HitMon11 v0.24 Manual

COLLABORATORS			
	TITLE : HitMon11 v0.24 Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 11, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

HitMon11 v0.24 Manual

1.1 Main Screen

```
                          HitMon11 v0.24 Manual.

$VER: HitMon11_Manual_Guide 0.24 (15-Feb-94)

See  Contents
```

1.2 1.0 Copyright notice

```
1.0 Copyright notice

This  program is  freeware. There should  be no charge  when copying
this  program. You are not allowed to make any money by distributing
this  program.  You  are allowed  to  take  a nominal  fee to  cover
copying  and disk  costs but  nothing more.  If you  distribute this
program  see to that all files  (listed in the file "OnTheDisk.txt")
are kept together. You are not allowed to change any of the files.
```

1.3 1.1 Disclamer

```
1.1 Disclamer

I  believe this program workes as stated in this manual, but I don't
guarantee  anything. Neither  do I  take any responsibility  for any
damage made directly or indierctly by this program.
```

1.4 1.2 Author

```
1.2 Author

All code by: Richard Karlsson
```

If you want to send in a bug report, have any questions or for some other reason want to contact me, do so at:

During September-June:

Mail: Richard Karlsson
c/o Kleivan
Palmaersgata 28
S-582 49 Linkoping
SWEDEN

Phone: +46 13 173 423

E-mail: d93ricka@und.ida.liu.se

During June-September:

Mail: Richard Karlsson
Klovergrand 4
SF-22100 Mariehamn
FINLAND

Phone: +358 28 22 441

E-mail: No certain e-mail. You can always try my "winter" e-mail address. But most certainly you will have to send snail mail.

1.5 1.3 Copyright & Trademarks

1.3 Copyrights & Trademarks

1.6 1.4 Special thanks

1.4 Special thanks

Special thanks to Fredrick Karlsson for writing the expression evaluator for the assembler included in this package, to Motorola for their great processors, and to Commodore for the greatest Computer, the Amiga.

1.7 2.0 Contents

1.8 3.0 General information

3.0 General information

The HitMon11 is a monitor intended to be used to upload Motorola S-Records to the Hc11 line of processors from Motorola. And also for faciliating debugging of programs. It should run on any Commodore Amiga computer. I believe it's the most powerful Hc11 monitor of it's kind that can be run on an Amiga.

1.9 3.1 Known Bugs and Weeknesses

3.1 Known Bugs and Weeknesses

This program is still in it's early stage and many functions is still to be added to the program, I hope. The program is based on a S-Record uploader I wrote a long time ago in pure hardware code, and that version was not intended to reach the public and thus was not very user friendly (i.e. if you did anything wrong the program would crash). Some of these weeknesses are still in the program so I would advice you to restart the program (Write restart at the prompt) if anything goes wrong. If for example you loose contact with the Hc11 the program has no possability to retain contact. You'll have to Restart. As soon as you see an error message regarding the contact with the Hc11 no more contact will be possable after that point. But some commands will still give you output. For example if you MD (Memory Dump) after contact has been lost, you will first get an error message but then MD will still output a Memory Dump. The contents of this memory dump is most certainly wrong. Restart HitMon11 to start over.

- If you DefM (Define Macro) a macro that already exists no error will be reported. You will then have two macros with the same name, but only the first one defined will ever be called. Advice: DelM

the first macro before redefining it.

- The DOS command executed with the "DOS" command can't ask for inputs from the keyboard. I.e. 'Dir Df1:' works fine, but 'Dir ?' doesn't work. But it will return control to the monitor. So no tilts or hangs, as far as I can see.

- No other known bugs...

Features I would like to add to the program in the future:
- Step by step debugging.
- Breakpoints.
- Symbols.
- The possability to import a symboltable from the assembler.
- Better error control.
- I'm not very satisfied with the way the program handles numbers overall, it considers every value greater than 255 and less than -256 as 16 bit and the rest of the numbers as 8 bit. This is a weekness that will be removed.
- Other numeric formats than HEX will be allowed.
- Expressions will be allowed.

