PHUNSY
Signetics 2650
Micro Computer

by Frank Philipse

Contents

Document History

2010-10-23 initial
2010-11-04 added some stuff about the PHUNSY emulator.
2010-11-05 Chapter 1.3 updated
2010-11-05 Chapter 4 added
2010-11-09 Chapter 2.1 added
2010-11-11 Several additions across the entire document
2010-11-12 Chapter 3 corrected ram area for MDCR program
2010-11-12 Chapter 3 updated
2010-11-12 Chapter 7 added
2010-11-14 Chapter 6 updated
2010-11-28 Chapter 6 updated
2010-11-28 Chapter 8 added
2010-11-28 Chapter 9 added
2010-12-12 Chapter 8 updated
2010-12-12 Chapter 11 added
2013-01-27 Chapter 6 some corrections
2014-01-04 Changed Simulator into Emulator everywhere

1 PHUNSY

The PHUNSY is a 2650 microcomputer system developed in the early 1980's. It consisted of a 19" cabinet with a bus board. Several euro-cards were developed.

- CPU board
- Video board
- User i/o board
- Several memory boards
- Hard-disk controller board
- Universal I/O board

Ben Postema describes the hardware in a different document.

This document describes several software programs that were developed for the PHUNSY or that were adapted to run on the PHUNSY

PHUNSY is short for Philipse Universal System

1.1 I/O

The I/O of the PHUNSY is done by means of the control and data ports of the 2650. These 8-bit ports are easy to decode by hardware and the instruction set supports several I/O instructions to access the I/O ports. Note that the extended I/O is not used by the basic PHUNSY.

1.1.1 Data port inputs

This port is used to read an ascii keyboard. Bit 7 = 0 indicates a character is entered. Bit 7 remains 0 until it is cleared by the KEYBACK bit in the data port output.

1.1.2 Data port outputs

name	bit	description
TTYOUT	001h	serial data out to tty
SPEAKER	002h	output to small speaker
KEYBACK	004h	keyboard acknowledge
MDCRTST	008h	mdcr test cassette in position ?
MDCRREV	010h	reverse tape
MDCRFWD	020h	forward tape
MDCRWCD	040h	write command
MDCRWDA	080h	write data

1.1.3 Control port inputs

name	bit	description
TTYIN	001h	serial data in from tty
PARALEL	002h	if '1' keyboard and screen else tty
BPS300	004h	if '1' 300 bps else 110 bps tty
MDCRWEN	008h	mdcr write enable
MDCRCIP	010h	mdcr cassette in position
MDCRBET	020h	mdcr write enable
MDCRRDA	040h	mdcr read data
MDCRRDC	080h	mdcr read clock

1.1.4 Control port outputs

This port selects the memory banks.
Bit 0-3 are used for the Q bank (4000-7FFF).
Bit 4-7 are used for the U bank (1800-1FFF).

1.1.5 Flag & Sense

The monitor uses the FLAG and the SENSE (residing in the program status bytes of the 2650) for storing data to and retrieving data from an audiocassette recorder.

1.2 Memory

The address space of the 2650 is limited to 32768 locations. However, in the PHUNY two parts of the address space is banked.
1800-1FFF is called the U bank and can have 16 banks controlled by the control port bits 4-7.
4000-7FFF us called the Q bank and can have 16 banks controlled by the data output port bits 0-3.

address range	**usage**
0000-07FF	monitor in eprom
0800-0BFF	free memory
0C00-0FFF	ram used by monitor and MDCR
1000-17FF	display ram
1800-1FFF U0	general purpose ram
1800-1FFF U1	MDCR program in eprom
1800-1FFF U2	Disassembler in eprom
1800-1FFF U3	label handler in eprom
1800-1FFF U4-UF	not used
2000-3FFF	general purpose ram
4000-7FFF Q0-QF	general purpose ram

1.3 Display

The display memory is located in the address range 1000-17FF. The hardware converts this part of the ram to a screen of 32 lines with 64 characters. The order of the displayed characters is the same as the order in memory. The character in address 1000 is at the top left position and the character at location 17FF is displayed in the bottom right position. The memory is transparent so read or write actions do not cause interruptions in the display. The output signal is black & white with the 625 lines system as used in Europe. However, the lines are not interlaced so the actual number of lines is 625 divided by two.

