Appendix
DPU Diagnostic Programs
Jirka Hoppe

Version 9. December 1980

The 'hardware test program' is a package of M6800 programs providing a collection of test programs which will be used for the initial set up and the hardware maintenance of the personal computer Lilith.

The programs are organized in a tree of submonitors, each devoted to test either one PC board or a part of it. Each submonitor consists of a number of routines testing one special function of the board. Some of these routines provide an automatic test that checks a function and reports errors, others only prepare a situation that will be checked manually using a scope, logic state analyzer, etc.

Currently the following submonitors are implemented:

main hardware test submonitor
Micro controller
ALU
port controller and main memory
IFU
devices
micro memory
speed tests

All submonitors share a common error reporting routine called by the SWI instruction in case of an error. This routine prints a string 'ERROR IN ', gives a position of the error in the program and prints the value of the M6800 registers B, A, X, PC. The position of an error in the submonitor tree is determined by two strings: the first gives a name of the current submonitor (stored in a variable MAIN.ID), the second gives the name of the current test routine (stored in a variable SUB.ID).

In most error cases the B, A, and X registers indicate the expected and the actual value of a test variable. The exact meaning of the register values is described in each test. The PC register determines the position of the SWI instruction in the M6800 program.

For a frequent comparison of the X register and the CPU bus a PRINTBUS routine is provided. This routine works similar to the SWI routine except that at the beginnig the BA register pair is loaded with the

CPU bus.

Some tests print some additional information. For the exact meaning refer to each test description.

Messages printed by the system can be controlled by the TYPE command. It is possible to suppress error-printing or the end messages, to stop in case of an error or to send a signal in case of an error.

The test routines are called either by typing a key letter from a menu, or by executing a test chain. Each submonitor contains a local chain, the main hardware monitor contains a global chain combining the local chains of all submonitors.

COMMANDS OF THE MAIN MONITOR USED IN THE HARDWARE TES

```
Y -TYPE    This command controls the output of the test program.
           The control is set by typing 'nY' where 'n' has following meaning:

             0 - neither error nor end messages are typed.
                 This is used in cases  where an error situation is monitored on
                 the scope and the delays caused by the error messages should be

             1 - type error and end messages - the normal situation

             2 - type error messages only, don't type end messages.
                 This may be used in a long running chain test, where only error
                 messages are of interest.

             3 - like '1' but stop in case of an error.
                 This may be used to stop the execution of the program and then
                 to check the signals on the board using a logic probe. To conti
                 the program, strike any key.

             4 - like '1' but it sends a pulse  in case of an error.
                 This pulse may be used to trigger a scope or to stop a logic st
                 analyzer.  (The pulse is the bit 01 of the control register on
                 M6800 board - unfortunately, not tested yet - sorry ...)
```

G GO

Format: startaddress 'G' stopaddress

The execution of the program is started at address 'startaddress' and a breakpoint is inserted at 'stopaddress'. When the program reaches the breakpoint the execution is aborted and the contents of the B, A, X and PC registers are displayed on the screen.

When the 'startaddress' is ommited, the last breakpoint address is taken as a startaddress. When the 'stopaddress' is ommited, no breakpoint is inserted.

CAUTION: the startaddress and the stopaddress must be different. It is impossible to execute a loop n-times by omitting the startaddress and putting the breakpoint always on the same address.

The 'GO' command will be used to stop the execution of a test program at a suitable place, allowing a static check of all signals with a logic probe.

TROUBLE SHOOTING PROCEDURE

- Measure all voltages on the bus and on each board using a scope (to find noise or spikes) and a digital voltmeter (for accuracy).

- check the clock wave form using the CLOCK command from the main HW submonitor and a scope.

- run either a global or local chain program

- if a test gives too many error messages, abandon it and try another test to isolate the errors

- if you already suspect where the error might be, either use the 'DO BY HAND' co or stop the execution of a test programm with the stop option of a GO command to track the signals on the board.

