TITLE:

CR80 TEST_RAM
USER's MANUAL

DOCUMENT NO : CSS/709/USM/0052

PREPARED BY : Synnøve M. Andersen

AUTHORIZED BY: Niels Møller Jørgensen

DISTRIBUTION :

APPROVED BY:

AUTHORITY	DATE	SIGNATURE
Dept. Mngr.	831028	NMJ

ISSUE: 2

DATE: 831028

400-810-1

CR80 TEST_RAM USER's MANUAL

sign/date	page
SMA/831028	i
repl	**project**
TLM/800810	

PAGE ISSUE RECORD AND CHANGE LOG.

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
67								
68								
69								
70								
71								
72								
73								
74								
75								
76								
77								
78								
79								
80								
81								
82								
83								
84								
85								
86								
87								
88								
89								
90								
91								
92								
93								
94								
95								
96								
97								
98								
99								
100								

ISSUE	DATE	PREPARED BY	APPROVED BY	AUTHORIZED BY
1	800810	TLM	JHØ	JHØ
2	831028	SMA	NMJ	NMJ

A

LIST OF CONTENTS PAGE

LIST OF UPDATES

1. Scope

This document describes the use of CR30 RAM TEST, a program intended to check whether or not the memory of a CR30 AMOS/XAMOS system is functioning properly.

1.1 Introduction.

The purpose of this document is to provide:

- o A description of how the memory is tested.
- o A description of the commands to the program.
- o A list of error messages that may be output from the program.
- o Examples of test sequences using the program.

The program is intended to be used in two situations :

1. When a CR80 system has been assembled, and a test is required, verifying that the CPU is able to access any RAM location properly.
2. When the memory has shown signs that it is not functioning properly, and the service technician is trying to find the faulty RAM module.

The program exists in two versions :

1> The first version is loaded in the lower half of page 0, and is able to test all memory, except addresses 0 through #17FF.

2> The second version is loaded in the upper half of page 0, and is able to test all memory, except addresses #3000 through #97FF.

A full memory test requires test run with both program versions.

1.2 Abbreviations.

Throughout this document the following abbreviations are used :

RAM : Random Access Memory

2> PROM : Programable Read Only Memory

AMU : AMOS Masterclear Utilities.

TBS : To Be Supplied

2. Applicable Documents.

1. CR80 AMOS Masterclear Utilities (AMU)
   USER's Manual.
   CSS/395/USM/0040

3. Overview.

This section contains :

- Software and hardware requirements to facilitate the test.
- Syntactical rules for input commands.

3.1 Environment.

HW requirements : CPU
2> MBC/SCM
2> RAM (min. 6K)
AV24 I/F with device number 1
Terminal
Floppy disk and I/F

SW requirements : None
The program is completely selfsupporting.

3.2 Syntactical rules.

<command file> ::= {<command>}$_{0}^{oo}$ END

<command> ::= TEST <test spec.> {<range>}$_{0}^{1}$ <cr> |
REPEAT <times> <command> |
DUMP <range> <cr>

<test spec.> ::= ALL | <single>

<single> ::= TIMING | ADDRESSING |
DATAWIRE | REFRESH |
ACCESSTIME | ADDRESSWIRE |
BIT | BYTE

<range> ::= <address>,<address> |
<address> + <length>

<times> ::= <number>

<address> ::= <page> <displacement>

<page> ::= <number>

<displacement> ::= <number>

<length> ::= <number>

<number> ::= #<hexnumber> | <decnumber>

<hexnumber> ::= {0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F}$_{1}^{4}$

<decnumber> ::= {0|1|2|3|4|5|6|7|8|9}$_{1}^{5}$

3.3 Special Character Functions

<del> (rub out)
At any time during input this character will destroy the current line.

<break>
At any time during program execution pressing the button, the program will start up again from the beginning.
This is especially useful during memory dump and in case a lot of errors occur with following error messages.