1.3.1 ASCII display

Character ROMs are used to convert each character in the display memory to a 6*8 dot matrix. All characters from 20-7F are displayed according to the ASCII standard. There are two character ROMs type 82S115. The 1st ROM displays the bytes 20-5F. The 2nd ROM displays characters 60-7F and 00-1F. The control characters 00-1F are normally not displayed but the 2nd character ROM contains Greek symbols in the control character area.

1.3.2 GRAPHIC display

If bit-7 of a byte in the display memory is 0, the character ROM is decoding the character to a display. If bit-7 of the byte in memory is set to 1, the byte is displayed in graphic mode. The resolution of graphic display in the total screen is 128x64 'dots'. Each dot consists of ¼ of the space normally used for an ASCII character. Bit 0 to 3 of a byte in the display memory control four dots. 1 is on, 0 is off. Of each dot, the intensity can be controlled from black to white in 8 steps. Bits 4 to 6 of the displayed byte control the intensity of the four dots. Note: Each dot is 4 lines high and 3 dots wide.

1.4 MDCR

MDCR = Mini Digital Cassette Recorder
The MDCR, made by Philips, is used by the PHUNSY as main data storage. The mini cassette tapes can store 32KB on each side. Data storage of the cassette is divided in 128 blocks of 256 bytes and the transfer speed is 6000bps. However, as the gap space between the blocks is about the same as the space of a block on tape, the effective speed is some 3000 bps. It takes just over 90 seconds to get from the beginning to the end of the tape.
See the MDCR manual for details.

1.5 Audio cassette interface

The Flag and the Sense bit are special I/O bits of the 2650 that can be accessed form the upper status byte of the CPU. These I/O bits are used for I/O to an audiocassette recorder. Data can be stored on and retrieved form an audiocassette.
I don't remember the format but it was copied from Apple.

1.6 TTY interface

2 Monitor

The monitor program resides in memory space 0000-07FF. Ram used is from 0EE0-0FFF. 0F00-0FFF is the command-input buffer.

2.1 Screen display routine

The screen consists of 64x32 characters. The display routine in the monitor treats this as 32 lines of 64 characters.
Beside the normal character display the display routine interprets several control characters and performs an operation when a valid control character is presented.
These control characters and their function are:

Table 1 Screen control bytes

hex	dec	function
00	0	Reset screen, margins are set to max
05	5	Clear from cursor to end of line
06	6	Clear from cursor to end of screen
07	7	bell (beep to speaker is issued
08	8	backspace, move cursor one position left
0C	12	Clear screen
0D	13	carriage return, cursor to next line, left most position
11	17	move cursor to left most position
12	18	move cursor to right most position
13	19	move cursor to top of screen
14	20	move cursor to bottom of screen
15	21	move cursor one position right
1C	28	move cursor one position right
1D	29	move cursor one position left
1E	30	cursor one position lower
1F	31	cursor one position higher

When a carriage return is issued and the cursor is at the bottom line, the lines are shifted up one line. The top line is lost.

The screen routine has a zero branch entry. The instruction 'ZBSR WRTZV' will send the character in R0 to the screen routine. CPU registers R1..R3 in register bank 0 are not affected.

Four bytes define the margin of the screen so it is possible to use only part of the screen and use the other part e.g. for graphic purposes.

Table 2 screen margin locations

address	name	function
0EF0	UPSTRE	upper start position
0EF1	LESTRE	left start position
0EF2	LOENRE	lower end position
0EF3	RIENRE	right end position

2.2 Monitor commands

The following characters are recognized as commands:
-:=>GIMPQRSTUVWX

Commands usually start with zero to three hex values followed by the command character. Multiple commands may be entered on the same line separated by a space. The command line is executed after the <cr> is key is entered. The hex values are separated by special characters such as '-' and '>'.
The monitor expects all uppercase letters.

