE STACK
                ;GET FOLLOWING PARAMETER (IF ANY) AND PUT IT IN HL.
                ;SET THE CARRY FLAG IF NO PARAMETER PRESENT.
                ;LEAVE DE POINTING TO THE 1ST CHR AFTER THE 1ST PARAMETER.
                ;'RETURN' TO THE COMMAND ROUTINE ON THE STACK, WITH CARRY SET
                ;IF NO HEX VALUE WAS FOUND. B=0.

                ;FALL INTO HLFHEX

                ;***SUBROUTINE************************
                ; SCAN PAST BLANKS AND GET A HEX VALUE
                ; ON EXIT:
                ;   TOP-OF-STACK=PREVIOUS HL VALUE
                ;   HL=VALUE
                ;   DE ADVANCED PAST CHR
                ;   CARRY SET IF NONE FOUND
                ;   CARRY CLEAR IF CHR FOUND
                ;*************************************
057B E3         HLFHEX: XTHL                    ;PUT HL ON STACK, GET RET ADDR
057C E5                 PUSH    H               ;REINSTALL RET ADDR

                ;FALL INTO FNDHEX

                ;***EXTERNAL SUBROUTINE***************
                ; SCAN PAST BLANKS AND GET A HEX VALUE
                ; ON EXIT:
                ;   HL=VALUE
                ;   DE ADVANCED PAST CHR
                ;   CARRY SET IF NONE FOUND
                ;   CARRY CLEAR IF CHR FOUND
                ;**************************************
057D CD2AF7     FNDHEX: CALL    SKIPB+ROF       ;Z SET IF NO CHR FOUND
0580 37                 STC

                ;FALL INTO INPHEX WITH Z AND CARRY SET IF NO CHR

                ;***SUBROUTINE*******************************************
                ; CONVERT ASCII HEX DIGITS FROM INPUT BUFFER TO BINARY
                ; STOP WHEN A SPACE OR THE CURSOR IS ENCOUNTERED.
                ; BAD HEX ABORTS THROUGH ERROR.
                ; ON ENTRY:
                ;    ASCII HEX AT (DE)
                ;    Z SET CAUSES IMMEDIATE RETURN WITH CARRY SET & HL=0
                ; ON EXIT:
                ;    HL=VALUE
                ;    DE ADVANCED PAST CHR
                ;    CARRY CLEAR, Z SET UNLESS Z & CARRY SET ON INPUT
                ;********************************************************
0581 210000     INPHEX: LXI     H,0             ;INITIAL VALUE

0584 C8                 RZ                      ;EXIT WITH CARRY SET FOR
                                                ;..FNDHEX IF NO CHR FOUND

0585 1A         IHEXLP: LDAX    D               ;GET CHARACTER
0586 E67F               ANI     7FH             ;VIDEO CHRS HAVE MSB SET

0588 FE7F               CPI     CURSOR AND 7FH  ;CURSOR MEANS END OF INPUT
058A C8                 RZ

058B FE20               CPI     ' '             ;VALUE SEPARATOR
058D C8                 RZ

058E 29                 DAD     H               ;MAKE ROOM FOR THE NEW ONE
058F 29                 DAD     H
0590 29                 DAD     H
0591 29                 DAD     H
0592 CD09F6             CALL    HEXCON+ROF      ;DO THE CONVERSION
0595 D243F7             JNC     ERROR+ROF       ;NOT VALID HEXIDECIMAL VALUE

0598 85                 ADD     L
0599 6F                 MOV     L,A             ;MOVE IT IN
059A 13                 INX     D               ;BUMP THE POINTER

059B C3F7F4             JMP     IHEXLP+ROF

                ;==========================
                ; BASE ADDRESS FOR COMMANDS
                ;==========================
059E =          CMDBAS  EQU     $

                ;***COMMAND ROUTINE******************************
                ; TE (TERMINAL MODE)
                ;
                ; SEND ALL KEYBOARD DATA TO THE SERIAL PORT, AND
                ; SEND ALL SERIAL PORT DATA TO THE SCREEN.
                ; 'ALT-MODE' KEY ON THE KEYBOARD TO EXIT
                ;
                ; ON ENTRY:
                ;   E<>0 (FORCES GETSER TO ECHO)
                ;************************************************
059E CDEEF6     TERMNL: CALL    CHKKBD+ROF      ;ANY KBD DATA?
                                                ;ALSO CHECKS FOR ABORT
```

```
05A1 4F                 MOV     C,A             ;CHR IN C FOR SODATA
05A2 C4D6F6             CNZ     SODATA+ROF

05A5 CDADF6             CALL    SISTAT+ROF      ;ANY SERIAL DATA?
05A8 C454F6             CNZ     GETSER+ROF      ;GET & ECHO SERIAL DATA

05AB C310F5             JMP     TERMNL+ROF

                ;***COMMAND ROUTINE*******************************
                ; DU <ADR1> <ADR2> (DUMP MEMORY)
                ;
                ; PRINT MEMORY CONTENTS FROM <ADR1> TO <ADR2> ON
                ; THE CONSOLE IN HEX.  IF ONLY <ADR1> IS SPECIFIED,
                ; THEN PRINT JUST THE CONTENTS OF THAT ADDRESS.
                ; ON ENTRY:
                ;    HL=ADDR1
                ;    (DE) POINTS TO <ADR2>, IF ANY
                ;*************************************************
05AE CDEDF4     DUMP:   CALL    HLFHEX+ROF      ;PUSH 1ST ADDRESS,
                                                ;GET HL=SECOND ADDRESS OR 0

05B1 D1                 POP     D               ;RECOVER START ADDRESS
05B2 EB                 XCHG                    ;HL HAS START, DE HAS END

                ;PRINT THE ADDRESS AT THE BEGINNING OF EACH LINE

05B3 CD73F6     DLINE:  CALL    PHLADR+ROF      ;PRINT HL AS AN ADDRESS

                ;PRINT 16 BYTES OF HEX DATA (SEPARATED BY SPACES) ON EACH LINE

05B6 CD35F7     DLOOP:  CALL    ILPRNT+ROF      ;PRINT A SPACE
05B9 A0                 DB      ' '+80H

05BA 7E                 MOV     A,M             ;GET THE CHR
05BB CD2BF6             CALL    PAHEX+ROF       ;SEND IT OUT WITH A BLANK

05BE 7D                 MOV     A,L             ;COMPARE DE AND HL
05BF 93                 SUB     E
05C0 7C                 MOV     A,H
05C1 9A                 SBB     D
05C2 D0                 RNC                     ;ALL DONE