1.10 3.2 DOS Command line format and options

3.2 DOS Command line format and options

Usage:
HitMon11 <Talker> <Crystal> [-m <Macro file>] [-d <Device>] [-u <Device Unit>]

<Talker>	The filepath to the Talker file to be used.
<Crystal>	The processors crystal frequency, in kilo herz.
<Macro file>	The file path to a macro file to load at startup. If no macro file is specified none is loaded. If this macrofile contains a macro called AutoStart, then that macro will be ran as soon as the Talker has been uploaded to the Hc11.
<Device>	The device to use when recieving/sending data from/to the Hc11. If this is not specified it defults to "serial.device"
<Device unit>	Which unit number of the device to use. If this is not specified it defults to 0.

The first thing the program will do is load the talker and macro files. Then it will try to make contact with the Hc11, and upload the talker. The talker is a little but powerful program written by Motorola that resides inside the Hc11 RAM. There are different talkers written for all Hc11 models. Remember that the talker is in RAM and thus you can't change the RAM during the operation of the monitor. This shouldn't be a problem since the RAM will be cleared anyway when the Hc11 looses it's power and thus you will never have any data there that is needed when your program starts. If it is not possible to upload the talker to the Hc11 HitMon11 will tell you to reset you Hc11 in BootStrap mode (ModA = 0, ModB = 0). When the processor is reset the program starts uploading the talker

again. If it fails to do so you will be instructed to reset your Hc11 again. This is repeated until it works without any problems. Sometimes you will have to reset your processor many times and only after a while does it work. I don't know why this is the case and I will try to fix the problem till the next release. Sometimes it helps to adjust the crystal frequency given to HitMon11. I don't think there is any idea to change it by more than plus or minus 200 kilohertz. If the program tells you over and over again to reset your Hc11, but the upload fails every time, you probably have set the wrong crystal frequency, or picked the wrong talker file. If nothing happens when you reset you Hc11, you have maybe reset it in the wrong mode (Should be BootStrap), or else the connection between the Amiga and the Hc11 is faulty. Remember that you have to convert the voltages from the Amiga to the Hc11 to TTL levels and the voltages from the Hc11 to the Amiga to RS232 levels. When the talker has uploaded properly the processor state is written to the screen. Also if a macro file was specified at the command line and that macro file contains a macro called AutoStart, then that macro is run.

1.11 3.3 Internal Command Line Format And Options

3.3 Internal Command Line Format And Options.

When the monitor is ready to accept commands from you if writes a prompt to the left of the screen. It looks like this:

>

At the prompt you can write any of the supported commands. No white spaces are allowed infront of the command (not including the one after the '>').
If the command takes any options they should be separated by spaces, any number of spaces works fine. All commands accepted are listed and explained in chapter 4.

1.12 4.0 Command Summary

4.0 Command Summary

This is a list of all supported commands right now:

```
BF Addr1 [Addr2] Byte|Word       Set   MCU  Memory   byte(s)  (Block
CALL Addr                        Calls a soubroutine in MCU memory.
CLRM                             Clear Macros
DASM [Addr1] [Addr2]             Disassemble MCU memory.
DEFM MacroName                   Defines a macro
DELM MacroName          - Deletes a macro.
DOS Command [Arg1] [Arg2] [...]        Executes a DOS command.
DR                               Display Registers
EEPROM [Addr1] [Addr2]           Define EEPROM range
EEPROM CLR                       Clear all EEPROM ranges.
```

```
EEPROM BULK Addr                  Bulk erase EEPROM
EEPROM ROW Addr                   Row erase EEPROM space.
EEPROM ERASE [Enable|Disable]             EEPROM erase  before write
                                          enable or disable.
FIND <Addr1> <Addr2> <8|16|32>    Finds  a sequence  of bytes  in MCU
G Addr                            Starts a MCU program.
LOADM <MacroFile>                 Load Macros
LOADS <File>                      Load motorola S19-record
LSTM                              List Macros.
MD [Addr1] [Addr2]                Memory Dump
MM Addr                           Memory Modify
MS Addr Value                     Memory Set
QUIT                              Ends the HitMon11 program.
RESTART                           Restarts HitMon11.
SAVEM FileName                    Saves macros to file.
VERF <S-Record>                   Verifys MCU content to a S-Record
```

1.13 Command BF (Block Fill)

```
BF Addr1 [Addr2] Byte|Word        Set   MCU  Memory   byte(s)  (Block
                                  Fill).
```

This command programmes one or more MCU memory locations, staring at Adr1 ending at one location less than Adr2. If no Adr2 is specified only one byte|word is written.

