Kaypro Emergency Monitor ROM
June 10, 2023

Table of Contents

Introduction....1
The ROM Image....1
Using The Monitor....1
Examples....4
  Getting Help....4
  Executing Code and Breakpoints....5
Running Programs....5
Memory Test ROM....6
Adapting 8K ROMs....7
Connecting The Serial Port....8

Introduction

The purpose of the monitor is to help debugging a broken Kaypro. Replace the normal boot/BIOS ROM with an EPROM created from this monitor image. The Kaypro will now interact using the serial port (“Serial I/O” on early models, “Serial Data” on later models). No other hardware will be initialized, which means the display and keyboard will not work, nor will the disk. The display will usually be filled with random characters, or at least will not be cleared.

There is also a emergency memory test ROM available, for cases when the monitor ROM won’t even work.

The ROM Image

The monitor binary image may be downloaded here: http://sebhc.durgadas.com/kaypro/monitor.bin. Note that this is a 4K image and will only work in models capable of taking a 2732 (or larger) ROM. There is a 2K ROM image with reduced functionality (Kaypro-II features only), mon2k.bin, which maybe used in a 2716 EPROM. An image may also be put in larger EPROMs (2732 or 2764) for use with models that support larger ROMs. When using larger EPROMs, it should not be necessary to replicate the image throughout the extra space (although see “**Adapting 8K ROMs**” below for exceptions). Note that the Kaypro IV/83 model (may also apply to some later Kaypro II) has a jumper-selectable ROM socket that takes either 2716 or 2732. Rather than cut/solder the jumper traces, use whatever EPROM is configured. The Kaypro 10/83 has a socket wired exclusively for the 2732. The */84 models should work with either 2732 or 2764 without changing any configuration.

Note that the AT28C16 EEPROM (FLASH) should work in place of the 2716 in Kaypro circuits. The AT28C64 should work in place of the 2764.

Using The Monitor

The monitor ROM does not depend on any hardware besides the SIO1 channel A serial port (and associated baud generator). However, it does provide a “heartbeat” indicator on the drive select LEDs

on */84 (and 10) models – provided that the sysport is operating correctly. On */83 models, this heartbeat will be in the form of the first character on the display alternating between ‘A’ and ‘B’ – provided that the video is working. The heartbeat ceases after the first command is entered.

The monitor prompt is the colon character (‘:’). When the system is RESET the monitor will initialize the serial port and print a signon string and then prompt for input. Input lines may be edited using the Backspace key (Ctrl-H). Ctrl-C will abort the input and go back to the prompt. Upon pressing RETURN, the input line is then parsed by the monitor. Pressing RETURN at the prompt (empty input line) will just re-prompt.

Note that the ROM must be enabled by the sysport in order to operate. Low memory addresses contain the ROM contents and are not writeable. The exact boundary between ROM and RAM depends on the model. Also, the Kaypro II and IV models map the video RAM into addresses 3000-3BFF. In order to access the RAM at low addresses, a special program must be created to transfer control into high memory and update the sysport to turn off ROM (and video RAM). On */83 models, the sysport is implemented in a Z80-PIO which must be initialized before it can be used.

The command ‘?’ will print a brief list of commands with some help text.

The following commands are implemented:

D *start end*

Dump memory in hex. Both *start* and *end* are required, and are interpreted as hexadecimal. Up to 16 bytes are displayed per line, with ASCII representation following the hexadecimal byte values.

S *start*

Substitute memory content interactively. The current address (initially *start*) and byte are displayed in hex and input is accepted. Pressing RETURN will go to the next address (+1) without changing the current byte. Pressing dash (‘-’) will go to the previous (-1) address without changing the current byte. Pressing period (‘.’) will end the command and return to the monitor prompt. Entering hex digits and RETURN will replace the current byte with the new value and advance to the next address (+1).

G *start*

Go (jump) to *start* and begin executing the code there, after prompting for confirmation. If the code executes any RST instruction, control will be returned to the monitor, where the message “*** RST” and the calling address (after the RST instruction) will be printed. No registers are saved and the code may not be resumed.