05C3 23                 INX     H

05C4 7D                 MOV     A,L
05C5 E60F               ANI     0FH             ;NEW LINE EVERY XXX0 HEX

05C7 C228F5             JNZ     DLOOP+ROF       ;NOT ZERO IF MORE FOR THIS LINE

                ;GIVE USER A CHANCE TO PAUSE OR QUIT AT THE END OF EACH LINE

05CA CDEEF6             CALL    CHKKBD+ROF      ;SEE IF A CHAR WAITING
                                                ;..AND ABORT IF IT'S ALT-MODE
05CD FE20               CPI     PAUKEY          ;IS IT A PAUSE?
05CF CC0DF7             CZ      GETKBD+ROF      ;Y: GO WAIT FOR ANOTHER CHR
                                                ;..OR USER ABORT

05D2 C325F5             JMP     DLINE+ROF       ;NEXT LINE

                ;***COMMAND ROUTINE***************************************
                ; EN <ADR> (ENTER DATA INTO MEMORY)
                ;
                ; GET HEX VALUES FROM THE KEYBOARD AND ENTER THEM
```

```
               ; SEQUENTIALLY INTO MEMORY, STARTING AT <ADR>. A BLANK
               ; LINE ENDS THE ROUTINE AND RETURNS CONTROL TO THE
               ; COMMAND MODE. VALUES MAY BE SEPARATED BY SPACES OR CR'S.
               ; PRINT THE CURRENT ADDRESS AT THE BEGINNING OF EACH LINE.
               ; ON ENTRY:
               ;  HL = ADDRESS
               ;  CARRY SET IF NONE ENTERED
               ;***********************************************************
05D5 DA43F7    ENTER:  JC      ERROR+ROF       ;MUST HAVE ADDRESS

               ;PRINT THE ADDRESS AT THE BEGINNING OF EACH LINE

05D8 CD73F6    ENLINE: CALL    PHLADR+ROF      ;PRINT HL AS AN ADDRESS

               ;GET A LINE OF USER INPUT

05DB E5                PUSH    H               ;SAVE ADDRESS
05DC CD7FF6            CALL    GETLIN+ROF      ;INIT AND PROCESS A LINE
05DF E1                POP     H               ;HL=BEGINNING OF LINE,
                                               ;..PAST ANY LEADING BLANKS
05E0 C8                RZ                      ;Z=BLANK LINE TERMINATES

               ;GET HEX DATA FROM THE USER INPUT LINE AND WRITE IT TO MEMORY

05E1 E5        ENLOOP: PUSH    H               ;SAVE ADDRESS
05E2 CDF3F4            CALL    INPHEX+ROF      ;GET/CONVERT VALUE (Z CLEAR HERE)

05E5 7D                MOV     A,L             ;GET LOW BYTE AS CONVERTED
05E6 E1                POP     H               ;RECOVER MEMORY ADDRESS

05E7 77                MOV     M,A             ;PUT IN THE VALUE
05E8 23                INX     H               ;NEXT ADDRESS

05E9 CD2AF7            CALL    SKIPB+ROF       ;SCAN TO NEXT INPUT VALUE
05EC C253F5            JNZ     ENLOOP+ROF      ;NOT END OF LINE: CONTINUE

05EF C34AF5            JMP     ENLINE+ROF      ;END OF LINE: START NEW LINE

               ;***COMMAND ROUTINE*******************
               ; EX <ADR> (EXECUTE)
               ;
               ; JUMP TO <ADR>
               ; ON ENTRY:
               ;  HL = ADDRESS
               ;  CARRY SET IF NONE ENTERED
               ;*************************************
05F2 E9        EXEC:   PCHL

               ;***COMMAND ROUTINE**********************************
               ; FI <ADR1> <ADR2> <VAL> (FILL MEMORY)
               ;
               ; FILL MEMORY WITH <VAL> FROM <ADR1> THROUGH <ADR2>.
               ; IF <VAL> IS NOT PROVIDED, THEN FILL THE SPECIFIED
               ; RANGE WITH 00. IF <ADR2> IS NOT PROVIDED, THEN FILL
               ; JUST ONE BYTE AT <ADR1> WITH 00.
               ; ON ENTRY:
               ;  HL=<ADR1>
               ;  CARRY SET IF NONE ENTERED
               ;  (DE) POINTS TO <ADR2>, <VAL> FOLLOWS, IF ANY
               ;****************************************************
05F3 DA43F7    FILMEM: JC      ERROR+ROF       ;MUST PROVIDE A START, SO THAT
                                               ;DEFAULT DOESN'T KILL INT VECTOR
```

```
05F6 CDEDF4             CALL    HLFHEX+ROF      ;PUSH 1ST ADDRESS,
                                                ;GET HL=SECOND ADDRESS OR 0
05F9 CDEDF4             CALL    HLFHEX+ROF      ;PUSH 2ND ADDRESS,
                                                ;GET HL=SECOND ADDRESS OR 0
05FC 4D                 MOV     C,L             ;FILL DATA IN C

05FD D1                 POP     D               ;DE HAS END ADDRESS
05FE E1                 POP     H               ;HL HAS START ADDRESS

05FF 71         FMLOOP: MOV     M,C

0600 7D                 MOV     A,L             ;COMPARE DE AND HL
0601 93                 SUB     E
0602 7C                 MOV     A,H
0603 9A                 SBB     D
0604 D0                 RNC                     ;HL>DE: ALL DONE

0605 23                 INX     H
0606 C371F5             JMP     FMLOOP+ROF

                ;***COMMAND ROUTINE*************
                ; PO (RETURN TO POLY-88 MONITOR)
                ;*******************************
0609 AF         GOPOLY: XRA     A               ;RESET BAUD RATE,
060A D304               OUT     BRG             ;..ENABLE ONBOARD MEMORY

060C 210D01             LXI     H,KSR           ;RESTORE POLY-88 INT ROUTINES
060F 22180C             SHLD    SRA5            ;..IN ONBOARD MEMORY
0612 215400             LXI     H,USRTSR
0615 22160C             SHLD    SRA4

                ;REVERT TO POLY-88 STACK IN ONBOARD RAM AND RETURN

0618 31FE0F             LXI     SP,POLYSP       ;ORIGINAL STACK POINTER
061B C9                 RET                     ;RETURN TO POLY-88 MONITOR

                ;***COMMAND ROUTINE**********************
                ; SB <BAUD> (SET BAUD RATE)
                ;
                ; ON ENTRY: L=<BAUD> VALUE FOR BRG:
                ;    <BAUD> BITS 3:0 SELECT THE BAUD RATE
                ;    <BAUD> BIT 4 SELECTS THE PORT:
                ;       0 SELECTS THE CASSETTE TAPE PORT
                ;       1 SELECTS THE PRINTER PORT
                ;    BITS 7:5 ARE IGNORED
                ; ON EXIT:
                ;    BRG AND BRGSAV HAVE BEEN UPDATED
                ;    ONBOARD MEMORY IS DISABLED
                ;    UART RX IS ENABLED
                ;    UART TX IS DISABLED
                ;    TRASHES A
                ; STACK USAGE:
                ;   0 BYTES
                ;*****************************************
061C 7D         SBCMD:  MOV     A,L             ;BAUD VALUE TO A