MAIN HARDWARE SUBMONITOR

X EXECUTE CHAIN The global test chain combining all submonitor chains is executed.

L LOOP ROUTINE Format: startadr 'L'

A procedure starting at address 'startadr' is executed in a loop. This will be used to monitor signals on the scope or to loop execution of a single routine if some reliability problems are expected.

T TEST BUS A 'waltzing zero's and one's' test on the bus is performed. This test detects short connection on the bus.

ERRORmessage: SUB.ID "CPU BUS"
(B,A)=actual bus content; X=expected value

Possible failures: Be sure that the microcontroller board is inserted and the bus destination logic is OK.
Check that the panel's CPU bus chips (74LS374) are enabled.
Pull out one board after another to find which board is causing trouble. Check if all output buffers to the bus are disabled.

N PANEL SELF TEST A self test of the M6800 board for a proper setting of MIR and of M6800 board and immediately verified.

ERROR messages

SUB.ID "MIR.TST" test the setting of MIR
B=actual value; A=expected; X=address of the MIR register

MIR4 = 3C08 ...... MIR0 = 3C0C

SUB.ID "UAR.TST" test the micro address
(B,A)=actual value; X=expected

Possible failures: Pull out all boards and check if the selftest is working. There may be some short connections caused by other boards. Check the proper selection of 74LS374 and 74LS244.

O CLOCK The function of the CPU clock is controlled by typing a 'nO' command. 'n' has following meaning:

0 - stop the CPU clock

1 - Motorola sends repeatedly a 'single step'

2 - the CPU clock is set into the 'run with full speed' mode

Striking any key on the terminal causes the program to stop the CPU clock and to return back to the HW monitor.

Possible failures: If the CPU clock is not running, check if the 74s124 oscillator is working and the following inputs are HIGH: CPU DIS L, RDY H, MIR14. Depending on the value of 'n' SS CLK L must either give low going pulses or RUN H must be HIGH.

H DO BY HAND This command allows a manual setting of a microinstruction, of the CPU bus and a manual control of the CPU clock. The 'H' command provides following subcommands:

I - set a microinstruction.
The program expects typing of 5 bytes which are interpreted as MIR4, MIR3,...,MIR0

B - set the CPU bus - a word is expected.
Be sure that the microinstruction contains BUSsource=panel (D).

G - set the CPU clock into the running mode.

H - stop the CPU clock

cr- exit from the procedure

any other key causes one single step to be executed.
After each command the current content of the CPU bus
and the current micro address are displayed.

G GO Format: startadr 'G'
The microprogram starting at 'startadr' is executed.

U uCTRL Program control is transfered to the micro controller submonitor.

P CPU Program control is transfered to the CPU submonitor.

S STORE AND PORT Program control is transfered to the main memory and MDP port controller submonitor. On the IFU tape some commands of the HW submonitor are omitted and some new are introduced:

U IFU Program control is transfered to the IFU submonitor.

M UMEM Program control is transfered to the micro memory submonitor.
The DEVICES submonitor is accessible from the main monitor ('V')

MICRO CONTROLLER SUBMONITOR

To execute this submonitor the following boards are necessary: monitor, microcontroller, and ALU.

To check the function of the condition jump switch (U37) run the 'N CONDITION' procedure of the ALU monitor.

To check the microROMs run the 'U uROM COMPARE' procedure in the micromemory submonitor.

Most tests of this submonitor will give error messages if the CC (condition code) pin is left open. This pin should be held LOW.

If the ALU board is inserted the CC pin will be LOW.

Commands

J JUMPS

This procedure verifies the jump operations of the 2911. A jump every possible address is executed.

```
Error Message:  SUB.ID   "JUMP OP"
                (B,A)=actual adr;  X=expected
```

Possible failures: The input of condition code CC H must be low (it will be high if ALU is not inserted!). Check that the control vector of the 2911 is correct S0=S1=FE=PUP=high and that the address from the microinstructio is set up on the 2911. Check if MCLK is arrives at the 2911 and the output of the 2911 is enabled.