F *start end data*

Fill memory with the byte *data* repeated throughout the range.

M *start end dest*

Move a block of memory. If *start end* overlaps with *dest* then the results are undefined.

I *port* **[** *num* **]**

Input from *port* and display the value in hex. If *num* is given, then the same port is read that number of times, and each value displayed. Both *port* and *num* are hexadecimal. There is no delay between inputs, other than the time it takes to print the value in hex.

O *port data* **[** ... **]**

Output *data* to *port*. If more than one data bytes are given, then those values are successively output to *port*. There is no delay between outputs, other than the time it takes to parse the next value.

N *hw*

iNitialize a hardware component. *hw* may be:
KB83 – Initialize the keyboard SIO channel on a */83 model.
KB84 – Initialize the keyboard SIO channel on a */84 or 10 model.
CRTC – Initialize the CRT controller chip on a */84 or 10 model.
HDD – Initialize (perform RESET of) the HDD controller on Kaypro-10 models. Output is similar to **T HDD**.

T *hw* [*args...*]

Test a hardware component. *hw* may be:

KBD

Run a simple test for receiving codes from the keyboard. Waits for keys to be pressed on the keyboard and prints the hexadecimal code that is received. To end the test, type any character on the monitor console.

CRTC

Run a simple test for CRTC update status. Tries to detect the status change three times, and prints “Update” for each. The test may be aborted, if hung, by typing any character on the monitor console.

VRT

Run a simple test for CRT vertical retrace activity. Gathers up to 10 transitions of the VRT bit in the status register, then displays results. Output is similar to that for FLPY, but the 6845 status register is being watched instead. Normal VRT rate seems to be about every 20 milliseconds (50Hz).

CRTR

Run a pass/fail integrity test for CRTC register access. Since the only CRTC registers that can be both read and written are the cursor position hi/low registers, this test runs through all possible values 0000-3FFF and checks that the written value can be read back. If fail, prints the expected and actual values for the cursor position of the first failure (test stops).

CRTF [*fill* [*attr*]]

Fill video RAM with *fill* byte, optionally setting video attributes *attr*. The same character and attribute value is used for all locations on the screen. Default fill is the blank space character, default attributes is 00 (none).

FLPY [*cmd* [*count*]]

Run a test of the FDC system. User must first turn on motor and select a drive, and may need to have a floppy diskette in the drive. Default command is FORCE INTERRUPT. Default count is 256 (maximum number of changes recorded).

FDRD [*sector*]

Do a floppy disk read of one sector into 8000h. User must first turn on motor, select a drive, select DD, select a side, and must have a formatted floppy diskette in the drive. Note that *sector* is the sector ID value to be matched against data formatted on the diskette, and is enetered in hexadecimal. Default *sector* is 00.

HDD [*cmd* [*count*]]

Run a test of the HDD system. Issues a command (default Self Test) to the WD1002 and captures the changes in the status register, similar to **FLPY**.

HDRD

Read the WD1002 data buffer into 8000H. If DRQ is not active, reads nothing. Prints final address (last byte + 1). Typically used after “T HDD 20”, with setup of other registers as needed.

V

Display the ROM version. This just prints the ROM signon message again, which contains the version.

Several of the test and initialization commands will watch for changes on their respective “status” registers and display the results as a list of new register values and iteration counter at the time of the change. Each entry is printed in the form “SS NNNN” where “SS” is the register value in hexadecimal and “NNNN” is the iteration counter, also in hexadecimal. The iteration count values are in units of 12.25uS, i.e. one count represents 12.25uS on the 4MHz Kaypros (*/84, 10). On the */83 Kaypros (2.5MHz), this changes to 19.6uS. These tests only run until the iteration count wraps (reaches FFFF). When this limit is reached, a sample is always stored regardless of whether the register value changed. The commands using this method are: “T FLPY”, “T HDD”, “T VRT”, and “N HDD”. When making precise timing measurements for short events, factor in that recording each change adds 23.25uS (37.2uS). So, events that occur more frequently than 35.5uS (56.8uS) can’t be resolved.