                ;FALL INTO STBAUD

                ;***EXTERNAL SUBROUTINE*****************
                ; SET BAUD RATE
                ;
                ; ON ENTRY: A= VALUE FOR BRG:
                ;    BITS 3:0 SELECT THE BAUD RATE
```

```
               ;    BIT 4 SELECTS THE PORT:
               ;     0 SELECTS THE CASSETTE TAPE PORT
               ;     1 SELECTS THE PRINTER PORT
               ;    BITS 7:5 ARE IGNORED
               ; ON EXIT:
               ;    BRG AND BRGSAV HAVE BEEN UPDATED
               ;    ONBOARD MEMORY IS DISABLED
               ;    UART RX IS ENABLED
               ;    UART TX IS DISABLED
               ;    TRASHES A
               ; STACK USAGE:
               ;   0 BYTES
               ;*****************************************

               ; SET THE BAUD RATE DIVISOR, ENABLE OFFBOARD MEMORY

               STBAUD:
061D E61F              ANI     1FH             ;BIT 7 MUST BE LOW
061F F660              ORI     BRGOMD          ;SELECT OFFBOARD MEM
0621 32F9F7            STA     BRGSAV          ;..AND SET BAUD RATE
0624 D304              OUT     BRG

               ;SEND COMPLETE (4-BYTE) ASYNC RESET SEQUENCE TO 8251 UART

0626 3EAA              MVI     A,0AAH          ;<1> FAKE SYNCH CHR
0628 D301              OUT     UARTS           ;ALSO DIABLES TX, RX INTS

062A 3E40              MVI     A,UCMDRE        ;<2> RESET TO MODE
062C D301              OUT     UARTS

062E 3E4E              MVI     A,UMOD16        ;<3> CLOCK DIVISOR ETC.
0630 D301              OUT     UARTS

               ;FALL INTO UTXDIS FOR THE FINAL RESET SEQUENCE BYTE

               ;***SUBROUTINE*****************************
               ;DISABLE UART TRANSMITTER, ENABLE RECEIVER
               ; TRASHES A
               ;*****************************************
0632 3EFF      UTXDIS: MVI     A,0FFH          ;REMEMBER DISABLED STATE
0634 3264F7            STA     URTDIS+ROF
0637 3E26              MVI     A,UCRXEN        ;<4> ENABLE RECEIVER
0639 D301              OUT     UARTS           ;..DISABLE TRANSMITTER
063B C9                RET

               ;***COMMAND ROUTINE*****************************************
               ; HL [0/1] (INTEL HEX LOAD)
               ;
               ; LOAD AN INTEL HEX FILE FROM THE POLY-88 SERIAL PORT INTO
               ; MEMORY AT THE ADDRESSES SPECIFIED IN THE HEX FILE. DONE
               ; WHEN ANY 0-BYTE RECORD ENCOUNTERED, OR IF THE ALT-MODE
               ; OR ESCAPE KEY IS TYPED. PARAMETER>0 MEANS ECHO RECEIVED
               ; CHRS. PARAMETER=0 OR NO PARAMETER MEANS NO ECHO;
               ; PRINT A PACIFIER DOT PER RECORD INSTEAD.
               ;
               ; ON ENTRY:
               ;    B=0
               ;   HL=0 AND CARRY SET IF NO PARAMETERS
               ;   CARRY SET IF NO PARAMETER WAS INCLUDED
               ;
               ; REGISTER USAGE DURING HEX LOAD:
               ;   B: SCRATCH
               ;   C: RECORD BYTE COUNT
```

```
                ;   D: RECORD CHECKSUM
                ;   E: ECHO STATE (0 MEANS PACIFIER DOTS, <>0 MEANS ECHO)
                ;   HL: MEMORY ADDRESS (AND RECORD COUNT, ON STACK MOSTLY)
                ;**********************************************************
063C 5D         HEXLOD: MOV     E,L             ;PARAMETER LOW BYTE
                                                ;0 OR NO PARAMETER MEANS
                                                ;NO ECHO

063D CD35F7             CALL    ILPRNT+ROF      ;BE PRETTY: NEW LINE
0640 8D                 DB      CR+80H

                ;INITIALIZE

0641 60                 MOV     H,B             ;B=0: INIT RECORD COUNT
0642 68                 MOV     L,B
0643 50                 MOV     D,B             ;INIT CKSUM

                ;FLUSH SERIAL PORT RECEIVE QUEUE IN CASE OF OLD JUNK

0644 CDADF6     SFLUSH: CALL    SISTAT+ROF      ;ANYTHING IN THE QUEUE?
0647 C4BCF6             CNZ     GSDATA+ROF      ;Y:GO GET IT & FORGET IT
064A C2B6F5             JNZ     SFLUSH+ROF      ;Y:SEE IF THERE'S MORE

                ;EAT ALL CHRS UNTIL WE GET RECORD-START COLON

064D CD54F6     GETCOL: CALL    GETSER+ROF
0650 FE3A               CPI     ':'
0652 C2BFF5             JNZ     GETCOL+ROF

0655 3E2E               MVI     A,PCFIER        ;PACIFIER CHARACTER
0657 1D                 DCR     E               ;PRINT PACIFIER IF ECHO DISABLE
0658 1C                 INR     E               ;E=0 MEANS ECHO DISABLED
0659 CC62F6             CZ      PRINTA+ROF

                ;BUMP RECORD COUNT AND PARK IT ON THE STACK (D=0 HERE)

065C 23                 INX     H               ;16-BIT INCREMENT
065D E5                 PUSH    H               ;PUT RECORD COUNT ONTO STACK

                ;GET 4-BYTE RECORD HEADER: (CHECKSUM=D=0 HERE)
                ; C GETS 1ST BYTE = DATA BYTE COUNT
                ; H GETS 2ND BYTE = ADDRESS HIGH BYTE
                ; L GETS 3RD BYTE = ADDRESS LOW BYTE
                ; 4TH BYTE = RECORD TYPE GETS CHUCKED
                ; D = CHECKSUM OF THE ABOVE 4 BYTES

                ;SHIFT IN THE FOUR HEADER BYTES: C <- H <- L <- B
065E 3E04               MVI     A,4             ;LOOP COUNTER

0660 4C         HEDRLP: MOV     C,H             ;C=BYTE 1: BYTE COUNT
0661 65                 MOV     H,L             ;H=BYTE 2: ADDRESS MSB
0662 68                 MOV     L,B             ;L=BYTE 3: ADDRESS LSB
0663 F5                 PUSH    PSW
0664 CDF6F5             CALL    GETBYT+ROF      ;GET B=HEADER BYTE, DO CKSUM
0667 F1                 POP     PSW
0668 3D                 DCR     A               ;NEXT HEADER BYTE
0669 C2D2F5             JNZ     HEDRLP+ROF

                ;REMEMBER BYTE COUNT ON STACK, BUMP C TO INCLUDE CKSUM BYTE.