2.2.1 Change memory (:)

To change bytes in memory type the 1st memory address followed by a semicolon. Then type the new bytes separated by spaces.
Example: **800: 48 55 4E 53 59** sets the word PHUNSY at address 800. Leading zeros are not required.

A '.' after the last byte will terminate the command after which a new command can be issued on the same line.

```
*800: 0 1 2 3 AA BB CC DD.800-807I
0800: 00 01 02 03  AA BB CC DD
*
```

Figure 1 Change memory and Inspect memory

2.2.2 Preset memory (P)

This command enables the user to fill part of the memory with all the same byte.
Example **800-8FF>58P** will fill the given memory range with 'X'.

2.2.3 Inspect memory (I)

To display memory in hex, the I command is used.
Example: **800-8FFI** will display the contents of this address range in hex.
Example: **800I** displays only one address location.
Example: **800III** displays the 800 to 802 memory contents.

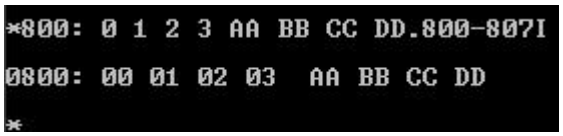


Figure 2 Inspect memory example

2.2.4 Go execute program (G)

To be able to run a program that is loaded in memory, the G is used.
Example: **800G** will cause the 2650 to run instructions starting at address 800. If the program is returned as a subroutine, it returns to the monitor and interpreting commands from the command line is resumed.

2.2.5 Move memory (M)

To move part of the memory contents to a different location, the M command is used.
Example: **800-8FF>900M** will copy the contents from 800-8FF to the target address 900.
The source and the target ranges may overlap. The monitor decides weather to start at the beginning or at the end of the address range.

2.2.6 Verify memory (V)

To check if two memory areas contain the same data the verify command is used.
Example: **800-8FF>900M** will compare two areas in memory. Differences are displayed.

2.2.7 U bank select / execute (U)

The memory range 1800-1FFF has 16 banks. This command is used to select the desired bank.
Example: **3U** will select bank 3.
To start execute a program at 1800, the U command is issued without bank select value.
Example: **U** will cause the CPU to execute a program at address 1800. This is the same as command 1800G.
Example: **2UU** selects bank 2 and starts a program at 1800.
Note: After reset the selected bank is bank 0.

2.2.8 Q bank select / execute (Q)

The memory range 4000-7FFF has 16 banks. This command is used to select the desired bank.
Example: **3Q** will select bank 3.
To start execute a program at 4000, the Q command is issued without bank select value.
Example: **Q** will cause the CPU to execute a program at address 4000. This is the same as command 4000G.
Example: **2QQ** selects bank 2 and starts a program at 4000.
Note: After reset the selected bank is bank 0.

2.2.9 Display selected banks (=)

To display the current selected banks the = command is used.
Example: **=** will display a byte indicating both the U and the Q bank in hex. The most significant digit displays the U bank and the least significant digit displays the Q bank.
This is the same data as present at the

2.2.10 MDCR command (T)

This command is used to issue MDCR commands. The MDCR is the main storage medium of the PHUNSY for storing files. This command selects U bank 1 and executes program instructions at address 1800. In fact this is short for the command 2UU or 2U1800G.
Example: TCAT will show the catalog of the tape residing in the MDCR.
After this command is done, the MDCR program returns to the monitor so that commands following TCAT (e.g. TREW) can be interpreted from the command line.

2.2.11 Write audiocassette (W)

This command us used to write data from memory to an audiocassette.
Example: **800-8FFW** will output the digital data in the form of an audio signal that can be recorded on an audiocassette.

2.2.12 Read audiocassette (R)

This command is issued to read data from an audiocassette to memory.
Example: **R** will cause the program to be stored at the same address it had when it was put on the audiocassette.
Example **800R** will read the audiocassette and store the data at address 800.

2.2.13 Show ERROR message (S)

The command S will show the error message.
This is probably not very useful.

2.2.14 Show monitor id (X)

The command X will show the monitor id.

3 MDCR

MDCR = Mini Digital Cassette Recorder
This is a small digital storage device made by Philips in the 1080's. It uses small digital cassettes. The cassettes are of the same size as Philips made for their audio dictation recorders.