P PUSH.POP

This procedure verifies the jump subroutine and return mechanis of the 2911.

```
Error Message:   SUB.ID    "PSH/POP"
                 (B,A)=actual adr; X=expected
```

Possible failures: see JUMP test.
Control vector for JSR is S0=S1=PUP=high; FE=low;
Control vector for RTN is S1=high; S0=FE=PUP=low

T DEPTH OF STACK

This procedure verifies the depth of the 2911 stack.

```
Error Messages:  SUB.ID   " DEPTH"
                 (B,A)=actual adr;   X=expected
```

Possible failures: if JUMPS and PSH/POP are OK then a 2911 may

I INC OF PC	This procedures tests the CONTINUE function of the 2911. Error Message: SUB.ID "PC-INC" (B,A)=actual adr; X=expected Possible failures: see JUMPS The vector for CONTINUE is S0=S1=PUP=low, FE=high.
K CONST	This procedure checks that MIR0..MIR7 are propagated as a const to the CPU bus if MIR12 is high. Error Message: SUB.ID "CNST.TEST" (B,A)=actual; X=expected Possible failures: Check the function of the 74S244 drivers U34
S SRC/DST	Format: n 'S' { ',' / '/' } A bus source and destination field in the MIR is specified by 'n'. Typing ',' increments the souce field by one, typing '/' increments the destination field by 10h. The proper decoding of src/dst must be checked by a logic probe Possible failures: Check function of the 74S138 decoders U32,33
R INTERRUPT	This test shows the interrupt entry micro addresses for various interrupt requests and various masks. Additionally the correct setting of the interrupt mask is checked. To execute this proce ground one or more interrupt lines and run the program. The pro displays the entry addresses for all masks. The correct values

Request	Masks	uADR
none	00 to FF	value greater than 010
0	00 FE	00F

1	00	FC	00E	
2	00	F8	00D	
3	00	F0	00C	for all other masks is uADR>=010
4	00	E0	00B	
5	00	C0	00A	
6	00	80	009	
7	00	00	008	

Print out: uADR   given mask   reread of mask

If the interrupt is not active for the given mask, the uADR poi
either to 0FFF (if the MAP PROM is not inserted), or to a rando
label of a macro instruction (if the MAP PROM is inserted).

Possible failures: Check that INT DIS L on the U24
is HIGH. Check that MAP ENABL L stays H and VECTOR ENABLE L goe
Check that the request is properly masked (U35, U45, U46).
Check outputs of 74LS348 U47.

There may be some interrupt lines active (clock, disk etc.).
To disable the clock interrupt pull out the CPU-port board,
to disable the disk interrupt pull out the disk board.

M MAP PROM

The content of the map PROM is compared with the master tape.
Insert the tape into the left drive and run the program.
The program reports all differences between tape and PROM.

Errormessage: SUB.ID   "MAP"
(B,A)= actual uADR; X=expected

Possible failures: If all addresses are wrong check
the MAP ENBL L signal - it should have low going pulses.
If only few addresses are wrong - check the PROM with
the ROM burner.

X EXEC CHAIN The micro controller test chain is executed.
It contains the following tests: JUMPS, PUSH.POP, STACK.DEPTH, INC.PC, CONST.TEST

ALU Submonitor

For this submonitor the following boards are necessary: DPU, MCU, ALU.

0 OUTP

This procedure verifies the output lines of 2901's. The program first executes a CLEAR R0 instruction and then increments R0 each step by one.

Error Messages: SUB.ID "OUT1" result of CLEAR R0
B,A)=actual output; X=expected

SUB.ID "OUT2" result of increment
(B,A)=actual output; X=expected

Possible failures: This is the first test of the ALU, therefore anything might be wrong. Stop execution of the program after an error or use the DO BY HAND command to check if all MIR bits are properly set on 2901. Check if output buffers 74S244 U23 and U32 are enabled. Test the CPU bus for short connections (test CPU.BUS in the main HW submonitor).