066C C5                 PUSH    B               ;REMEMBER C=BYTE COUNT
                                                ;(B=RECORD TYPE...)
```

```
066D C2                 DB      JNZ             ;JNZ OPCODE SKIPS 2 BYTEs


                ;LOOP TO GET C DATA BYTES INTO MEMORY AT HL. C HAS BEEN BUMPED
                ;SO THAT THIS LOOP WILL ALSO READ THE CHECKSUM BYTE. EXIT WITH
                ;THE RECORD CHECKSUM IN A.

066E 70         DATALP: MOV     M,B
066F 23                 INX     H               ;NEXT MEMORY LOC

0670 CDF6F5             CALL    GETBYT+ROF      ;GET B=DATA BYTE, A=D=CKSUM

0673 0D                 DCR     C               ;NEXT DATA (OR CKSUM) BYTE
0674 F2E0F5             JP      DATALP+ROF

                ;VERIFY THE RESULT, AND CONTINUE LOADING UNLESS AN EOF RECORD
                ;WAS RECEIVED. (AN EOF RECORD IS ONE WITH 0 DATA BYTES.)

0677 C1                 POP     B               ;RECOVER C=BYTE COUNT
                                                ;(AND B=RECORD TYPE...)
0678 E1                 POP     H               ;RECOVER RECORD COUNT

0679 B7                 ORA     A               ;VALIDATE CKSUM (IN A AND D)
067A C216F6             JNZ     HLERR+ROF       ;NZ MEANS CKSUM ERROR

                ;       DCR     B               ;TYPE 1 RECORD?
                ;       JZ      PRECS+ROF

067D B1                 ORA     C               ;ZERO BYTE RECORD?
067E C2BFF5             JNZ     GETCOL+ROF      ;NO: GO GET ANOTHER RECORD

0681 C31DF6             JMP     PRECS+ROF       ;YES: PRINT REC COUNT AND EXIT

                ;***SUBROUTINE*****************************************
                ; GET 2 HEX DIGITS FROM THE SERIAL PORT, COMBINE THEM
                ; INTO A BYTE, AND ADD RESULT TO THE CHECKSUM IN D
                ; ON ENTRY:
                ;   D = CHCKSUM SO FAR
                ;   E = ECHO FLAG
                ; ON EXIT:
                ;   B=BYTE OF DATA
                ;   A=D=NEW CHECKSUM VALUE
                ;   Z FLAG SET IF CHECKSUM IS 0
                ;   ALL OTHER REGISTERS PRESERVED, UNLESS ERROR ABORT
                ;*****************************************************
0684 CD06F6     GETBYT: CALL    SERHEX+ROF      ;GET HIGH NIBBLE
0687 87                 ADD     A               ;SHIFT HIGH NIBBLE IN PLACE
0688 87                 ADD     A
0689 87                 ADD     A
068A 87                 ADD     A
068B 47                 MOV     B,A
068C CD06F6             CALL    SERHEX+ROF      ;GET LOW NIBBLE

068F B0                 ORA     B               ;COMBINE NIBBLES
0690 47                 MOV     B,A             ;SAVE RESULT FOR RETURN
0691 82                 ADD     D               ;COMPUTE CHECKSUM
0692 57                 MOV     D,A             ;RET WITH CKSUM IN A & D

0693 C9                 RET

                ;===LOCAL SUBROUTINE=========================
                ; GET A HEX DIGIT FROM THE SERIAL PORT,
                ; CONVERT IT TO BINARY, AND RETURN IT IN A<3:0>
                ;==============================================
0694 CD54F6     SERHEX: CALL    GETSER+ROF

                ;FALL INTO HEXCON

                ;***SUBROUTINE*************************
                ;  CONVERT ASCII HEX DIGIT TO BINARY
                ; ON ENTRY:
                ;   A=CHR TO CONVERT
                ; ON EXIT:
                ;   A=BINARY, IF OK
                ;   CARRY SET IF OK, CLEAR IF BOGUS CHR
                ;**************************************
0697 D630       HEXCON: SUI     '0'             ;REMOVE ASCII BIAS
0699 FE0A               CPI     10
069B D8                 RC                      ;IF 0-9 THEN WE'RE DONE

069C D611               SUI     9+('A'-'9')     ;SHOULD BE 0-5 NOW
069E FE06               CPI     6               ;GAP CHR OR TOO HIGH?
06A0 D0                 RNC                     ;ERROR IF SO

06A1 D6F6               SUI     0F6H            ;ADD 0AH, SET CARRY
06A3 C9                 RET                     ;RET WITH CARRY SET

                ;***SUBROUTINE*********************
                ; PRINT HEX LOAD ERROR MESSAGE
                ; THEN PRINT RECORD COUNT AND ABORT
                ; ON ENTRY:
                ;    B = 1 CHARACTER ERROR MESSAGE
                ;    HL = RECORD COUNT
                ; TRASHES A
                ;**********************************
06A4 CD35F7     HLERR:  CALL    ILPRNT+ROF
06A7 0D4572F2           DB      CR,'Er','r'+80H

                ;FALL INTO PRECS TO PRINT RECORD COUNT

                ;***SUBROUTINE******************
                ; PRINT THE RECORD-COUNT MESSAGE
                ; ON ENTRY:
                ;    HL=RECORD COUNT
                ; TRASHES A
                ;*******************************
06AB CD35F7     PRECS:  CALL    ILPRNT+ROF
06AE 0D52656373         DB      CR,'Recs',':'+80H

                ;FALL INTO PHLHEX TO PRINT HL IN HEX & RETURN TO MAIN

                ;***SUBROUTINE**********************
                ; PRINT HL ON CONSOLE AS 4 HEX DIGITS
                ; TRASHES A
                ;***********************************
06B4 7C         PHLHEX: MOV     A,H             ;H FIRST
06B5 CD2BF6             CALL    PAHEX+ROF
06B8 7D                 MOV     A,L             ;THEN L

                ;FALL INTO PAHEX

                ;***EXTERNAL SUBROUTINE*************
                ; PRINT A ON CONSOLE AS 2 HEX DIGITS
                ; TRASHES A
                ;**********************************
06B9 F5         PAHEX:  PUSH    PSW             ;GET THE CHARACTER
```

```
06BA 0F                 RRC                     ;MOVE THE HIGH FOUR DOWN
06BB 0F                 RRC
06BC 0F                 RRC
06BD 0F                 RRC
06BE CD34F6             CALL    PNIBLE+ROF      ;PUT THEM OUT
06C1 F1                 POP     PSW             ;THIS TIME THE LOW FOUR