4. Function Description.

This chapter describes in detail each of the functions performed by the program.

The following abbreviations are used :

I for input syntax.

O for output message.

E for possible error messages.

FCT for function performed.

EX for input examples.

4.1 Tests in General

2> During initialization the program determines its own location in
2> memory and the memory configuration. The memory configuration is
2> determined by trying to read and write the first word of each 2K
2> address space (address 0, address #800 ect.).
2>
2> The memory is classified into: read-and-write sections, read-only
2> sections (PROM), program-containing sections and absent sections.
2> During the tests, only read-and-write sections of the test range
2> specified in the test command are tested. This is to avoid large heaps
2> of nonsense error messages.
2>
2> The smallest area the program tests is a 2K address space, even if the
2> test range in the test command specifies less then 2K.
2>
2> E.g. the command
2>
2> TEST ALL 0 #A100, 0 #C4FF <cr>
2>
2> will imply that page 0 address #A000 to page 0 address #C7FF are
2> tested. When a test command is used without a range specification page
2> 0 address #0000 to page 3 address #FFFF are tested.

Tests terminating, without errors occurring , will only give the response that the program prints a ':' and waits for a new command. Error messages are described in section 6.

4.2 TEST ALL

FCT: Perform all tests in all memory or in the specified range.

Only the tests that do not require operator reactions are included.

The following tests are performed :

Addressing	(see 4.4)
Datawire	(see 4.5)
Refresh	(see 4.6)
Address Wire	(see 4.3)
Bit	(see 4.9)
Byte	(see 4.10)

The result of the TEST ALL command should give a quite good hint to where eventual errors are to be located.

I: TEST ALL {<range>}$^{1}_{0}$ <cr>

E: Union of the possible error messages from the single tests.

EX :

1. TEST ALL <cr>
2. TEST ALL 0 #A000, 1 #BFFF <cr>
3. TEST ALL 0 #A000 + #FFF <cr>

4.3 TEST TIMING

FCT: This command does not actually make the program test the timing (BUS timing), but it gives a possibility to test it.

The program repeats reading from the same address in memory until <break> is pressed.

If a range is specified, the program will read from the first address of the specified range, rounded down to the nearest 2K displacement.
Otherwise it will read from the first address in the area where test is possible.

The timing may be measured using an oscilloscope or another electronic device.

I : TEST TIMING {<range>}$^{1}_{0}$ <cr>

E : 9, 10, 11

EX :

1. TEST TIMING <cr>
2. TEST TIMING 1 #F000 + #FFF

4.4 TEST ADDRESSING

FCT: This test contains two subtests: a) an addressing test within pages and b) an addressing test across pages. During subtest a) it is checked that every word within a page is able to hold its own address (page offset). During subtest b) it is checked that the first word in the first read-and-write memory section in every page is able to hold the value: pagenumber + 1, and that this value is not contained in any other word in the page.

Operation sequence :

Subtest a)

1. Clear the memory area (insert 0's)
2. For each address in ascending order the following is done :
   - The contents of the address is fetched, and it is checked that it is a zero.
   - The value of the address is stored in the location.
   - The contents of the address is fetched and it is checked, that it has the same value as the address.
3. Clear the memory area.
4. Operation 2 is performed for addresses in descending order.

Subtest b)

1. Clear the memory area.
2. Store pagenumber+1 in the first word in the first read-and-write section in every page.
3. Check that the contents of the first word in the first read-and-write section in every page is pagenumber+1, and that the contents of every other word in the pages is zero.

I : TEST ADDRESSING {<range>}$^{1}_{0}$ <cr>

E : 2, 10, 11

EX :

1. TEST ADDRESSING <cr>
2. TEST ADDRESSING 0 #8000, 0 #3FFF <cr>
3. TEST ADDRESSING 0 #8000 + #FFF <cr>

4.5 TEST DATAWIRE

FCT: This is both a speed and a data test

2> Operation sequence:

2> 1 The memory is at highest speed filled with a
2> checkerboard pattern (#AAAA,#5555).
2> 2. It is checked that the memory contains the
2> expected pattern.
2> 3. Operation 1 and 2 is performed with inverted
2> checkerboard pattern (#5555,#AAAA).