Examples

In the examples, characters typed by the user are underlined. *(CR)* means pressing the RETURN key.

Getting Help

This shows the signon message and how to display the help menu.

```
Kaypro Monitor v2.6

: ?(CR)
D <start> <end> - display memory in HEX
S <start> - set/view memory
    (CR) = skip fwd, '-' = skip bkwd, '.' = done
G <start> - go to address
F <start> <end> <data> - fill memory
M <start> <end> <dest> - Move data
I <port> [num] - Input from port
O <port> <value> [...] - Output to port
N <hw> - iNitialize hardware (KB83, KB84, CRTC, HDD)
T <hw> - Test hardware
  (KBD, CRTC, VRT, CRTR, CRTF, HDD, HDRD, FDRD, FLPY)
V - Show ROM version
```

```
^C aborts command entry
:
```

Executing Code and Breakpoints

Set a RST 7 instruction into memory and jump to it.

```
: S8000(CR)
8000 00 FF
8001 00 .
: D8000 800F(CR)
8000 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
: G8000(CR)
GO 8000 ?Y
*** RST 8001

:
```

Running Programs

Long and complex programs are not well-suited for use with this monitor, mainly due to the process required to enter them into memory. Some things that may help:

If you have an assembler available (i.e. a cross-assembler running on your PC), you can write and assemble the code, and use the listing file to read the stream of hexadecimal values to be entered into memory. For example, the program used to strobe an I/O port select line would be:

```
8000                    org     8000h
8000  DB00      loop:   in      00h
8002  C30080            jmp     loop
8005                    end
```

Where the I/O port “00h” would be replaced by the desired I/O port to access. This then shows the hex bytes to be entered at 8000: DB 00 C3 00 80.

The monitor ROM provides some entry points to utility functions that programs may use.

Entry	**Parameters**	**Function**
0003H	C=character	Console output (monitor serial port)
000BH	A=value	Print value in hexadecimal on console
0013H	HL=string	Print NUL-terminated string on console
001BH	none	Send CR/LF to console
0023H		reserved
002BH		reserved
0033H	C=port, B=init, A=count	Samples port for changes, starting with initial value, recording a maximum count of samples. Returns DE pointing to end of sample buffer. Sample buffer begins at 8000H.
003BH	DE=end of buffer	Prints all samples in sample buffer.

An example for using the entry points to gather a sample of the keyboard SIO transmit operation would be:

```
F000                    org     0f000h
F000  0E07              mvi     c,07h   ; SIO1B status reg
F002  0600              mvi     b,0     ; initial value
F004  AF                xra     a       ; 00 is NOP to kbd
F005  D305              out     05h
F007  D305              out     05h     ; fill Tx
F009  CD3300            call    0033h   ; gather samples (A=0)
F00C  CD3B00            call    003bh   ; print samples
F00F  FF                rst     7       ; return to monitor
F010                    end
```

Entering the above program at F000 and then doing “G F000” will run and print the samples, indicating whether the SIO is functioning correctly enough to transmit characters, and also giving a hint as to the baud rate (by computing the time it takes to empty the Tx buffer). Note that the sample buffer is at 8000H, so the program needs to reside outside of that area (e.g. F000H). Output would look something like:

```
68 0000
6C 09EB
6C FFFF

*** RST F012
```

This program must be run after initializing the keyboard serial port. The Tx Empty bit is 04H. The iteration count of of 09EB computes to approximately 31.1mS which is roughly 30 chars/sec or 300 baud. The computation won’t be exact because of the particular timing of when the Z80-SIO sets the Tx empty status. Two character must be sent back-to-back because the Z80-SIO has a Tx shift register and a Tx holding register, and both must be filled to get a reasonable result.