-- **** --

1.14 Command CALL

```
CALL Addr                         Calls a soubroutine in MCU memory.
```

This command calls a soubroutine in the MCUs memory at address <Addr>. Control is turned back to the monitor as soon as the subroutine has ended.

Example:

```
Call $ee00                        Calls  the  subroutine  at  address
                                  $ee00
```

-- **** --

1.15 Command CLRM (Call Macro)

```
CLRM                              Clear Macros
```

Clears all macro definitions. Macros not saved will be lost.

Example:

```
Clrm                              All macros cleard.
```

-- **** --

1.16 Command DASM (Disassemble)

```
DASM [Addr1] [Addr2]            Disassemble MCU memory.
```

Disassembles MCU memory starting from address <Addr1> ending at <Addr2> or as soon as the opcode at <Addr2> ends. If Addr2 is not specified HEX 100 (256) bytes of memory will be assembled, starting from <Addr1>. If no address is specified disassembly will start at the current address, and contine for 256 more bytes. The current address will be updated, so that disassembly can be contined by writing only 'DAsm'.

Example:

```
Dasm $ee00 $f000                Will disassemble MCU memory from
                                address HEX ee00 to address HEX
                                $f000.
```

-- **** --

1.17 Command DEFM (Define Macro)

```
DEFM MacroName                  Defines a macro
```

Defines a macro with the name <MacroName>. You can now write the macro contents separated by returns (Enter). When you're finished press return twize.

If you want your macro to contain commandline arguments specify them as \n. Where n is the number of the argument starting from 0. When the macro is run these will be replaced by the command line arguments, or with nothing if there are no arguments.

Example:

```
DefM LD <Enter>                 Defines the macro LD
EEPRom Bulk \0 <Enter>
LoadS S19:Current.S19 <Enter>
EEPRom Erase Enable <Enter>
<Enter>
```

Here you're returned to the prompt. When the macro is run \0 will be replaced by the first argument.

-- **** --

1.18 Command DELM (Delete Macro)

```
DELM MacroName          - Deletes a macro.

The  macro with the name <MacroName> will be deleted from memory. If
the macro has been saved to disk it will be safe there.

Example:
DelM AutoStart                  The macro called  AutoStart will be
                                deleted.

                               -- **** --
```

1.19 Command DOS

```
DOS Command [Arg1] [Arg2] [...]        Executes a DOS command.

Executes  the DOS  command <Command>  with the  specified arguments.
Control  is returned  to the monitor  as soon as the  DOS command is
finished. No Ctrl-C's can be sent to the DOS command.

Example:
DOS Dir Sou:                    Dirs the volume or assign Sou:. But
                                only if dir is found in the current
                                path and  if that  is what  the dir
                                command found does...

                               -- **** --
```

1.20 Command DR (Display Registers)

```
DR                              Display Registers

Display  the current  values in the MCU  register, stackpointer, CCR
and  PC. The CCR is displayed both in HEX  and BIN form. DR takes no
aruments.

Example:
DR                              Displays the current MCU registers.

                               -- **** --
```

1.21 Command EEPROM

```
EEPROM [Addr1] [Addr2]          Define EEPROM range

Defines  the address  space between  <Addr1> and  <Addr2> (Excluding
<Addr2>)  as EEPROM  space. If  only <Addr1>  is specified  only the
byte at <Addr1> is defined as EEPROM space.

Anytime  a value  is written to  an address  that is included  in an
```

EEPROM range a special EEPROM algorithm will be used to program that address. This is completely transparent to the user.

If no address is specified a list will be displayed with all the current EEPROM ranges.

Examples:

EEPROM $103f	Defines address HEX 103f as EEPROM space.
EEPROM $b600 $b800	Defines addresses between $b600 and $b800 as EEPROM memoory. (Including $b600, excluding $b800)
EEPROM	Displays all current EEPROM ranges.