Figure 3 Philips Mini Cassette

The MDCR program is located in memory space 1800-1FFF in U-bank 1. RAM space used is 0C00-0E4F.

Note: In the real world, you can see and hear the cassette action. The emulator will give an indication of the position of the cassette in percents.

3.1 MDCR commands

From the monitor all MDCR commands can be given by typing a T. (T stands for tape.) T causes the monitor to select bank U1 and to jump to address 1800. From there the rest of the command is interpreted.

3.1.1 Catalog

TCAT reads the first block of the inserted cassette and displays the contents of the cassette.
This must be done before any file is read from the cassette.

1. The first column (CN) indicates the position in the catalog block.
2. The second column shows the file name (1 to 8 ASCII characters).
3. The third and fourth column shows the address range.
4. The fifth column indicates if the file is locked.
5. The sixth column indicates the position on the cassette. There are 128 blocks of 256 bytes.
6. The seventh column indicates the language. Some codes have been assigned:
   - 00 undefined, can be anything.
   - 01 2650 code that can be executed from the first address location.
   - 02 plane ASCII text.
   - 03 basic program in ASCII (no special codes).
7. The eighth column lists the bank address. See Chapter 1.2.
8. NR shows the file part. Files may consist of more then one (upto 15) part. Whenever a file is loaded, all parts are loaded all with their specific parameters.
9. The hex number at the right (BU:) indicates the number of blocks used. Total number of blocks for payload is 7F (127). The catalog block is included in the displayed number so the max number is 80 (128).

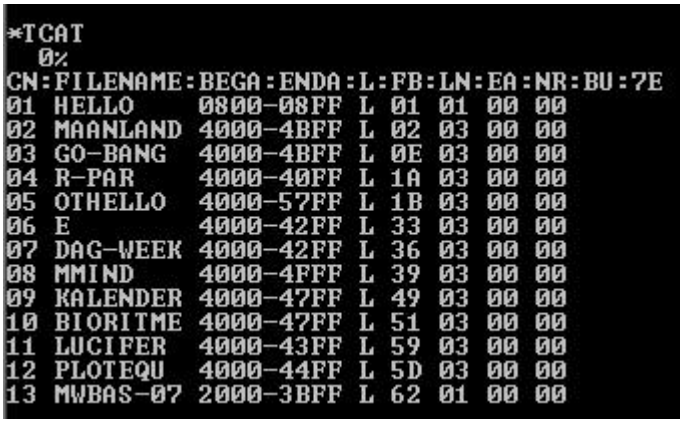


Figure 4 Catalog example

(Note: The '0%' in Figure 4 is printed by the emulator indicating the cassette position in percents.)

TCAT,X will add some mdcr program information.

3.1.2 Load File

TLOAD filename1; filename2 ; etc
This will load a file from cassette to memory according to the specified parameters in the catalog.

3.1.3 Save file

This command saves a file from memory to the cassette.

Command syntax:
TSAVE aaaa,bbbb-cccc,dL,eU,fQ,gOFh

aaaa	file name	1 to 8 characters
bbbb	start address	any memory address
cccc	end address	the last byte saved
d	language	0 to FF
e	U bank number	0 to F, bank memory 1800-1FFF
f	Q bank number	0 to F, bank memory 4000-7FFF
g	part of file	0 to F, files may consist of multiple parts saved separately but loaded all in one command
h	number of parts	indicates the number of parts of a file

If only the file name is entered, the MDCR program will check the catalog for that filename. If present, the file is saved, using the parameters from the catalog.

3.1.4 Rename

TRENAME oldfilename1 (;) newfilename1; oldfilename2 (;) newfilename2 ; etc

This command will rename a file.

3.1.5 Lock

TLOCK filename1; filename2; etc

This command sets the lock-bit in the catalog. It prevents the save and delete command to overwrite or erase the program.

3.1.6 Unlock

TUNLOCK filename; filename2; etc

This command clears the lock-bit.

3.1.7 Initialize

TINIT

This command initializes the cassette regardless of what is on the cassette. All data will be lost. It writes 128 blocks with 256 zero-bytes to the cassette. Each block is numbered with its block number.