I INP

This procedure verifies the input lines from the CPU bus to the 2901. It puts a bit combination on the CPU bus, loads it into R0 and reads it again.

Error Messages: SUB.ID "INP"
(B,A)=actual; X=expected

Possible failures: Be sure the OUTP test is OK.

Are the 25S10 shifters U33- U36 and U15 - U18 enabled?

Is the stack output buffer U13, U14 disabled?

Is MIR19 low? Are there any short connections on the lines between the first and second level of shifters?

If only few bits are wrong, try the shifter test.

H SHIFTER

This procedure verifies the 25S10 shifter. Various values are put onto CPU bus and all 16 shift possibilities are checked.

```
Error Messages:   SUB.ID   "SHIFTER"    shift count
                  (B,A)=actual;  X=expected
```

Possible failures: be sure the INP test is OK.
Check if the S0, S1, S2 and S3 signals correspond to the shift count given in the error message.

M MASK

This procedure verifies the function of the 25S10 shifter for the masking operation.

```
Error Messages:    SUB.ID  "MASK"      shift count
                   (B,A)=actual;  X=expected
```

Possible failures: see SHIFTER test.

Check that MIR19 is high. The OR's between shifters might be bad.

R REG ADR

This procedure verifies the addressing of the internal registers of the 2901. R0 is loaded with 0001, R1 with 0002,... R15 with 8000. After loading all values are checked.

```
              (B,A)=actual;  X=expected
```

Possible failures: Try to find out from the error messages which bit is wrong. Use DO IT BY HAND command to verify it.

U SOURCE

This procedure verifies the source field of the MIR. R0, R1, Q and the CPU bus are loaded with various values. The result of an OR function between various sources is compared with the table.

A = 2222; B = 1111; Q = 4444; bus = 8888

AQ	AB	ZQ	ZB	ZA	DA	DQ	DZ
6666,	3333,	4444,	1111,	2222,	AAAA,	CCCC,	8888

```
Error Messages:     SUB.ID  "SRC"     #source
                   (B,A)=actual;  X=expected
```

Possible failures: see REGADR test

P OPERATIONS

This procedure verifies the FCT field of the MIR.
R0 is loaded with 1111 and R1 is loaded with 2222.
The results of all operations are compared with the table

+	-+	-	OR	&	-&	XOR	-XOR
3333	1110	EEEE	3333	0000	2222	3333	CCCC

```
Error Message:     SUB.ID  "FCT"    function#
                  (B,A)=actual;  X=expected
```

Possible failures: see REGADR test
If there are problems with addition or subtraction only, use the CARRY test to check carry.

T SH.DEST

This procedure verifies the BQL and BQR destinations.
R0 and Q are loaded with various values and then the results are compared with a table (see listing). The program displays a pointer to the table to find which combination was tested.

Error Messages: SUB.ID "BQL" or "BQR" pointer to the table
(B,A)=actual; X=expected

Possible failures: Check MIR.DST on 2901.
Check the connection between 2901's.
Is the input to the RAM3 on most significant 2901
or the input to the Q0 on least significant 2901 OK?

L MULT

This procedure verifies the function of the multiply circuit.
For the detailed description of this circuit consult the paper "........".
The principle of this test is similar to the BQR/BQL test.

Error Messages: SUB.ID "MULT" pointer to the table
(B,A)=actual; X=expected

Possible failures: Be sure the BQR/BQL test is OK.
Is BUS.DST=MDST low? Is the Q0 input to the least significant 2901 enabled and controlling MIR.I1? (Q0=low -> MIR.I1=high;
Q0=high -> MIR.I1=low)

V DIV

This procedure verifies the function of the divide circuit.
For the detailed description consult the paper "......".
The R0, R1 and Q are loaded with various values
and then the two double steps of the divide sequence are executed.