I : TEST DATAWIRE {<range>}$^{1}_{0}$ <cr>

E : 3, 10, 11

EX :

1. TEST DATAWIRE <cr>
2. TEST DATAWIRE 1 #3000, 2 #7FFF <cr>
3. TEST DATAWIRE 1 #3000 + #1FFF <cr>

4.6 TEST REFRESH

FCT: Checks if the memory is able to hold
a checkerboard pattern for more
2> than 60 seconds without CPU access.

Operation sequence :

1. The memory is filled with a checkerboard pattern (#AAAA,#5555).
2. The program waits for approx.
2> 60 seconds without memory access.
3. It is checked that the pattern is still a correct checkerboard pattern.
2> 4. Operation 1,2 and 3 is performed with inverted
2> checkerboard pattern (#5555,#AAAA).

I : TEST REFRESH {<range>} <cr>
1
0

E : 4, 10, 11

EX :
1. TEST REFRESH <cr>
2. TEST REFRESH 3 #E000, 3 #FFFF <cr>
3. TEST REFRESH 3 #E000 + #1FFF <cr>

4.7 TEST ACCESSTIME

FCT: This command does not actually test the access time, but it gives a possibility to test it.

The program accesses the RAM area with 2**24 read cycles and 2**24 write cycles.
The time for these accesses may be measured.
A bell-key signal is output before the accesses begin and after they are completed.
A normal time between the bell-key signals should be approx. 22.5 seconds.

I : TEST ACCESSTIME {<range>} <cr>
1
0

E : 9, 10, 11

EX :

1. TEST ACCESSTIME <cr>
2. TEST ACCESSTIME 1 #A000, 1 #BFFF <cr>
3. TEST ACCESSTIME 1 #A000 + #FFF <cr>

4.8 TEST ADDRESSWIRE

FCT: Checks if the address wires function properly.

Operation sequence:

1. The program inserts the address of the locations in the locations, all over the memory.

2. For each word in the memory, the program accesses the word, inserts its contents in the word with the bit inverted address. Then it checks if the words have the same contents. If the word with the inverted address does not exist, the test is not performed in that location.

I : TEST ADDRESSWIRE {<range>}$^{1}_{0}$ <cr>

E : 6, 10, 11

EX :

1. TEST ADDRESSWIRE <cr>
2. TEST ADDRESSWIRE 0 #2000, 0 #3FFF <cr>
3. TEST ADDRESSWIRE 0 #2000 + #1FFF <cr>

4.9 TEST BIT

FCT: Checks if any bit in the memory is able to be set and cleared.
It is done, first by using a walking one pattern, where a 1 is moving from right to left in all bits in the words.
Then it is done by letting a zero walk in all of the bits.
At test termination the tested memory area contains the pattern #7FFF.

I : TEST BIT {<range>}$^{1}_{0}$ <cr>

E : 7, 10, 11

EX :

1. TEST BIT <cr>
2. TEST BIT 3 #F000, 3 #FFFF <cr>
3. TEST BIT 3 #F000 + #0FFF <cr>

4.10 TEST BYTE

FCT: Checks if any byte in the memory is addressable (both readable and writable).
At test termination the tested memory area contains the pattern #AA55.

I : TEST BYTE {<range>}$^{1}_{0}$ <cr>

E : 8, 10, 11

EX :

1. TEST BYTE <cr>
2. TEST BYTE 0 #4000, 2 #1FFF <cr>
3. TEST BYTE 0 #4000 + #FFFF <cr>

4.11 REPEAT

FCT: With this command it is possible to make the program repeat one of the other commands a number of times.
It should be quite useful to test systems which occasionally have a parity error. It is then possible to make a long term test and find out where the parity error occurs and replace the faulty module.