Note: In the emulator this command has no effect. A cassette file with the first 256 bytes set to 0 represents an initialized cassette. Cassette files are 32768 bytes long (128 blocks of 256 bytes).

4 Disassembler

Typing 2UU from the monitor starts the disassembler. It has a new prompt character '.'. The command syntax is similar to that of the monitor.

4.1 Disassembler Commands

The disassembler has several commands that are equal to the monitor commands. In fact, the routines from the monitor are used to execute them. These commands are:

- Change memory (:)
- Inspect memory (I)
- Move memory (M)
- Verify memory (V)
- Preset memory (P)
- Go execute program (G)
- Display bank (=)
- Q-bank select / execute (Q)

See Chapter 2.2 for a description of these commands.

4.1.1 Disassemble (X)

The X command is used to display the code in a specified memory area.
Example: **800-83FX** will show the code form 800 to 83F.
If a table for labels is defined in memory containing valid labels, the labels are printed together with the mnemonics of the instructions.

4.1.2 Set Breakpoint (S)

If a program to be debugged resides in RAM, a breakpoint can be set on a memory address. When the breakpoint is encountered, the program will be interrupted and the CPU registers are displayed.
Example: **808S** sets a breakpoint to at address 808. If the program starts at address 800, type 800G. When the instruction on 808 is reached the program stops and the CPU registers are displayed. After that, type 'C' to continue the program or any other character to abort.
Note that the breakpoint replaces two bytes of the program in order to jump to the register display routine. The original bytes are replaced when the breakpoint occurred. When the program is continued, all registers of the CPU are restored.

4.1.3 Select Table area for labels (T)

The disassembler can work with labels. That usually makes code readable. The labels are stored in a table in RAM.
Example: **2000-27FFT** indicates to the disassembler that labels can be found at address 2000 up to 27FF.
It is possible to save the table area with labels to e.g. MDCR tape and reload it when it is required again. Load the data and define the table or vice versa.

4.1.4 Add a label (()

A label consists of 1 to 6 characters and is accompanied by an address.
Example: **18CE(DSASS)** will assign the label DISAS to address 18CE. When the disassembler encounters address 18CE it will add the label to the disassembled code. See Figure 5. To remove a label, use the same command but without any contents between the brackets. When the defined table area for labels is full, 'ERROR' will be shown.

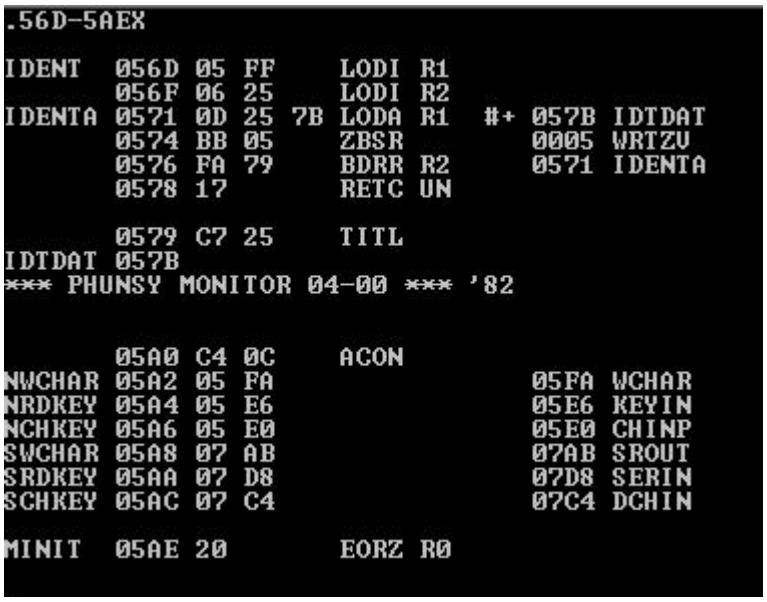
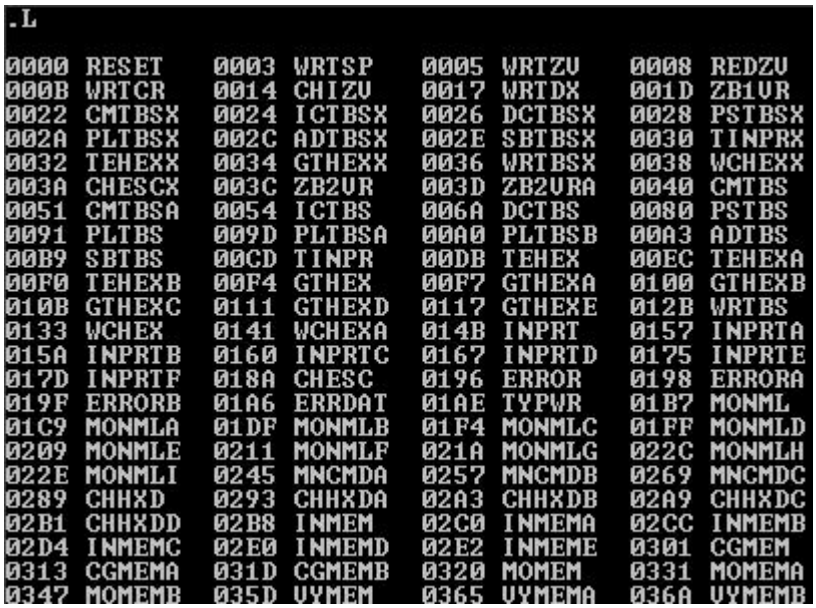