                ;FALL INTO PNIBLE

                ;===LOCAL SUBROUTINE=========
                ; PRINT LOW NIBBLE OF A IN HEX
                ; TRASHES A
                ;============================
06C2 E60F       PNIBLE: ANI     0FH             ;FOUR ON THE FLOOR
06C4 C630               ADI     '0'             ;WE WORK WITH ASCII HERE
06C6 FE3A               CPI     '9'+1           ;0-9?
06C8 DA62F6             JC      PRINTA+ROF      ;YUP: PRINT & RETURN

06CB C607               ADI     'A'-'9'-1       ;MAKE IT A LETTER
06CD C362F6             JMP     PRINTA+ROF      ;PRINT IT & RETURN

                ;***CP/M EXTERNAL SUBROUTINE*********************************
                ; PRINT C ON THE CONSOLE SCREEN
                ; THIS CREATES A LOCAL STACK BECAUSE A CALLING PROGRAM'S STACK
                ; MAY BE IN THE 1ST 4K OF RAM, WHICH WILL DISAPPEAR WHEN WE
                ; GO ONBRD. THE LOCAL STACK MUST HAVE ROOM FOR 28 BYTES TO
                ; HANDLE DSPLY'S NEEDS, AS WELL AS SEVERAL STACKED ONBOARD
                ; INTERRUPTS.
                ;
                ; THE BS CHR IS TRANSLATED TO A DEL HERE.
                ;
                ; ON ENTRY:
                ;   THE CHARACTER FOR OUTPUT IS IN C
                ; ON EXIT:
                ;   C=CHR
                ;   ALL OTHER REGS PRESERVED
                ;   INTERRUPTS ENABLED
                ; STACK USAGE:
                ;   2 BYTES
                ;*************************************************************
06D0 E5         PRINTC: PUSH    H               ;SAVE INCOMING HL

06D1 210000             LXI     H,0
06D4 39                 DAD     SP              ;HL= INCOMING SP

06D5 31C2F7             LXI     SP,VDSTAK       ;CREATE LOCAL STACK

06D8 79                 MOV     A,C             ;A=CHR TO PRINT
06D9 CD62F6             CALL    PRINTA+ROF      ;PRINT ON SCREEN
                                                ;ENABLES INTS ON EXIT

06DC F9                 SPHL                    ;RESTORE INCOMING STACK
06DD E1                 POP     H               ;RESTORE INCOMING HL
06DE C9                 RET

                ;***SUBROUTINE*******************************
                ; SERIAL PORT INPUT FOR HEX LOADER
                ; STRIPS PARITY, ECHOS TO THE CONSOLE.
                ; CHECKS FOR ABORT FROM THE KEYBOARD
                ; oN eBTRY:
                ;   E = 0 IF ECHO IS DISABLED
                ; ON EXIT:
                ;   CHR IN A, WITH PARITY STRIPPED
                ;   INTERRUPTS ARE ENANLED
                ;-->ENTRY IS AT GETSER<--
                ;*********************************************
06DF CDEEF6     GSWAIT: CALL    CHKKBD+ROF      ;USER ABORT?

06E2 CDADF6     GETSER: CALL    SISTAT+ROF      ;ANY SERIAL CHRS?
06E5 CA51F6             JZ      GSWAIT+ROF      ;SISTAT ENABLES INTS

06E8 CDBCF6             CALL    GSDATA+ROF      ;GET SERIAL CHR
06EB E67F               ANI     7FH             ;STRIP PARITY

06ED 1D                 DCR     E               ;ECHO ENABLED?
06EE 1C                 INR     E
06EF C8                 RZ                      ;N: DONE

                ;FALL INTO PRINTA TO ECHO THE CHR IN A

                ;***SUBROUTINE*****************************************
                ; SEND A TO SCREEN (TRANSLATE BS TO DEL IF NECESSARY)
                ; ON ENTRY:
                ;   THE CHARACTER FOR OUTPUT IS IN A
                ; ON EXIT:
                ;   ALL REGS PRESERVED
                ;*****************************************************
06F0 CD07F7     PRINTA: CALL    BS2DEL+ROF      ;TRANSLATE BACKSPACE TO DEL
06F3 CD2AF4             CALL    ONBRD+ROF       ;ENABLE ONBOARD MEMORY
06F6 CD7F00             CALL    DSPLY           ;USE POLY88 VIDEO DRIVER
                                                ;ALSO ENABLES INTS

                ;FALL INTO OFFBRD

                ;***EXTERNAL SUBROUTINE*******************************
                ; SELECT OFFBOARD MEMORY
                ; NOTE: YOU CAN'T CALL THIS SUBROUTINE IF THE STACK IS
                ; (OR MIGHT BE) IN THE 1ST 4K OF MEMORY, AS THE STACK
                ; WILL DISAPPEAR WHEN SWITCHING OFFBOARD.
                ;*****************************************************
06F9 F5         OFFBRD: PUSH    PSW
06FA 3AF9F7             LDA     BRGSAV          ;OFFBOARD MEMORY
06FD D304               OUT     BRG
06FF F1                 POP     PSW
0700 C9                 RET

                ;***SUBROUTINE*****************************************
                ; PRINT HL AS AN ADDRESS (CR, ADDRESS, COLON)
                ; TRASHES A
                ;*****************************************************
0701 CD35F7     PHLADR: CALL    ILPRNT+ROF      ;CR BEGINS LINE
0704 8D                 DB      CR+80H

0705 CD26F6             CALL    PHLHEX+ROF      ;HL=ADDRESS

0708 3E3A               MVI     A,':'
070A C362F6             JMP     PRINTA+ROF      ;RETURN FROM THERE

                ;***SUBROUTINE*********************************************
                ; READ A COMMAND LINE FROM THE KEYBOARD, ECHOING AND LETTING
                ; THE POLY-88 SUBROUTINE SAVE IT IN VIDEO MEMORY
                ;
                ; CR INPUT ENDS THE SEQUENCE. THE CR IS NOT SAVED IN VIDEO
                ; MEMORY. INSTEAD, THE LINE IS TERMINATED BY THE CURSOR THAT
                ; THE POLY-88 VIDEO ROUTINE LEAVES AT THE END OF THE LINE.
                ;
                ; ON EXIT:
                ;   COMPLETE COMMAND LINE IS IN VIDEO MEMORY, TERMINATED
                ;   WITH THE CURSOR CHARACTER (FFH)
                ;   DE=ADDRESS OF THE FIRST NON-BLANK CHR ON THE LINE
                ;   A = FIRST NON-BLANK VALUE FOUND
                ;   Z SET IF NOTHING BUT BLANKS FOUND
                ;   INTERRUPTS ENABLED
                ;   BC,HL TRASHED
                ;***********************************************************
070D CD2AF4     GETLIN: CALL    ONBRD+ROF          ;ONBRD SO WE CAN ACCESS POS
0710 2A0E0C             LHLD    POS                ;GET POLY-88 CURSOR POS
0713 CD6BF6             CALL    OFFBRD+ROF