I : REPEAT <times> <command> <cr>

O : *** END OF REPEATING ***

E : Any of the errors, that may occur in the command to be executed.

EX :

1. REPEAT 100 TEST BIT <cr>
2. REPEAT #100 TEST ALL 1 #C000 + #FFF <cr>
3. REPEAT 10000 TEST BYTE <cr>

4.12 DUMP

FCT: Dumps a part of the memory.
Any memory in the 256K address space of CR80 may be dumped.
The starting address will be rounded down to the nearest 16 word displacement.
Crossing a page boundary, the program will continue the dump from the next memory page.
<break> will at any time terminate the memory dump.
An attempt to dump non-existing memory will give the error message TIMEOUT ERROR for each non-existing word in the area to be dumped.

I : DUMP <range> <cr>

O : Data locations in hex. code.

EX :
1. DUMP 0 #2000 + #100 <cr>
2. DUMP 3 #F000, 3 #FFFF <cr>

4.13 END

FCT: Terminates the execution of the program.

I : END

5. Activation.

The program is activated via the bootstrap load facility in the AMU program.

It is activated like this :

Masterclear the system.
Load the relevant floppy disk in disk drive 0 or 1.
Type this to the AMU program :

```
Boot Fd  {0 | 1}
^    ^   ^  ^
```

(Only the characters with the arrows under are typed)

After a little while the program should reply

```
2>   CSS/709 RAM TEST VERSION 0103
2>   RAM: 11111111111111111111111111111111,11111111111111111111111111111111
2>        ................................,11111111........1111111111111111
2>
2>   NOT: 111.............................1,................................
2>        11111111111111111111111111111111,........11111111................
:
```

and is now ready to accept and execute commands.

The line 'RAM : ' is interpreted like this :

```
2>    A '1' indicates a responding 2K
2>          memory area.
2>    A '.' indicates a 2K word memory area
2>          which does not respond to read
2>          access attempts.

2>         |<---------- page 0 ---------->| |<---------- page 1 ---------->|
2>    RAM: 11111111111111111111111111111111,11111111111111111111111111111111
2>         |<---------- page 2 ---------->| |<---------- page 3 ---------->|
2>         ................................,11111111........1111111111111111
2>         ^                ^                ^
2>       adr                |              adr
2>     0..#07FF             |          #F800..#FFFF
2>                        adr
2>                   #7000..#77FF
```

2> The line 'NOT:' is interpreted like this:

2> A '1' indicates that no test can be performed in this 2K area
2> (because it is nonexsisting,PROM or contain the RAM_TEST
2> program itself).
2>
2> A '.' indicates that the tests can be performed in this area.

6. Error Codes and Messages.

The error messages resulting from data faults in the memory, has the following format :

*** <type> ERROR *** PAGE <no> WORD <no> { CONT <no> EXP <no> }

Where <type> is one of :

1 TIMING TEST
2> 2a ADDRESSING TEST WITHIN PAGE
2> 2b ADDRESSING TEST ACROSS PAGES
3 DATAWIRE TEST
4 REFRESH TEST
5 ACCESSTIME TEST
6 ADDRESSWIRE TEST
7 BIT TEST
8 BYTE TEST
9 PARITY
10 TIMEOUT
11 TRAP

PAGE <no> is the physical page number.

WORD <no> is the displacement within the page.

CONT <no> is the contents of the fault word.

EXP <no> is the expected contents of the fault word.

CONT and EXP are only output with the error messages 1 through 8.

The program reacts to syntax- and semantic errors by printing :

*** SYNTAX ERROR ***

and skipping the current line.