Figure 5 disassembler output example

4.1.5 List labels (L)

Example: **L** will list all labels to the screen.



```
.L

0000 RESET   0003 WRTSP   0005 WRTZV   0008 REDZV
000B WRTCR   0014 CHIZV   0017 WRTDX   001D ZB1VR
0022 CMTBSX  0024 ICTBSX  0026 DCTBSX  0028 PSTBSX
002A PLTBSX  002C ADTBSX  002E SBTBSX  0030 TINPRX
0032 TEHEXX  0034 GTHEXX  0036 WRTBSX  0038 WCHEXX
003A CHESCX  003C ZB2VR   003D ZB2VRA  0040 CMTBS
0051 CMTBSA  0054 ICTBS   006A DCTBS   0080 PSTBS
0091 PLTBS   009D PLTBSA  00A0 PLTBSB  00A3 ADTBS
00B9 SBTBS   00CD TINPR   00DB TEHEX   00EC TEHEXA
00F0 TEHEXB  00F4 GTHEX   00F7 GTHEXA  0100 GTHEXB
010B GTHEXC  0111 GTHEXD  0117 GTHEXE  012B WRTBS
0133 WCHEX   0141 WCHEXA  014B INPRT   0157 INPRTA
015A INPRTB  0160 INPRTC  0167 INPRTD  0175 INPRTE
017D INPRTF  018A CHESC   0196 ERROR   0198 ERRORA
019F ERRORB  01A6 ERRDAT  01AE TYPWR   01B7 MONML
01C9 MONMLA  01DF MONMLB  01F4 MONMLC  01FF MONMLD
0209 MONMLE  0211 MONMLF  021A MONMLG  022C MONMLH
022E MONMLI  0245 MNCMDA  0257 MNCMDB  0269 MNCMDC
0289 CHHXD   0293 CHHXDA  02A3 CHHXDB  02A9 CHHXDC
02B1 CHHXDD  02B8 INMEM   02C0 INMEMA  02CC INMEMB
02D4 INMEMC  02E0 INMEMD  02E2 INMEME  0301 CGMEM
0313 CGMEMA  031D CGMEMB  0320 MOMEM   0331 MOMEMA
0347 MOMEMB  035D VXMEM   0365 VXMEMA  036A VXMEMB
```

Figure 6 Label list example

4.1.6 Return to monitor (Y)

Example: **Y** will return to the monitor.

5 Label Handler

The label handler can be used to perform some operations on the table with labels defined with the disassembler. From the monitor, type 3UU to invoke the label handler.

