

FAILURE ANALYSIS

UCB2 FAMILY - E102-E104-WATCHDOG

January 18, 2007

Product Information

Customer Name	General
Product Name	CD Series 5 PicoDrive Series 5 TRMM2 Rack Mount Drive
Failure Analysis Status	Complete
Prepared by	Sergey Salnikov, Hanan Hurwitz Hanan.hurwitz@danahermotion.com

Problem Description

We have had quite a few cases of the UCB2-based drives going into watchdog at power up, with the display showing E102-E104 and then WD.

The problem seems to occur randomly, on units that are in the field. At least, there is no recognizable pattern for the problem.

Background

Drive Electronics Commonality

Three product lines share the same basic electronic design on the digital control board. These products are the CD Series 5, the PicoDrive Series 5, and the TRMM2 Rack Mount Drive. The problem has been seen on all these products.

Drive Firmware Architecture

The drive firmware is actually composed of three different files. These are:

- Firmware for the HPC micro-processor
- Altera FPGA image
- Firmware for the DSP processor

Root Cause Analysis

During initialization the drive tests the Dual-Port RAM (DPRAM) that is implemented in the Altera FPGA. This test fails, resulting in the E102 fault.

The root cause is not known. However, it has been found that the problem seems to be solved by compiling the Altera (FPGA) image with the QUARTUSII compiler instead of with the MAXPLUSII compiler. The functionality of the FPGA does not change.

The Appendix contains a complete technical report by Sergey Salnikov.

Corrective Action

The Altera image is recompiled for specific firmware versions. At present, versions are being made based 7.3.3, 7.4.2 and 6.1.0e.

The Altera images in all future revisions of UCB2 firmware in the future will be compiled using the QUARTUS compiler.

Appendix: Technical Report

Fault “e102,e104” description and RCA.

The fault “e102” means “Altera DPRAM failure (during initialization)”.
This fault occurs during drive initialization process.

Description:

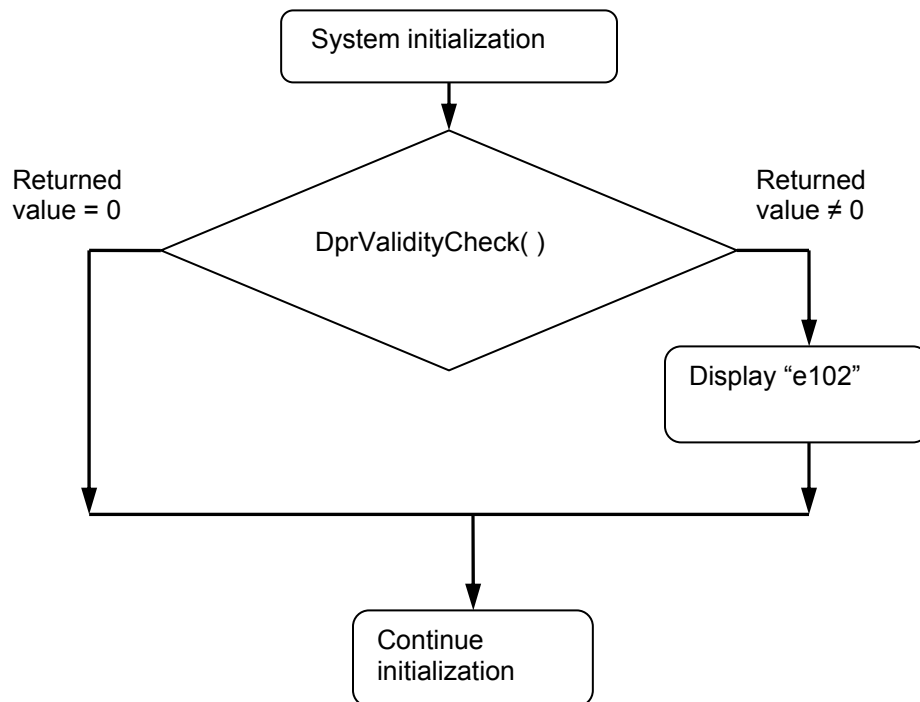
DPRAM (dual-port RAM) is Altera’s megafunction that we are using for interfacing between HPC and DSP like intermediary.

DPRAM is testing during system initialization after Altera initialization.

DPRAM test:

System initialization program call “DprValidityCheck” function.

If returned value is NOT equal to zero then indicate that the DPRAM validity check failed. Display “e102” error and continue.