Registers R0 and Q are compared with a table.
In case of an error the pointer to the table is displayed
to determine which situation was tested.

```
ERRORmessages:     SUB.ID  "DIV"     pointer to the table
                   (B,A)=actual;    X=expected
```

Possible failures: Be sure BQL/BQR and MULT tests are OK.
Run the CONDITION bit test to check the function of the
status carry bit.

Y CARRY

This procedure verifies the function of MIR.CARRY and of the
carry-circuit. Various values of R0 and R1 are added with various
carry control inputs. Consult the listing to determine which situation
was tested.

```
Error Messages:     SUB.ID  "CARRY.IN"
                   (B,A)=actual;  X=expected
```

Possible failures: Check MIR.CARRY on 2901.
Is 2902 (carry look ahead) OK?

S STACK

Format: n'S'

The expression stack is filled with various values and read back.
'n' is the start value which is rotated each run to the left.
If 'n' is not specified FFFE is assumed.

```
Error Messages:      SUB.ID   "STACK"     stack counter
                     (B,A)=actual; X=expected
```

Possible failures: Is the output buffer U13, U14 enabled?
Use the DO BY HAND command and check that the addresses of stack RAM are incremented by PUSH and decremented by POP.
Check the bus input to the RAM.

N CONDITION

The registers R0 and R1 are loaded with various values and then added. The next executed microinstruction is a conditional jump with various condition masks. The micro address is examined to see if the jump was executed. The results are compared with a table. To check the stack empty bit the same test is performed with push/pop stack instead of an addition.

If uADR points to 3 => the jump was not executed,
if uADR points to 10 => the jump was executed.

Error Messages: SUB.ID "COND" test with addition and result in the table
B=mask; A=uADR (jump/nojump); X=pointer to the table

SUB.ID "CONDS" test stack empty bit
(B,A)= no meaning; X=uADR

Possible failures: Halt the execution of the program and check if bits are correctly set on 74S174 U21
Check the CC H output.
Check the circuit on the micro controller board U19, U20 and U37 to see if the change jump/nojump works properly.

X EXEC CHAIN

The ALU test chain is executed.
The chain contains following tests:
OUTP, INP, SHIFTER(1), SHIFTER(FFFE), MASK, REG.ADR
SOURCE, OPERATIONS, SH.DEST, MULT, DIV, CARRY.IN,

```
    STACK(1), STACK(FFFE), CONDITION
```

MAIN MEMORY and PORT CONTROLLER SUBMONITOR

To execute this submonitor, the following boards are necessary: monitor, microcontroller, CPU-port controller and at least two memory boards.

The address parameters of all procedures may be specified in the range 0000 - FFFFh.

All tests are performed on one memory bank only.

To change the memory bank use the 'S' command.

The tests of this monitor should be combined with the microcoded tests (use the 'G' command of the hardware submonitor) and with the MODULA memory tests.

```
R READ LOOP

    Format:    adr 'R'

    This procedure reads the memory at the address 'adr' in a loop.
    This test is used to monitor all memory signals with the scope.
    Strike any key to interrupt the procedure.


W WRITE LOOP|

    Format:    adr 'W' val

    This procedure writes the value 'val' on the address 'adr'
    in a loop.
    Strike any key to interrupt the procedure.


Q RD.WR LOOP

    Format:    adr 'Q' val

    This procedure writes the value 'val' on the address 'adr'
    and reads it immediately.
    No comparison is performed.
```

Strike any key to interrupt the procedure.

P PATTERN

Format: value 'P'

One memory bank is filled with the pattern 'value' starting at the address 0000 to the address ffffH.
All memory locations are verified.

```
ERROR message:     SUB.ID  'MEM'  address
                  (B,A)=actual value; X=expected value
```

Possible failures:
Verify the form of CAS, RAS, R/W signals.
Is the board selected?
If there are only errors on specific addresses or on specific bits, try to find which row or which chip may be bad.

