Memory Test Software for the M68000 Family

By Gordon Lawton
Datacommunications Applications
Motorola, East Kilbride, Scotland

Introduction

On initialisation, many embedded systems will run a complete system test to identify any problems. As memory arrays become larger, the potential for errors increases, so an integral part of the system test are memory tests. It is important that such code is compact and fast to minimize the overall effect on the system.

This Engineering Bulletin contains memory test code for 68K family processors. The code is written is 68K assembly code to optimise exectution time and reduce size.

Test Methodology

The code executes the memory tests sequentially as shown in figure 1. If a test fails the code aborts and flags the error. On successful completion of all the a pass flag is set.

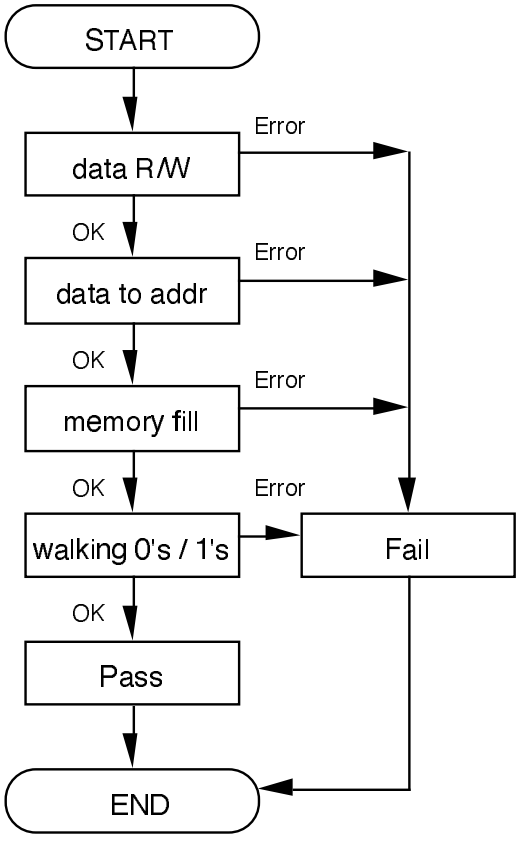

Figure 1. **Memory Test Flow Diagram**

The test are carried out over all the memory locations. The code consists of four separate tests:

- **Data write with read-back compare**
- **Write data address to memory**
- **Memory Fill and Test**
- **Walking 0's and 1's**

Data write with read-back compare

This test writes several data patterns, $0, $5, $A and $F, to each memory location. and then reads the data back into a CPU register. The register contents are then compared with the original data. In effect the test run four times, once for each pattern. Read and write cycles are long word wide. The outcome of this test verifies correct storage operation to RAM.

Write data address to memory

This test stores every memory address as a data value to store at that address. This stored value is then read-back and compared with the original address value. The whole of memory is filled with data before the read-back operation begins. If the comparison is correct then all address lines must be functioning correctly.

Memory Fill and Test

All internal RAM is filled with a background pattern of $0. A selected location is then loaded with a data pattern $AAAAAAAA. The module then checks all other locations to determine if any location containing the background pattern has been modified, ie it checks for aliasing. This test checks that the address and data lines are correctly connected.

Walking 0's and 1's

This is a bit oriented test which, after block filling memory with 1's, sets every bit to 0 and then back to 1 sequentially. When the bit is set to 0, the data is compared with the known correct value to check for correct operation of each bit of RAM.