Memory Test ROM

The memory test binary image may be downloaded here:

http://sebhc.durgadas.com/kaypro/memtest.bin

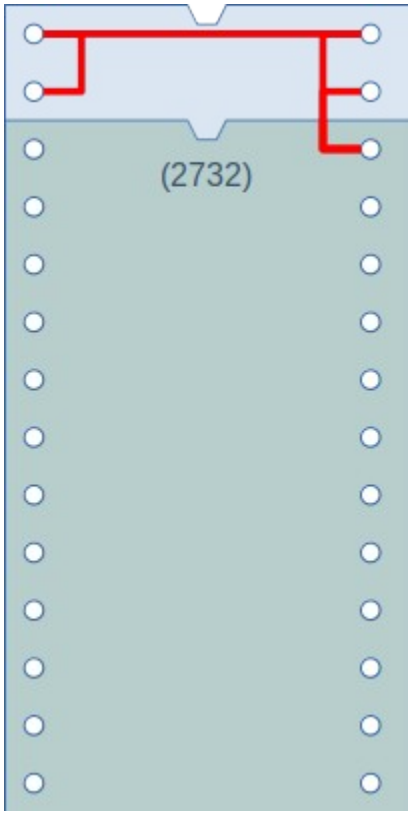
This image does not depend on RAM to operate (uses no stack or RAM variables) but it does use the serial port for progress (output) only. It does a memory test in the range 8000H-FFFFH and will alternate drive A and B LEDs (*/84 and 10 models) or cycle through ‘0’ to ‘9’ in the first character of the display (*/83 models). If a memory error is detected, it will flash the drive A LED or alternate ‘E’ and ‘R’ in the first character of the display (depending on the Kaypro model).

Serial port output begins with a signon message showing the test version. During the test, it will output the current “seed” value being run (00-99 BCD). If an error is encountered, it will print the error address, expected value, and actual value (and “Error”).

The test method is to write a BCD value 00-99H to each memory location. The seed value dictates the value put in location 8000H, with each successive location receiving the previous value +1 (BCD). On each pass, the seed value is incremented (BCD).

Adapting 8K ROMs

Using an 8K ROM (2764, 28C64, etc) on a Kaypro that requires a 2732 may be done using an adapter. There are adapter kits available online, or you may build one from a 28-pin socket (machined pins recommended). The following diagram shows how to wire the socket, which is inserted into the 2732 socket with the top overhanging (aligned on bottom pins). Note that the wire soldered to pin 26 must not interfere with insertion into the 2732 socket. Pins 1, 2, 27, and 28 may need to be clipped to avoid interference with components on the mainboard.



Using a socket adapter requires that the 28C64 be programmed with the image in the upper half (1000H-1FFFH), or else replicated in both halves.

This same adapter could be used to put a 28C64 in a 2716 socket, but only a 2K image can be used and must be programmed into the upper quarter (1800H-1FFFH) of the 28C64 (or replicated into all quarters).

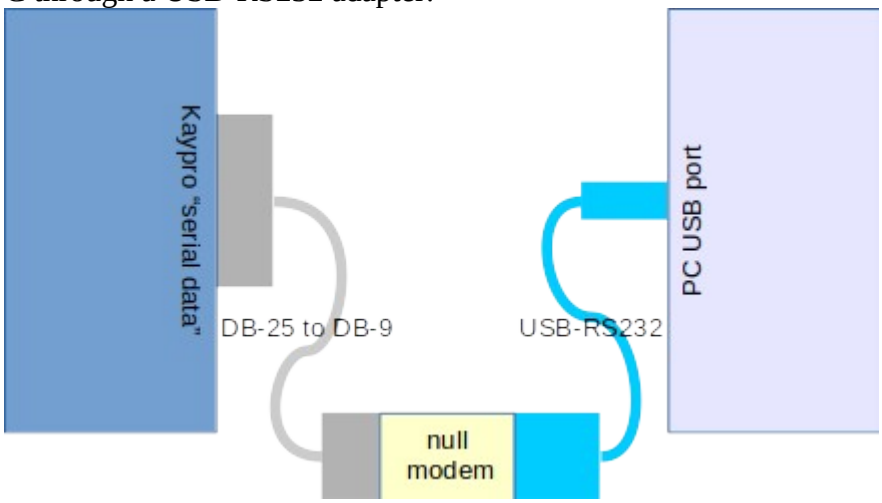
Connecting The Serial Port

The following setup should work for connecting the Kaypro serial port to a PC (either legacy serial port or USB-RS232 device). The Kaypro uses a DB-25F, and the standard PC (and USB) serial connector is a DB-9M. The typical DB-25M-to-DB-9F cables and adapters seem to have this pinout:

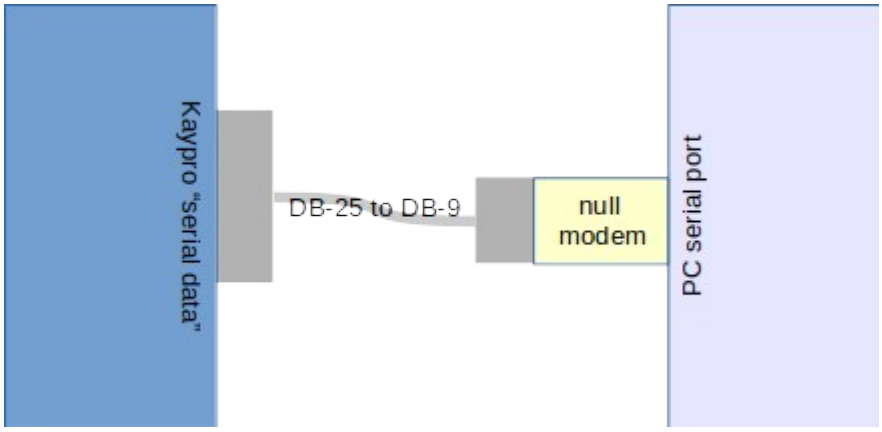
D-sub 9 F		D-sub 25 M	
DCD	1	8	DCD
RXD	2	3	RXD
TXD	3	2	TXD
DTR	4	20	DTR
GND	5	7	GND
DSR	6	6	DSR
RTS	7	4	RTS
CTS	8	5	CTS
RI	9	22	RI

Using a cable or adapter with this pinout, a “null modem” may still be required between the Kaypro and the PC. These diagrams show the hookups:

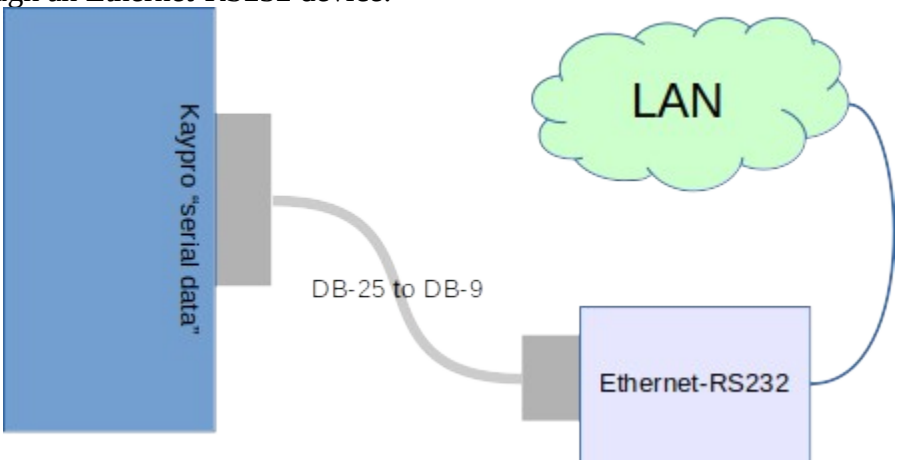
Connecting to a PC through a USB-RS232 adapter:



Connecting to a PC that has a legacy serial port:



Connecting through an Ethernet-RS232 device:



The configuration of the Ethernet-RS232 device is beyond the scope of this document. Attaching a terminal program over the LAN is also beyond the scope of this document. A gender adapter may be required, or even a null-modem, depending on the model of Ethernet-RS232 device.

Note that attaching a terminal program to an Ethernet-RS232 may result in some “garbage” characters initially. This is often the result of telnet negotiation, which may not be handled correctly. The terminal program should not be echoing and needs to operate in a “character mode” (as opposed to a “line mode”). For telnet, this is the “mode character” command.