-- **** --

1.22 Command EEPROM CLR

EEPROM CLR	Clear all EEPROM ranges.

Any EEPROM ranges specified by the EEPROM command are turned back to normal memory. The actual memory of the MCU isn't changed only the list that specifies what memory that should be treated as EEPROM is cleard.

-- **** --

1.23 Command EEPROM BULK

EEPROM BULK Addr	Bulk erase EEPROM

Bulk erases the EEPROM area in which <Addr> is included. The whole area will be erased. An erased EEPROM byte reads HEX FF.

See the manual to your Hc11 to see how the EEPROM areas are defined in your Hc11 and how they can be moved around.

Example:

EEPROM Bluk $b600	Clears the EEPROM area at $b600.

-- **** --

1.24 Command EEPROM ROW

EEPROM ROW Addr	Row erase EEPROM space.

Row erases the EEPROM Row in which <Addr> is included. All bytes in

the row will be erased. An erased EEPROM byte reads HEX FF.

See your Hc11 manual how the ROWs are defined in your Hc11, and how they can be moved.

Example:

EEPROM Row $b600	Erases the EEPROM Row in which address HEX $b600 is included.

-- **** --

1.25 Command EEPROM ERASE

EEPROM ERASE [Enable|Disable]	EEPROM erase before write enable or disable.

Enables or disables the 'EEPROM erase before write' feature. If it's enabled any EEPROM byte that is written to will be byte erased before if is written to. If it's disabled it will not be erased before it's written to.

If this feature is enabled programming time will take about twize as long. And if it's not enabled and the EEPROM byte to be programmed is not previously erased the write may fail. Good practice is to normally have this feature enabled, but if you are going to program a large amount of EEPROM addresses and the area in which theese are contained can be erased without loss of data, then you should BULK erase the whole area, disable the EEPROM erase before write feature, then program the desired addresses, possably by loading a S-Record, when this is finished enable the EEPROM erase before write feature again.

Actually its only the first character after ERASE that is checked for. So, E for enable and D for disable.

If neither Enable or Disable are specified the current state of the EEPROM erase before write feature will be listed.

Examples:

EEPROM Erase	Lists the current state of the EEPROM erase before write feature.
EEPROM Erase Enable	Enables the EEPROM erase before write feature.
EEPROM Erase Disable	Disables the EEPROM erase before write feature.

-- **** --

1.26 Command FIND

```
FIND <Addr1> <Addr2> <8|16|32>   Finds  a sequence  of bytes  in MCU
                                 memory.
```

Searches from <Addr1> to <Addr2> for the occurence of the byte sequence in the last operand. If is't a byte (8 bit), that byte will be searched for. If it's a 16 bit word, that 16 bit word will be serached for. And if it's a 32 bit word, that 32 bit word will be searched for. Even if a 16 or 32 bit word is specified the search will be performed at every byte boundery.

```
Examples:
FIND $8000 $9000 $12             Searches for  the occurence  of HEX
                                 12 from address $8000 to $8fff

FIND $8000 $9000 $123            Searched for  the occurence  of HEX
                                 0123 in  the same address  space as
                                 above.

FIND $8000 $9000 $12345          Searched for  the occurence  of HEX
                                 00012345 in the  same address space
                                 as above.
```

-- **** --

1.27 Command G (Go)

```
G Addr                           Starts a MCU program.
```

Starts MCU program at address <Addr>. Control will not return to the monitor.

```
Example:
G $ee00                          Starts  code  execution at  address
                                 HEX ee00.
```

-- **** --

1.28 Command LOADM (Load Macros)

```
LOADM <MacroFile>                Load Macros
```

Loads macros from the file <MacroFile> and adds them to the current macros in memory. This file may either be produced by the SaveM command or edited by a standard text editor as for example Ed on the Workbench disk. The file should look like this:

```
DefM <MacroName>
Begin
<Command1> [Arg1] [Arg2] [...]
[<Command2>] [Arg...]
[...]
```

```
End
DefM <Macro2>
Begin
[<Command...>]
[...]
End
```

... I hope you get the idea. '\n' will as usual with macros be replaced by the command line arguments (n is the number of the argument starting from 0).