Assembly Code Operation

The assembly code rountines below define start and end addresses, **MEMST** and **MEMEND**. These should be set as for the system memory. On successful operation the contents of D1 are $12345678. On a test failure the contents of the D1 are $DEADDEAD. The erroneous address will be stored in A2. In this example a TRAP #$0 command is used to return control to a dubugger, this should be altered for specific systems.

```
*************************************************************************
* Memory Test      Gordon Lawton                                        *
* Last Modified: 31/3/94    Version 1.0                                 *
* this code runs a suite of memory tests over a specified memory        *
* area.  The tests are as follows:                                      *
*                                                                       *
* data r/w                                                              *
* data/address connectivety                                             *
* memory fill test                                                      *
* walking 1/0s test                                                     *
*                                                                       *
* A fuller description of the code is given in the subroutines.         *
*                                                                       *
* <C> Copyright Motorola 1994                                           *
* Change History:    Ver 1.0  Initial release                           *
*************************************************************************
* Initial conditions: NONE                                              *
* Final conditions:   Memory Tested                                     *
*                                                                       *
* Input parameters:   NONE                                              *
* Output parameters:  Test results in D1                                *
*                     Pass: $12345678 in D1                             *
*                     Fail: $DEADDEAD in D1                             *
*************************************************************************


* Definition of constants used in the code   *

MEMST     EQU         $02000000   Memory to be tested start address
MEMEND    EQU         $020FFFFF   Memory to be tested end address
PATTERNA  EQU         $AAAAAAAA   Test pattern A
PATTERNB  EQU         $55555555   Test pattern B
PATTERNC  EQU         $FFFFFFFF   Test pattern C


* XRAM_Test                                      *
* All memory tests are called from this routine  *
* The exit TRAP requires to be altered depending *
* on the system used                             *
          MOVEA.L    #MEMST,A0            load start address
          MOVEA.L    #MEMEND,A1           load end address
          BSR        XRAMDRW              do ram data r/w test
          BSR        XRAMDA               do ram data address test
          BSR        XRAMMF               do ram memory fill test
          BSR        XRAMWALK             do ram walking 1/0s test
          MOVE.L     #$12345678,D1        put pattern in d1 if passed
EXIT      TRAP       #$0                  exit back to probe debugger

* RAM_Data_R/W                                    *
*                                                 *
* Test writes a data pattern to memory then reads *
* It back to check it has been written correctly  *
* This test is run for 00000000, 55555555, AAAAAAAA *
* and FFFFFFFF.  It will verify correct storage   *
* operation of the RAM.                           *
*                                                 *
* Initial conditions: NONE                        *
* Final conditions:   Pass: then return           *
*                     Fail: then error routine    *
* Input parameters:   A0 = MEMST, A1 = MEMEND     *
* Output parameters:  NONE                        *

XRAMDRW   MOVE.L     A0,A2                copy start address
          CLR.L      D4                   clear to write 0s
FILL1     MOVE.L     D4,(A2)+             write to memory
          CMPA.L     A2,A1                check location
          BPL        FILL1                at end of block?
          MOVE.L     A0,A2                copy start address
COMP1     CMP.L      (A2)+,D4             check written ok
          BNE.L      XDRERR               quit on error
          CMPA.L     A2,A1                check location
          BPL        COMP1                at end of block
          MOVE.L     A0,A2                copy start address
          MOVE.L     #PATTERNB,D4         This time write 5s
FILL2     MOVE.L     D4,(A2)+             write to memory
          CMPA.L     A2,A1                check location
          BPL        FILL2                at end of block?
```

```
          MOVE.L     A0,A2                copy start address
COMP2     CMP.L      (A2)+,D4             check written ok
          BNE.L      XDRERR               quit on error
          CMPA.L     A2,A1                check location
          BPL        COMP2                at end of block
          MOVE.L     A0,A2                copy start address
          MOVE.L     #PATTERNA,D4         This time write As
FILL3     MOVE.L     D4,(A2)+             write to memory
          CMPA.L     A2,A1                check location
          BPL        FILL3                at end of block?
          MOVE.L     A0,A2                copy start address
COMP3     CMP.L      (A2)+,D4             check written ok
          BNE.L      XDRERR               quit on error
          CMPA.L     A2,A1                check location
          BPL        COMP3                at end of block
          MOVE.L     A0,A2                copy start address
          MOVE.L     #PATTERNC,D4         This time write Fs
FILL4     MOVE.L     D4,(A2)+             write to memory
          CMPA.L     A2,A1                check location
          BPL        FILL4                at end of block?
          MOVE.L     A0,A2                copy start address