0716 54                 MOV     D,H                ;SAVE INIT POS FOR RETURN
0717 5D                 MOV     E,L                ;L STILL HAS LINE POS

                ;LOOP TO GET INPUT LINE

                ;WAIT FOR A KEY TO BE PRESSED, ABORT IF USER WANTS TO

0718 CD0DF7     GCLI1:  CALL    GETKBD+ROF         ;GET NEXT KBD CHR INTO A
                                                   ;..ENABLES INTERRUPTS TOO

                ;IF CHR IS THE TERMINATING CR, THEN RETURN WITHOUT ECHOING

071B FE0D               CPI     CR                 ;TERMINATING CR?
071D CA2AF7             JZ      SKIPB+ROF          ;YES: GO SKIP LEADING BLANKS
                                                   ;..& SET Z IF NO INPUT

                ;PUT LOOP BEGINNING ONTO STACK

0720 018AF6             LXI     B,GCLI1+ROF
0723 C5                 PUSH    B

                ;CHUCK ANY CONTROL CHARACTERS, AS SOME OF THEM WILL MESS UP
                ;OUR CURSOR POSITION, AND WE DON'T USE ANY OF THEM ANYWAY

0724 FE20               CPI     ' '
0726 D8                 RC                         ;RET TO GCLI1

                ;IF DEL, DELETE IF POSSIBLE

0727 FE7F               CPI     DEL                ;DEL?

0729 67                 MOV     H,A                ;TEMP SAVE CHR
072A 7D                 MOV     A,L                ;GET CURSOR POS LOW BYTE

072B C2A4F6             JNZ     GCLI2+ROF

072E BB                 CMP     E                  ;AT BEGINNING OF LINE?
072F C8                 RZ                         ;Y: CAN'T DELETE
                                                   ;...SO RET TO GCLI1

0730 3D                 DCR     A                  ;DELETE
0731 3D                 DCR     A                  ;ACCOUNT FOR UPCOMING INR A

                ;ECHO, AND STORE BYTE IN VIDEO MEMORY, IF ROOM

0732 3C         GCLI2:  INR     A                  ;END OF LINE (X100 0000B)?
0733 E63F               ANI     VDROW-1            ;VDROW IS A POWER OF 2
0735 C8                 RZ                         ;FULL LINE: CHUCK CHR
                                                   ;..AND RET TO GCLI1
```

```
0736 6F                 MOV     L,A             ;SAVE LINE POSITION
0737 7C                 MOV     A,H             ;RECOVER CHR
0738 C362F6             JMP     PRINTA+ROF      ;ECHO, ENABLE INTS
                                                ;RET TO GCLI1

                ;***CP/M EXTERNAL SUBROUTINE*************
                ; GET SERIAL PORT RX STATUS
                ; ON EXIT:
                ;   A=0 & Z SET IF NO DATA
                ;   A=FF  & Z CLEAR IF DATA
                ;   INTERRUPTS ENABLED
                ; STACK USAGE:
                ;   0 BYTES
                ;****************************************
073B 3A66F7     SISTAT: LDA     SRQCNT+ROF

                ;FALL INTO DOSTAT

                ;***SUBROUTINE RETURN********************
                ; CP/M-STYLE STATUS RETURN
                ; ON EXIT:
                ;   A=00 & Z SET MEANS NO CHR AVAILABLE
                ;   A= FFH & Z CLEARED MEANS CHR AVAILABLE
                ;   INTERRUPTS ENABLED
                ;****************************************
073E FB         DOSTAT: EI                      ;IN CASE THEY WERE DISABLED
073F B7                 ORA     A
0740 C8                 RZ
0741 3EFF               MVI     A,0FFH
0743 C9                 RET

                ;***CP/M EXTERNAL SUBROUTINE*************
                ; WAIT FOR AND GET SERIAL RX PORT DATA
                ; ON EXIT:
                ;   A=BYTE FROM QUEUE
                ;   Z CLEARED
                ;   INTERRUPTS ENABLED
                ; STACK USAGE WHILE INT'S ENABLED:
                ;   2 BYTES
                ;****************************************

                ;SPIN UNTIL THE QUEUE IS NOT EMPTY

0744 CDADF6     SIDATA: CALL    SISTAT+ROF
0747 CAB6F6             JZ      SIDATA+ROF

                ;FALL INTO GSDATA

                ;***SUBROUTINE***************************
                ; GET SERIAL RX PORT DATA FROM QUEUE
                ; ON ENTRY:
                ;   A CHR IS KNOWN TO BE IN THE QUEUE
                ; ON EXIT:
                ;   A=BYTE FROM QUEUE
                ;   Z CLEARED UNLESS WE GOT THE LAST BYTE
                ; STACK USAGE WHILE INT'S ENABLED:
                ;   2 BYTES
                ;****************************************
074A E5         GSDATA: PUSH    H
074B 2A69F7             LHLD    SRQOPT+ROF      ;RX Q OUTPUT POINTER

                ;CIRCULAR-INC SRQOPT. ASSUME SUPER-NICE QUEUE ALIGNMENT
                ;AND DON'T MESS UP HL YET
```

```
074E 7D                 MOV     A,L
074F 3C                 INR     A
0750 E6DF               ANI     SRQSIZ XOR 0FFH
0752 3269F7             STA     SRQOPT+ROF      ;H NEVER CHANGES

                ;GET QUEUE DATA BEFORE BUMPING COUNT, SO INTS
                ;WONT MESS IT UP.

0755 7E                 MOV     A,M             ;GET Q DATA FOR RET

                ;BUMP QUEUE BYTE COUNT

0756 2166F7             LXI     H,SRQCNT+ROF
0759 35                 DCR     M               ;ATOMIC READ THEN WRITE

075A E1                 POP     H
075B C9                 RET

                ;***CP/M EXTERNAL SUBROUTINE*************************
                ; GET SERIAL TX PORT QUEUE STATUS
                ; ON EXIT:
                ;   Z CLEAR AND A=FFH IF THE QUEUE CAN ACCEPT A CHR
                ;   Z SET AND A=0 OF THE SERIAL TX QUEUE IS FULL
                ;   INTERRUPTS ENABLED
                ; STACK USAGE:
                ;   0 BYTES
                ;****************************************************
075C 3A65F7     SOSTAT: LDA     STQCNT+ROF
075F D604               SUI     STQSIZ          ;QUEUE EMPTY?