N RANDOM

Format: value 'N'

One memory bank is filled with 'random' numbers generated by a very simple random number generator:

```
mem[adr] := (value * adr) MOD 10000H
```

For value=1 each memory cell is filled with its own address, for any other odd value pseudo random numbers are generated. The quality of this generator is poor - but who cares?

```
ERROR messages:     SUB.ID  'MEM' address
                    (B,A)=actual value; X=expected value
```

Possible failures: see the pattern test

I INSPECT

Format: adr 'I' value [':' newvalue] [',' ' ']\

The content of the main memory at the address 'adr' is displayed.
The content may be changed by typing \| ':' newvalue |.\
Typing a comma couses the program to display the next address
and next word of the memory; typing a backslash causes
the program to display the previous address and its contents.
Any other character terminates the INSPECT command.

S SET BANK

Format: banknr 'S'

The bank for all following tests is set to 'banknr'.
Currently only bank 0 and 1 are implemented.
The default bank is 0.

X EXEC CHAIN

The main memory test chain is executed.
It contains the following tests:
PATTERN(0), RANDOM(1), PATTERN(FFFFh) and RANDOM(5531h).

IFU SUBMONITOR

To execute this submonitor the following boards are necessary: monitor, microcontroller, CPU-port controller, memories and IFU. Be sure that the memory test works before starting the IFU tests.

R TEST.F.REG

This procedure loads the F register of the IFU with all
possible combinations and then verifies them.

ERROR messages: SUB.ID 'LOAD.F.REG'
                 (B,A)=actual; X=expected

Possible failures: Check if U8 / U11 are enabled for write and U9 / U12 enabled for read. Check the CPU-bus with the T command of the hardware submonitor.

O TEST OFFSET

This procedure loads the offset register with all possible combinations and then verifies them.

ERROR message: SUB.ID 'LOAD OFFSET'
                 (B,A)=actual; X=expected

Possible failures: check if U16,17,18,19 are enabled for write and U7,10 are enabled for read.

N INC OFFSET

The offset register is loaded with zero and then incremented to FFFFh for following increment modes:
IR8+, IR8-, IR8* and IR4.

ERROR messages: SUB.ID	'INCPC.IR8+'	for IR8+
	'INCPC.IR8-'	for IR8-
	'INCPC.IR8*'	for IR8*
	'INCPC.IR4'	for IR4
	(B,A)=actual; X=expected	

Possible failures: check the INC PC H signal on U16.
Are all carry signals between U16, 17, 18, 19 ok?
Is the LOAD input of U16..19 high?

S SEQUENCING

All combinations of instructions are read into each
IFU instruction register from memory.

ERROR messages: SUB.ID 'IFUSEQ'
B=expected instruction byte; A=actual byte; X=offset

Possible failures: Check if the main meory is filled correctly.
For the first run, mem[0]=0; mem[1]=1; ... mem[ff]=ff.
All locations are incremented by one for the next run,
by two for the run after that, ... etc.
Is the IFU issuing memory requests?
Are there any clock pulses on U24..U32?
Is the select logic on U20 ok?

P PC ADDER

The F and OFFSET adder is verified for correct operation.
Each location in the main memory is filled with its own address.
The memory is read thru the IFU for various F and OFFSET values.
To speed up the test not all combinations are tested
(otherwise 2**32 tests are necessary!).

ERROR messages: SUB.ID 'ADDER'
(B,A)=actual; X=expected

Possible failures: check if the main memory is filled correctly.
Be sure all previous test are OK.
Are there any problems with the CARRY between U1..U4?

I INSPECT

Main memory inspect. See the main memory submonitor.

X EXEC CHAIN

The IFU test chain is executed.
It contains the following tests:
TEST.F.REG, TEST OFFSET, INC OFFSET, SEQUENCING, PC ADDER.

DEVICES SUBMONITOR

P PUT

Format: chan 'P' val
The value 'val' is written onto the I/O channel 'chan'.

G GET

Format: chan 'G'
The value read from the I/O channel 'chan' is displayed.

K KEYBOARD

This procedure displays the the keyboard data (hexadecimal) and status
register in a loop.
The LSB of the status register is set to 1 if there
are new data available, it is cleared each time
the data register is accessed.