5.1 Label Handler Commands

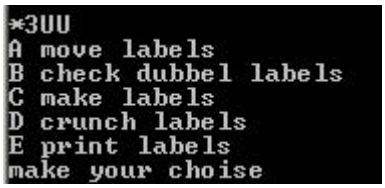


Figure 7 Label Handler Commands

The label handler has 5 commands and after each command label handler returns to the monitor. The label handler performs operations on a list of labels that was made with the phunsy disassembler. With the disassembler an area in memory can be defined and labels can be entered. These labels are used when the disassembler lists disassembled code.

5.1.1 Move Labels (A)

If the program you are writing or part of it was moved to a new location, the move labels command can be used to adjust the addresses of the labels to the new location.

Note: Programs or parts of programs can be moved by means of a program called MODEST. This is a line assembler that has some very nice features. When it moves a program, or part of it, all addresses in the source code are recalculated. This however requires special coding of your program. E.g. loading part of an address immediately in an 8-bit cpu register will not be recognized. So instead ACON's (address constants) must be used.

5.1.2 Check Double Labels (B)

This command checks if an address is used more then once and lists the double labels with their addresses.

5.1.3 Make labels (C)

This command generates labels. After an address range is entered the 2650 code found is disassembled and for each effective address a label is added in the table. These labels can be altered later for more meaningful ones.

5.1.4 Crunch Labels (D)

This command puts all the labels in the list in sequential order. The list is resized to the minimum required space by using deleted label space. All labels are sorted by address.

5.1.5 Print Labels (E)

This command prints labels. An address can be entered, pointing to a special printer routine. Default it will print to screen.

6 Emulator

Version 2010-11-28

The PHUNSY emulator is written in C for GCC. It is a dos-box program, which simulates the 2650 CPU, user I/O, the MDCR and the original screen.

The program is invoked from command.com. Syntax:

phunsy -f codefilename -c cassettefilename -l logfile -t address length

The code file name should contain code for the 2650. The file must be in Motorola S1, S2 or S3 format. S0 has a special meaning. If the S0 string contains the word PHUNSY in hex followed by one byte, bank switching is done and consecutive code is placed in the selected bank (U and Q). See Chapter 1.2.
The -f option can occur more then once to load more code.
The -c option can occur 4 times to load 4 cassettes.

All original ROMs and EPROMs don't have to be loaded as this code is included in the PHUNSY Emulator.

The cassette file name loads a file that is formatted the same as a PHUNSY MDCR cassette. It is 32768 bytes in length.

An output file can be created with the -l option. It enables output form the phunsy. A write to extended i/o address 127 (0x7f) adds the byte to the log-file. This can be used for e.g. printer output.

The -t option enables a trace of code. 'Address' is the address where to start the trace and 'length' is the number of instructions displayed.

To exit the emulator close the PHUNSY window by clicking X in the upper right corner. If one of the cassettes was changed, the cassette image is saved as: cx.mdcr where x is the cassette number 1 to 4.

The emulator stops when an invalid instruction, the HALT instruction has been encountered or when a write to the Monitor or MDCR ROM is done. The last 10 instructions are listed on screen.

To change cassette, use the function keys F1 to F4. Do not enter one of these function keys when the cassette is active. First rewind the cassette by typing TREW or TCAT.

7 MicroWorld Basic interpreter

The MicroWorld basic interpreter has been adapted to work on the PHUNSY microcomputer. This means that the complete interpreter was moved to a different location. Originally it was located in bank0 of the 2650 (0000h-1FFFh) now it resides in bank1 (2000h-3FFFh). The MicroWorld Basic manual lists several options (patch locations) that now have different address locations.

The basic interpreter has to be loaded and executed at address 2000h.
The basic program has to be loaded at address 4000h.

After the interpreter and a basic program have been loaded, type 2000G from the phunsy monitor.
A sign-on message will appear and a prompt.
To let the interpreter know a program is available, type O(LD)<cr>.
If a new program is to be written, type N(EW)<cr>.
To list the program type L(IST)<cr>
Listing part of the program, type L(IST) n1 (n2)<cr> where n1 is the first line number and n2 is the last line number.
To Run the program type R(UN)<cr>
To exit the basic interpreter type Q(UIT)<cr>
Adding to the basic program is done by typing a line number followed by program text. If the line number already exits, it is replaced.
To see the size of the basic program, type S(IZE)<cr>. The result will be a decimal number.