7. Examples

```
CSS/395/04  811101
CPU 0
>BOOT FD: 1, START= 0400 , BASE= 0400

CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: 111.............................1,................................
     11111111111111111111111111111111,........11111111................
:TEST ALL
:TEST ACCESSTIME              "TIME APPROX. 23 SEC."
:END                          "RETURN TO AMU PROGRAM"
*** MEMORY TEST COMPLETE ***


CSS/395/04 811101
CPU 0
>

BOOT FD: 1, START= 3400 , BASE= 3400

CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: ................111.............1,................................
     11111111111111111111111111111111,........11111111................
:REPEAT 10000 TEST DATAWIRE  " LONG TERM TEST"
```

```
CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: ................111.............1,................................
     11111111111111111111111111111111,........11111111................
:"<BREAK> WAS PRESSED"  *DEL
:DUMP 0 0, 3 #FFFF            "DUMPS 256K WORDS"
0 #0000L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0008L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0010L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0018L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0020L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0028L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0030L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555
0 #0038L #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555


CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: ................111.............1,................................
     11111111111111111111111111111111,........11111111................
:TEST BIT
:DUMP 0 #20F0 + #40
0 #20F0L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #20F8L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2100L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2108L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2110L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2118L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2120L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
0 #2128L #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF #7FFF
:END
*** MEMORY TEST COMPLETE ***
```

```
CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: 111.............................1,................................
     11111111111111111111111111111111,........11111111................
:TEST ALL 0 0 + #1FFF
:TEST REFRESH                  "TEST REFRESH IN ALL MEMORY"
:DUMP 1 #FFE0, 1 #FFFF
1 #FFE0L #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA
1 #FFE8L #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA
1 #FFF0L #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA
1 #FFF8L #5555 #AAAA #5555 #AAAA #5555 #AAAA #5555 #AAAA
:END
*** MEMORY TEST COMPLETE ***


CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: 111.............................1,................................
     11111111111111111111111111111111,........11111111................
:TEST BYTE
*** PARITY ERROR *** PAGE 3 WORD #0002
*** PARITY ERROR *** PAGE 3 WORD #0003
*** PARITY ERROR *** PAGE 3 WORD #0004
*** PARITY ERROR *** PAGE 3 WORD #0005
*** PARITY ERROR *** PAGE 3 WORD #0006
*** PARITY ERROR *** PAGE 3 WORD #0007
*** PARITY ERROR *** PAGE 3 WORD #0008
*** PARITY ERROR *** PAGE 3 WORD #0009
```

```
CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: 111.............................1,................................
     11111111111111111111111111111111,........11111111................
:TEST ADDRESSWIRE
:DUMP 3 0 + 6
*** PARITY ERROR *** PAGE 3 WORD #0000
*** PARITY ERROR *** PAGE 3 WORD #0001
*** PARITY ERROR *** PAGE 3 WORD #0002
*** PARITY ERROR *** PAGE 3 WORD #0003
*** PARITY ERROR *** PAGE 3 WORD #0004
*** PARITY ERROR *** PAGE 3 WORD #0005
*** PARITY ERROR *** PAGE 3 WORD #0006
*** PARITY ERROR *** PAGE 3 WORD #0007


CSS/709 RAM TEST VERSION 0103
RAM: 11111111111111111111111111111111,11111111111111111111111111111111
     ................................,11111111........1111111111111111

NOT: 111.............................1,................................
     11111111111111111111111111111111,........11111111................
:TEST BIT
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6001 EXP #0001
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6002 EXP #0002
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6004 EXP #0004
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6008 EXP #0008
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6010 EXP #0010
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6020 EXP #0020
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6040 EXP #0040
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6080 EXP #0080
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6100 EXP #0100
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6200 EXP #0200
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6400 EXP #0400
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #6800 EXP #0801
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #7000 EXP #1000
*** PARITY ARROR *** PAGE 3 WORD #0000
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #FFFF EXP #2000
*** PARITY ARROR *** PAGE 3 WORD #0000
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #FFFF EXP #4000
*** BIT TEST ERROR *** PAGE 3 WORD #0000 CONT #E000 EXP #8000
```