0761 C3B0F6             JMP     DOSTAT+ROF      ;CP/M-STYLE STAT RETURN

                ;***CP/M EXTERNAL SUBROUTINE***********
                ; SEND C TO SERIAL PORT QUEUE
                ; PSW TRASHED, ALL OTHER REGS PRESERVED
                ; STACK USAGE WHILE INTS ENABLED:
                ;   2 BYTES
                ;**************************************
0764 E5         SODATA: PUSH    H


                ;SPIN UNTIL TX QUEUE IS NOT FULL

0765 CDCEF6     SOWAIT: CALL    SOSTAT+ROF
0768 CAD7F6             JZ      SOWAIT+ROF

                ;ENQUEUE THE CHR

076B 2A6BF7             LHLD    STQIPT+ROF      ;TX QUEUE IN-POINTER
076E 71                 MOV     M,C             ;PUT CHR IN QUEUE

                ;CIRCULAR-INC STQIPT. ASSUME SUPER-NICE QUEUE ALIGNMENT

076F 2C                 INR     L
0770 3EFB               MVI     A,STQSIZ XOR 0FFH
0772 A5                 ANA     L
0773 326BF7             STA     STQIPT+ROF      ;H NEVER CHANGES

                ;BUMP THE QUEUE COUNT. THE RTC INTERRUPT WILL ENABLE THE
                ;UART TRANSMITTER IF NEEDED. THIS WILL USUALLY GIVE TIME
                ;FOR THE CALLING PROGRAM TO QUEUE AT LEAST TWO CHARACTERS
                ;FOR A MULTI-BYTE TRANSMISSION. HAVING QUEUED CHARACTERS,
                ;IN TURN, KEEPS THE TRANSMIT ISR FROM SHUTTING DOWN THE
                ;UART BETWEEN EVERY CHARACTER.

0776 2165F7             LXI     H,STQCNT+ROF
0779 34                 INR     M               ;ATOMIC READ THEN WRITE

077A E1                 POP     H
077B C9                 RET

                ;***EXTERNAL SUBROUTINE***************************
                ; GET KEYBOARD STATUS. IF A CHR IS WAITING,
                ; THEN CONVERT TO UPPERCASE AND RETURN IT IN A.
                ; IF THE CHR IS BS, CONVERT IT TO DEL.
                ; ABORT IF IT IS THE CABKEY (ALT-MODE) OR ESCAPE.
                ; ON EXIT:
                ;   IF A CHR IS WAITING, THEN CHR IS IN A
                ;   CONVERTED TO UPPERCASE, BS CONVERTED TO DEL
                ;   IF NO CHR WAITING OR NULL THEN Z SET, A=0
                ;   INTERRUPTS ENABLED
                ; NOTE: THIS WILL MESS UP '{' AND '|'. THIS IS
                ; OKAY BECAUSE NEITHER IS USED BY POLEX.
                ;*************************************************
077C CD14F7     CHKKBD: CALL    KSTAT+ROF       ;ANYTHING TYPED?
077F C8                 RZ                      ;N: RET W/ Z SET

0780 CD1AF7             CALL    KDATA+ROF       ;Y:GET THE DATA

0783 FE1B               CPI     ESC             ;ABORT?
0785 CAB3F4             JZ      MAIN+ROF        ;YES: MAIN LOOP FIXES STACK
0788 FE7D               CPI     CABKEY          ;OTHER ABORT CHARACTER TYPED?
078A CAB3F4             JZ      MAIN+ROF        ;YES: MAIN LOOP FIXES STACK

078D D0                 RNC                     ;DEL KEY SKIPPED, SINCE
                                                ;CABKEY='}' (7DH)

078E FE61               CPI     'a'             ;LEAVE BELOW a ALONE, BUT
0790 DA07F7             JC      BS2DEL+ROF      ;..TRANSLATE BACKSPACE TO DEL

0793 E6DF               ANI     ('a'-'A') XOR 0FFH ;MAKE IT UPPERCASE

                ;FALL INTO BS2DEL TO RETURN WITH Z CLEARED

                ;***SUBROUTINE********************************
                ; TRANSLATE BS TO DEL
                ; ON ENTRY:
                ;   CHR IN A
                ; ON EXIT:
                ;   CHR IN A, BS CONVERTED TO DEL
                ;   Z CLEARED
                ;*********************************************
0795 FE08       BS2DEL: CPI     BS
0797 C0                 RNZ

0798 C677               ADI     DEL-BS          ;TRANSLATE BACKSPACE TO DEL
                                                ;..AND CLEAR Z FLAG
079A C9                 RET

                ;***SUBROUTINE********************************
                ; GET A KEYBOARD CHARACTER, ABORT IF ALT-MODE
                ; OR ESCAPE
                ; ON EXIT:
                ;    A=KEYBOARD CHR, Z CLEARED
                ;   INTERRUPTS ENABLED
                ;********************************************
079B CDEEF6     GETKBD: CALL    CHKKBD+ROF       ;GET KBD CHR IF THERE
                                                 ;..AND TEST FOR ABORT
079E CA0DF7             JZ      GETKBD+ROF       ;WAIT FOR CHR
07A1 C9                 RET

                ;***CP/M EXTERNAL SUBROUTINE*************
                ; GET KEYBOARD STATUS
                ; ON EXIT:
                ;   A=0 AND Z SET IF NOTHING WAITING
                ;   A=FF AND Z CLEARED IF KBD CHR WAITING
                ;   INTERRUPTS ENABLED
                ; STACK USAGE:
                ;   0 BYTES
                ;****************************************
07A2 3A6FF7     KSTAT:  LDA     KBUFF+ROF        ;ANYTHING THERE?
07A5 C3B0F6             JMP     DOSTAT+ROF       ;RETURN CP/M-STYLE VALUE

                ;***CP/M EXTERNAL SUBROUTINE*************
                ; WAIT FOR AND GET KEYBOARD DATA
                ; ON EXIT:
                ;   A=KBD CHARACTER WITH PARITY STRIPPED
                ;   INTERRUPTS ENABLED
                ; STACK USAGE:
                ;   2 BYTES
                ;****************************************
07A8 E5         KDATA:  PUSH    H
07A9 216FF7             LXI     H,KBUFF+ROF

07AC FB         KDWAIT: EI                       ;IN CASE THEY WERE DISABLED
07AD 3E7F               MVI     A,7FH            ;STRIP PARITY, CHECK FOR CHR
07AF A6                 ANA     M
07B0 CA1EF7             JZ      KDWAIT+ROF

07B3 3600               MVI     M,0              ;CLEAR FLAG
07B5 E1                 POP     H

07B6 C9                 RET

                ;***SUBROUTINE***********************
                ; SCAN PAST BLANK POSITIONS LOOKING
                ; FOR THE FIRST NON-BLANK CHARACTER
                ; ON EXIT:
                ;   Z SET IF NONE FOUND
                ;   Z CLEAR IF CHR FOUND
                ;   A=CHARACTER VALUE IF FOUND
                ;--> ENTRY IS AT SKIPB <--
                ;*************************************
07B7 13         SBLOOP: INX     D                ;NEXT SCAN ADDRESS