COMP4     CMP.L      (A2)+,D4             check written ok
          BEQ.L      XDROK                quit on error
XDRERR    BSR        XRAMERR              jump to error routine
XDROK     CMPA.L     A2,A1                check location
          BPL        COMP4                at end of block
          RTS                             Return to do next test


* RAM_Data_Address                              *
* This test uses the current memory address    *
* as a data value to store at the address The  *
* data is read back and compared thus tests    *
* the address lines                            *
*                                              *
* Initial conditions: NONE                     *
* Final conditions:   Pass: then return        *
*                     Fail: then error routine *
* Input parameters:   A0 = MEMST, A1 = MEMEND  *
* Output parameters:  NONE                     *

XRAMDA    MOVE.L     A0,A2            copy start address
FILL5     MOVE.L     A2,(A2)+         write addr as data to addr
          CMPA.L     A2,A1            check location
          BPL        FILL5            at end of block?
          MOVE.L     A0,A2            copy start address
COMP5     CMPA.L     (A2),A2          write address as data to itself
          BNE.L      XDAERR           quit on error
          TST.L      (A2)+            ignore test, increment address
          CMPA.L     A2,A1            check location
          BPL        COMP5            at end of block?
          BRA        XDAOK            skip error call if here
XDAERR    BSR        XRAMERR          jump to error routine
XDAOK     RTS                         Return to do next test

* RAM_Memory_Fill                               *
* After memory is filled with a background of 0s *
* a single location is written with a pattern.   *
* this checks if the data and address lines are  *
* connected correctly                            *
*                                                *
* Initial conditions: NONE                       *
* Final conditions:   Pass: then return          *
*                     Fail: then error routine   *
* Input parameters:   A0 = MEMST, A1 = MEMEND    *
* Output parameters:  NONE                       *


XRAMMF    MOVE.L     A0,A2                copy start address
          CLR.L      D4                   clear to write 0s
FILL6     MOVE.L     D4,(A2)+             write to memory
          CMPA.L     A2,A1                check location
          BPL        FILL6                at end of block?
          MOVE.L     A1,A2                copy end address
          SUBA.L     A0,A2                start-end = block length
          MOVE.L     A2,D4                put length in D4 for shift
          LSR.L      #$4,D4               divide by 16 to get offset
```

```
          LSL.L      #$2,D4                mult by 4 :long word aligned
          MOVE.L     #PATTERNA,D5          copy As pattern
          MOVE.L     D5,0(A0,D4.L)         write patA to start+offset
          LEA.L      0(A0,D4.L),A3         copy address of offset
          MOVE.L     A0,A2                 copy start address
COMP6     TST.L      (A2)                  make sure location is 0
          BEQ        MFPASS                ok if 0
          CMPA.L      A3,A2                at offset?
          BNE        XMFERR                error if not at offset
          CMP.L      (A2),D5               check data at offset
          BNE        XMFERR                quit if not correct
MFPASS    TST.L      (A2)+                 increment address
          CMP.L      A2,A1                 check location
          BPL        COMP6                 at end of block?
          BRA        XMFOK                 skip error call if here
XMFERR    BSR        XRAMERR               jump to error routine
XMFOK     RTS                              Return to do next test


* RAM_Walking_1/0                                 *
* after block filling the memory with 1s every   *
* bit is set to 0, tested, and set back to 1     *
*                                                 *
* Initial conditions: NONE                        *
* Final conditions:   Pass: then return           *
*                     Fail: then error routine    *
* Input parameters:   A0 = MEMST, A1 = MEMEND     *
* Output parameters:  NONE                        *


XRAMWALK  MOVE.L     A0,A2                 copy start address
          MOVE.L     #PATTERNC,D4          This time write Fs
FILL7     MOVE.L     D4,(A2)+              write to memory
          CMPA.L     A2,A1                 check location
          BPL        FILL7                 at end of block?
          MOVE.L     A0,A2                 copy start address
          MOVE.L     #$FFFFFFFE,D4         blank LSB
COMP7     MOVE.L     #$1F,D5               count of 31
WALKLOOP  MOVE.L     D4,(A2)               write next bit memory
          CMP.L      (A2),D4               bit O.K?
          BNE        XWKERR                error if not the same
          ROL.L      #$1,D4                blank next bit
          DBF        D5,WALKLOOP           at last bit?
          MOVE.L     #PATTERNC,(A2)+       clear last bit, inc address
          CMP.L       A2,A1                check location
          BPL        COMP7                 at end of block?
          BRA        XWKOK                 skip error call if here
XWKERR    BSR        XRAMERR               jump to error routine
XWKOK     RTS                              return to main routine

* ERROR HANDLING ROUNTINE                        *
* Set error pattern on D1                        *
*                                                *
* Initial conditions: Failure                    *
* Final conditions:   Error condition set        *
* Input parameters:   NONE                       *
* Output parameters:  D1 = DEADDEAD              *
*                                                *
XRAMERR   MOVE.L     #$DEADDEAD,D1          set error pattern
          BRA        EXIT                   exit

         end
```