The SA(VE) and the LO(AD) command as described in the MicroWorld Basic Manual may not work. There will be serial data at the FLAG output of the 2650 when the SAVE command is entered and when the LOAD command is entered, data is expected at the SENSE input of the 2650. This was never tested on the PHUNSY and the format is unknown.
Basic Programs can be stored using the MDCR. See Chapter 3.

The basic interpreter works only with 'unstructured basic'. The stored program consists of exactly the text that is typed. No special codes are used. The program text starts with the STX and it ends with the ETX ASCII control character.

See the 'MicroWorld Basic Manual' for more information.

Some locations in the interpreter that can be altered:

address	**function**
200A	start location of basic program
200C	channel 0 input routine (INPUT#0) (default)
200E	channel 0 input routine (PRINT#0) (default)
2010	channel 1 input routine (INPUT#1)
2012	channel 1 input routine (PRINT#1)
2014	channel 2 input routine (INPUT#2)
2016	channel 2 input routine (PRINT#2)
2018	channel 3 input routine (INPUT#3)
201A	channel 3 input routine (PRINT#3)
201C	check if key pressed
201E	serial i/o initialization if required, else 0
2020	basic-program-file input routine
2022	basic-program-file output routine
2024	FILE ERROR routine
2026	quit basic program and jump to monitor

8 Text Editor

The text editor (PHEDTR2 or PEDT-0A3) resides at address 0x4000 in Bank 0Q. After it is loaded and called (e.g. 0QQ), it requires buffer settings. The buffer starts at address 0x4000 but in bank Q1 and can be set to end in the last bank QF.
Default the text buffer setting is 16384 to 32767, which is exactly the range of Q-bank 1 (16384 characters).

Make sure that there is some text at 1Q 4000 or e.g. the character 0xff because the editor may act strange if there are more then one line of printable characters.

On the top of the screen all options are listed. Type the first Letter of the option and a return to activate them.
When editing, there are always 25 lines displayed and editing is always done at the middle line, line 13. Here several control functions are active. These are listed at the to of the screen.
The Arrow keys are operational:

- Up and down are used to move up and down in the text.
- Right and Left are used to insert and remove characters at the cursor location.
- <ctrl> U is used to move the cursor right.
- <ctrl> E is used to go to the beginning of the line.
- <ctrl> F is used to skip words.
- <ctrl> C is used to return to the command mode.
- <ctrl> H acts the same as the back-space key.
- The Escape key returns to the command mode.

After a text is ready it should be saved to an MDCR cassette by means of the commands from the monitor. e.g.:
TSAVE TEXT,4000-7FFF,2L,1Q
2L means that it is about ascii text.
1Q is the bank address.
If a text is larger then one bank it must be saved in more steps. e.g.:
TSAVE TEXT,4000-7FFF,2L,1Q,1OF2
TSAVE TEXT,4000-5FFF,2L,2Q,2OF2
When these files have to be restored next time, type
TLOAD TEXT and both sections will be reloaded.

Note that I wrote this program myself in hex, a long time ago, so it may not be bug free;-) It was one of the largest programs I wrote. However, can you imagine a text editor of just over 4KB these days? ;-)

9 Hello

Hello is a small program that lists some PHUNSY info to the screen. Type TRUN HELLO and you will see it.
However, two extra functions are present.
Start Logging: 880G
End Logging: 890G
Start logging means that everything printed to the screen is also printed to the log-file, if a log-file is entered as parameter at the phunsy command line. Logging stops when 890G is executed.
The Hello program is on every cassette but it is also present when the PHUNSY Emulator is started.

10 MODEST

MODEST is a resident line assembler that has very neat features.

11 Forth

There is a Forth program for the 2650. I don't know who wrote this implementation but I changed the I/O for the Phunsy. It seems somewhat buggy though but some functions work. Maybe it was not yet finished. The program name is FORTH and it resides at address location 2000-3FFF. After loading, type 2000G. Type BYE to return to the monitor.