To interrupt this procedure strike any key on the HP.

M MOUSE

The X, Y and key registers of the mouse are displayed
in a loop.
To interrupt this procedure strike any key on the HP.

U UART

Format: val 'U'

The character corresponding to the hexadecimal value 'val'
is transmitted from the UART.
Typing a character 'newch' on the terminal connected to the UART
changes the transmitted character to 'newch'.

Possible troubles:
  Check the frequency of the UART clock.
  Check the function of the status bits DRDY and TBRE on the UART.
  Check the function of the level shifters U48,49.

L LOOP DISK (not implemented yet)

Format:| val 'L'

This command is used to loop the read or the write function
of the disk.
The value 'val' is written into the sector register
of the disk controller
and then the procedure waits until the XFERcompleted
or a timeout bit is set.
The 'read' bit in the value has position 80h,

the 'write' bit has position 40h.
The sector may be between 0..2Fh.
All other information (track, data in the internal buffers, etc)
must be set before the procedure is executed.
To interrupt this procedure strike any key.

H head seek (not implemented yet)

Format: val 'H'

This command seeks the track 'val' on the disk in a loop.
CAUTION: This command does not make any changes on the track address
as the microcode does.

To interrupt this procedure strike any key.

S SET MEMORY

Format: val 'S'

The main memory is set to the pattern 'val' in the range
1000h to 7FC0h.
This command is used to test the display controller.

T TEST PICTURE

A test picture pattern is generated in the main memory
and the BMD is set.
adr(bmd)=FFFCh;
adr(bitmap)=1000h; w=30h; h=253h; position=0

I INSPECT|

Main memory inspect. See the main memory submonitor.

MICRO MEMORY SUBMONITOR

To execute this submonitor the following boards are necessary: monitor, microcontroller and micromemory.

R READLOOP

```
Format:   adr 'R' subadr
The byte 'subadr' on the address 'adr' is read from
the micro memory in a loop.
To interrupt this procedure strike any key.
```

W WRITELOOP

```
Format:   adr 'W' subadr '>' val
The value 'val' is written in the micromemory
on the address 'adr' and byte 'subadr' in a loop.
To interrupt this procedure strike any key.
```

P PATTERN

```
Format:   val 'P'
The whole micro memory is filled with the pattern 'val'
and then verified.

ERROR messages:  SUB.ID  'PATTERN' expected value,
                 B=subadr; A=actual value; X=address

Possible failures:
```

V ADDRESSING

Format: val 'V'

All five bytes at the address 'adr' are filled with the
\value = (adr*val) MOD 256\ and verified.
For val=1 every byte is filled with its own address,
for any other odd value pseudo random number is generated.
(A very poor random number generator).

ERROR messages: SUB.ID   'ADR' expected value
                  B=subadr; A=actual value; X=address

Possible failures:

I INSPECT

Format: adr 'I' subadr [':' newvalue] [',']

The content of the micromemory at 'adr' and 'subadr' is displayed.
The content may be changed by typing \':' newvalue\.
Typing a comma causes the procedure to display the next byte of the
micromemory.
Typing any other character terminates the INSPECT command.

U uROM COMPARE

The content of the micromemory is compared with the master tape. The tape mu
contain five files (one for each byte) coded in a standard M6800 format. The
first file corresponds to the byte 0, the fifth file to the byte 4. Unfortun
the program displays only the first difference encoutered in a code line (see
format description). There is a possibility that there will be some other
unreported differences on the adjacent addresses in the same line.

If for any reasons not all files but only the n'th file of five should be com
(the Apple user may need it), you can patch the file number in the M6800 prog
Consult your listing, find the marked instruction (behind the label REREAD) a
patch the LDAB =0 to LDAB =n.