-- **** --

1.29 Command LOADS (Load S19-Record)

```
LOADS <File>                    Load motorola S19-record
```

Loads the file <File>, checks if it's a valid Motorola S19 record and if it is uploads it to the Hc11. While uploading the remaining bytes to upload are displayed on screen. The value is updated every 256 bytes or less. If data is uploaded to EEPROM, 256 bytes takes quite long so don't get impatient. If you have an LED on the TxD and/or on the RxD line(s) you can see that something indeed is happening, else you'll just have to wait for it to finish.

-- **** --

1.30 Command LSTM (List Macros)

```
LSTM                            List Macros.
```

Lists the macros currently in memory. Here you can see which macros are currently available.

Examples:

```
LstM                            Lists current macros.
```

-- **** --

1.31 Command MD (Memory Dump)

```
MD [Addr1] [Addr2]              Memory Dump
```

MD displays memory using the standard HEX dump format. I.e at the very left end of the line the address is displayed in HEX. Following the address is the current content (byte) of that address, followed by the contents of the following 15 addresses (totaling 16) separated by spaces. After the HEX values the same 16 bytes are listed again but in ASCII format enclosed by quotes. An example shows how it all can look:

```
$8000 20 43 6f 70 79 72 69 67 68 74 2c 20 28 63 29 31 "Copyright,
(c)1"
...
(Normally  it will be on one line. But this document is not 80 chars
wide, so it didn't fit.)
```

If only <Addr1> is specified the dump starts at <Addr1> and displays 256 bytes forward from that address. If <Addr2> is also specified the dump is displayed from <Addr1> to <Addr2>, excluding <Addr2>.

Examples:

Md $8000	Dumps memory between $8000 and $8100 (Excluding $8100)
Md $8000 $8300	Dumps memory between $8000 and $8300 (Excluding $8300)

-- **** --

1.32 Command MM (Memory Modify)

MM Addr — Memory Modify

Modifies the memory starting at address <Addr>. The address to be modified is displayed and it's current value. You can now write the value you would like this address to be changed to or press return not to change it and quit back to the monitor command line. After the address has been changed you'll be presented with the next address in the same way. Press return without any value to return to the monitor commandline. If the value entered can't be decoded or if the memory address doesn't change to the value you stated the same address will be showed again.

Example:

MM $b600	Start modifying memory from address HEX b600.

-- **** --

1.33 Command MS (Memory Set)

MS Addr Value — Memory Set

Sets the memory position <Addr> to the byte value of <Value>.

Example:

Ms $8000 $12	Sets the address HEX 8000 to the value HEX 12.

-- **** --

1.34 Command QUIT

```
QUIT                            Ends the HitMon11 program.
```

Quit will quit the program without confirmation. It wont check if you have saved changed macros. If anything is written but the quit command the program will not quit. This ensures you will not quit the program by mistake.

```
Usage:
Quit                            Quits the program.
```

-- **** --

1.35 Command RESTART

```
RESTART                         Restarts HitMon11.
```

If contact has been lost with the MCU, the MCU has to be reset (in bootstrap mode) and the HitMon11 will have to be restarted. The Restart command acts exactly like quiting the program and restarting it with the same arguments.

```
Example:
Restart                         Restart HitMon11, and wait for new
                                contact with the MCU.
```

-- **** --

1.36 Command SAVEM (Save Macros)

```
SAVEM FileName                  Saves macros to file.
```

Saves macros currently in memory to the file specified by <FileName>. To see what macros are beeing saved use the 'LstM' command. Macros that have been changed must be saved before the program is exited, or they will be lost.

```
Example:
SaveM S:Setup.Mcr               Saves macros to file 'S:Setup.Mcr'.
```

-- **** --

1.37 Command VERF (Verify)

```
VERF <S-Record>                  Verifys MCU content to a S-Record

Loads  the S-Record  <S-Record> (May  include path) and  verifys MCR
memory  content  to the  information  in the  S-Record. If  there is
differences  they are reported to the screen. Addresses not found in
the S-Record are not verified.

Example:
VERF T:Current.S19               Checks if MCU memory is the same as
                                 the S-Record says it should be.

                              -- **** --
```