07B8 1A         SKIPB:  LDAX    D                ;GET NEXT CHARACTER
07B9 E67F               ANI     7FH              ;VIDEO CHRS HAVE MSB SET

07BB FE20               CPI     ' '
07BD CA29F7             JZ      SBLOOP+ROF       ;KEEP SKIPPING SPACES

07C0 FE7F               CPI     CURSOR AND 7FH   ;END OF INPUT? (CURSOR=FF)
07C2 C9                 RET                      ;CARRY SET IF CURSOR (NO VALUE)

                ;***SUBROUTINE****************************
                ; PRINT INLINE MESSAGE AT (SP)
                ; CALLS TO ILPRNT ARE FOLLOWED BY THE STRING
                ; THE LAST STRING BYTE HAS ITS MSB SET
```

```
               ; PSW TRASHED, ALL OTHER REGISTERS PRESERVED
               ; INTERRUPTS ENABLES ON EXIT
               ;********************************************
07C3 E3        ILPRNT: XTHL                    ;SAVE HL, GET MSG ADDR

07C4 3E7F      IPLOOP: MVI    A,7FH            ;STRIP END-MARKER
07C6 A6                ANA    M                ;..FROM MESSAGE CHR
07C7 CD62F6            CALL   PRINTA+ROF
07CA BE                CMP    M                ;END?
07CB 23                INX    H
07CC CA36F7            JZ     IPLOOP+ROF

07CF E3                XTHL                    ;RESTORE HL
                                               ;..GET RET ADDRESS
07D0 C9                RET

               ;**********************
               ; GENERIC ERROR HANDLER
               ;**********************
07D1 CD35F7    ERROR:  CALL   ILPRNT+ROF
07D4 0DBF              DB     CR,'?'+80H

07D6 C3B3F4            JMP    MAIN+ROF         ;MAIN WILL FIX STACK

               ;********************************
               ; COMMAND TABLE
               ; NOTE THAT ASCII CHRS STORED IN
               ; VIDEO MEMORY HAVE THEIR MSB SET
               ;********************************
07D9 D0CF      COMTAB: DB     'P'+80H,'O'+80H ;RETURN TO POLY-88 MONITOR
07DB 6B                DB     GOPOLY-CMDBAS

07DC D3C2              DB     'S'+80H,'B'+80H ;SET BAUD RATE
07DE 7E                DB     SBCMD-CMDBAS
07DF D4C5              DB     'T'+80H,'E'+80H ;TERMINAL MODE
07E1 00                DB     TERMNL-CMDBAS

07E2 C4D5              DB     'D'+80H,'U'+80H ;DUMP
07E4 10                DB     DUMP-CMDBAS
07E5 C5CE              DB     'E'+80H,'N'+80H ;ENTER
07E7 37                DB     ENTER-CMDBAS
07E8 C5D8              DB     'E'+80H,'X'+80H ;EXECUTE
07EA 54                DB     EXEC-CMDBAS
07EB C6C9              DB     'F'+80H,'I'+80H ;FILL MEMORY
07ED 55                DB     FILMEM-CMDBAS

07EE C8CC              DB     'H'+80H,'L'+80H ;INTEL HEX LOAD
07F0 9E                DB     HEXLOD-CMDBAS
07F1 00                DB     0                ;END OF TABLE MARK

               ;====END OF PROGRAM==========================================

               ;******************************
               ; LOAD-INITIALIZED RAM VARIABLES
               ;******************************

               ;THE ORDER OF THE FOLLOWING THREE MUST NOT CHANGE
07F2 FF        URTDIS: DB     0FFh    ;0 IF TX ENABLED, FF IF DISABLED
07F3 00        STQCNT: DB     0       ;SERIAL TX QUEUE BYTE COUNT
07F4 00        SRQCNT: DB     0       ;SERIAL RX QUEUE BYTE COUNT

07F5 80F7      SRQIPT: DW     SRQ     ;SERIAL RX QUEUE IN-POINTER
07F7 80F7      SRQOPT: DW     SRQ     ;SERIAL RX QUEUE OUT-POINTER
```

```
07F9 A0F7       STQIPT: DW      STQ     ;SERIAL TX QUEUE IN-POINTER
07FB A0F7       STQOPT: DW      STQ     ;SERIAL TX QUEUE OUT-POINTER

07FD 00         KBUFF:  DB      0       ;KBD RECEIVE BUFFER: 0 MEANS NONE

                RAMEND:                 ;END OF COPIED CODE
                ;====END OF CODE COPIED TO RAM=================================

                ;************************************************************
                ; QUEUES IN FIXED RAM LOCATIONS
                ; LOCATIONS CHOSEN FOR EASY CIRCULAR-INCREMENT OF POINTERS:
                ;    SRQ=XXXX XXXX XX00 0000B
                ;    STQ=XXXX XXXX XXXX 0000B
                ;************************************************************
F780                    ORG RQLOC
F780            SRQ:    DS      SRQSIZ

F7A0                    ORG TQLOC
F7A0            STQ:    DS      STQSIZ

                ;********
                ; STACKS
                ;********
F7A4                    DS      VSSIZE  ;STACK FOR CALLS TO VIDEO DRIVER
F7C2 =          VDSTAK: EQU     $

F7F7                    ORG     FXDRAM
F7F7 =          SYSTP   EQU     $       ;MAIN STACK GROWS DOWN FROM HERE

                ;****************************
                ; UNINITIALIZED RAM VARIABLES
                ;****************************
F7F7            MODGO:  DS      2       ;MODULE RAM EXEC ADDRESS
F7F9            BRGSAV: DS      1       ;CURRENT BAUD SETTING ETC

                ;************************************************************
                ; FIXED-LOCATION VARIABLES THAT ARE DEFINED IN THE INTERFACE
                ;************************************************************
F7FA            PTMRISR DS      3       ;TIMER OVERFLOW ISR (RET OR JMP)
F7FD            PTIMER  DS      2       ;2-BYTE 60-HZ COUNT-DOWN TIMER
F7FF            PTOVFL  DS      1       ;<>0 MEANS TIMER OVERFLOWED

                ;============================================================
                ;CHECK FOR ASSEMBLY ERRORS. THE IF DIRECTIVE ONLY TESTS
                ;   BIT ZERO, SO THE SIGN BIT OF THE COMPARISON IS SHIFTED
                ;   DOWN TO BIT ZERO.
                ;============================================================

                ; RAMEND+ROF MUST BE LESS THAN OR EQUAL TO SRQ
                 IF (SRQ - (RAMEND+ROF)) SHR 15
                 ERROR: LOADED CODE OVERWRITES RX QUEUE (SRQ)
                 ENDIF

                ; CODE MUST FIT IN 1K PROM

                 IF (400H-(RAMEND-ROMLOC)) SHR 15
                 ERROR: CODE LARGER THAN 1K
                 ENDIF

F800                    END
```