ERROR messages: SUB.ID 'UROM CMP', ROM value
B=byte adr; A=tape value; X=address

Possible failures:

X EXECUTE CHAIN

The micro memory chain is executed.
It consists of the following tests:
PATTERN(0), ADDRESSING(1), PATTERN(FF) and ADDRESSING(31)

SPEED TESTS MONITOR

The speed monitor is used to verify that a given circuit works at the given speed or to estimate the maximum speed of a given circuit.

The principle of the tests is the following: The M6800 prepares a precondition for a test (sets the main memory, loads some registers, etc) and sets a short sequence of microinstructions in the writable microstore. This sequence is executed with the full processor speed and then the CPU clock is stopped. The M6800 now verifies the results (postcondition) of the execution (register values, microaddress, etc.) and reports differences.

Changing the value of potentiometer R11 on the microcontroller board changes the speed of the clock. It is therefore possible to start the test with slow execution, then speed the clock up until the first error messages occur. This estimates the maximum speed of a given circuit.

To interrupt any test strike a key on the HP keyboard.

L LOOP

Format: adr 'L'
The procedure starting at address 'adr' is executed in a loop.

S SHOW CLOCK

A sequence of 15 full speed CPU clocks is executed.
This procedure is used to measure the speed of the clock
with the scope.

To interrupt this procedure strike any key.

T TRAP

A 'jump on itself' microinstruction is loaded into
all micro memory locations.
Execute this command at the very beginning of the test session.

I INSPECT

Main memory inspect. See the description in the main memory submonitor.

U uCTRL

The program control is tranfered to the microcontroller speed
submonitor.

P CPU

The program control is transfered to the CPU speed submonitor.
.sp,3

M MEMORY

The program control is transfered to the main memory speed submonitor.

MICROCONTROLLER SPEED SUBMONITOR

J JUMPS

A microjump to all addresses is executed.

The expected maximum execution speed is about ... nsec.

ERROR messages: SUB.ID 'JUMPS'
(B,A)=Actual microaddress; X= expected microaddress.

K KONST

All constant combinations 0..255 are loaded from the CPU bus to the R0 register.

The expected maximum execution speed is about ... nsec

ERROR messages: SUB.ID 'CONST'
(B,A)=actual value; X=expected value

M JMAP

Format: 'J' 'INSTR>' instr ' U.ADR>' uadr

This procedure checks the speed of the JMAP instruction.
The procedure asks for the macro intruction 'instr'
and the microaddress of the corresponding interpreter routine 'uadr'.
The first 256 main memory locations are filled with the specified instruction and the JMAP is then executed 255 times.
The correct microaddress is verified.

The expected maximum execution speed is about ... nsec.

ERROR messages: SUB.ID 'JMAP'
(B,A)=actual microaddress; X=expected microaddress

CPU SPEED SUBMONITOR

R CARRY

The register R1 is loaded with FFFFh and
the addition with the bus constant 1 is performed.
The result should be 0.

The expected maximum execution speed is about ... nsec.

ERROR messages: SUB.ID 'CARRY'
(B,A)=expected (=0); X=actual

M MULT

Two multiplication steps with the intial values
R0=FFFFh; R1=1; Q=0 are executed.
The result should be R0=2; Q=8000h.

The expected maximum execution speed is about ... nsec.

ERROR messages: SUB.ID 'MULT'
(B,A)=expected; X=actual

P PUSH.POP

The registers R0 and R1 are loaded with various values.
In the high speed execution R0 and R1 is pushed on the stack,
then R2 and R3 popped from the stack and verified.

The expected maximum execution speed is about ...nsec.

ERROR messages: SUB.ID 'PUSH.POP'
(B,A)=expected; X=actuel

MAIN MEMORY SPEED SUBMONITOR

R READ.WRITE

Format: 'R' 'ADDR>' adr 'VALUE' val

The procedure asks for the address 'adr' and the value 'val'to be tested.
The value 'val' is written into the address 'adr', immediately
read back and verified.

The expected maximal execution speed is about ... nsec.

ERROR messages: SUB.ID 'READ.WRITE'
(B,A)=actual; X=expected.