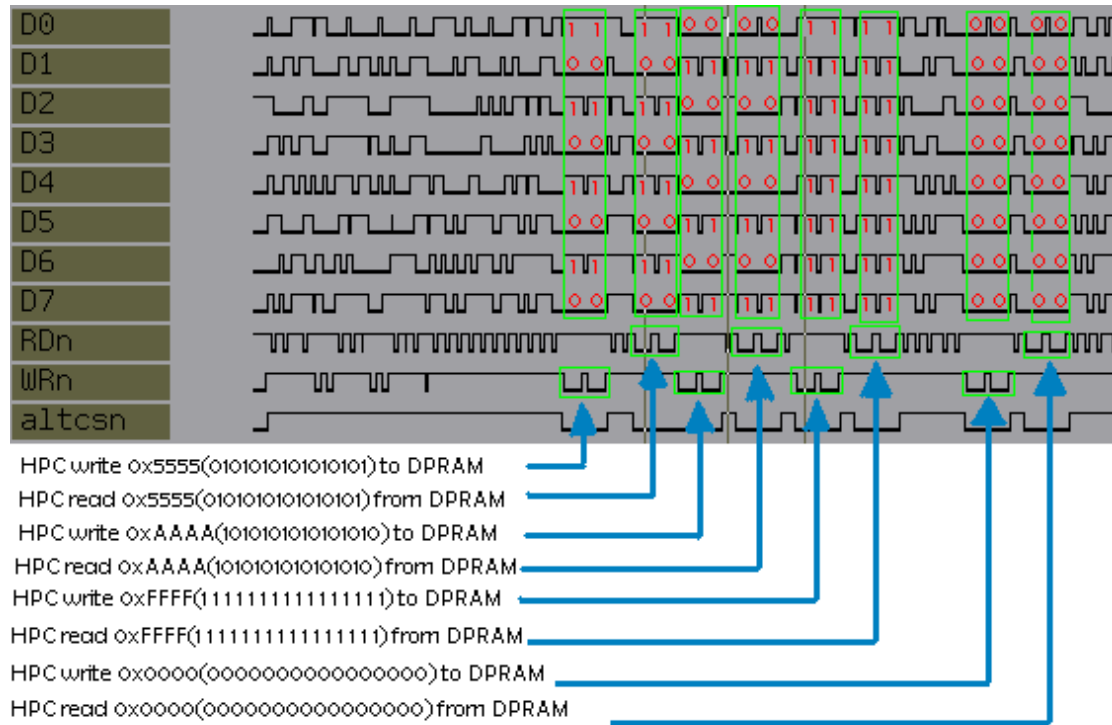


This function checks read/write validity of each bit of the DPRAM first block words (one by one). After all read/write operations with word the last result should be ZERO. It's a sign that all 16 bits of this word passed read/write validity and function can check next word. If result is greater than ZERO the function is stopping and returning this value. In this way function checks all 256 words of the DPRAM first block.

“DprValidityCheck” review:

1. HPC write 0x5555 to DPRAM start address (0xE800).
2. HPC read value from DPRAM (should be 0x5555) and then write 0xAAAA to DPRAM.
3. HPC read value from DPRAM (should be 0xAAAA) and add to previous value (0x5555). Result should be 0xFFFF.

4. HPC write calculated value to DPRAM.
5. HPC read value from DPRAM (should be 0xFFFF) and complement received value.
6. HPC check if value greater than zero.
7. If greater, jump to end and return last value.
8. Else write this value (0x0000) to DPRAM.
9. HPC read value from DPRAM (should be 0x0000).
10. HPC check if value greater than zero.
11. If greater, jump to end and return last value.
12. Else go to next DPRAM word check.
13. HPC check in this way all 256 words.



```
int DprValidityCheck(void)
```

```
{
/$
```

```
    LD    K, #DPR_B1_WSIZE
    LD    B, #DPR_B1_START_ADDR
CHECK_WORD:                                ;estimated execution time per word: 4 uS
    LD    A, #0x5555
    ST    A,[B].w                          ; write 0x5555
    COMP A
    X     A,[B].w                          ; read 0x5555, write 0xAAAA
    ADD   A,[B].w                          ; read 0xAAAA
    ST    A,[B].w                          ; write 0xFFFF
    LD    A,[B].w                          ; read 0xFFFF
    COMP A
    IFGT A,#0
    JP    DPR_CHECK_END
    ST    A,[B].w                          ; write 0x0000
    LDS   A,[B+].w                        ; read 0x0000
    NOP
```

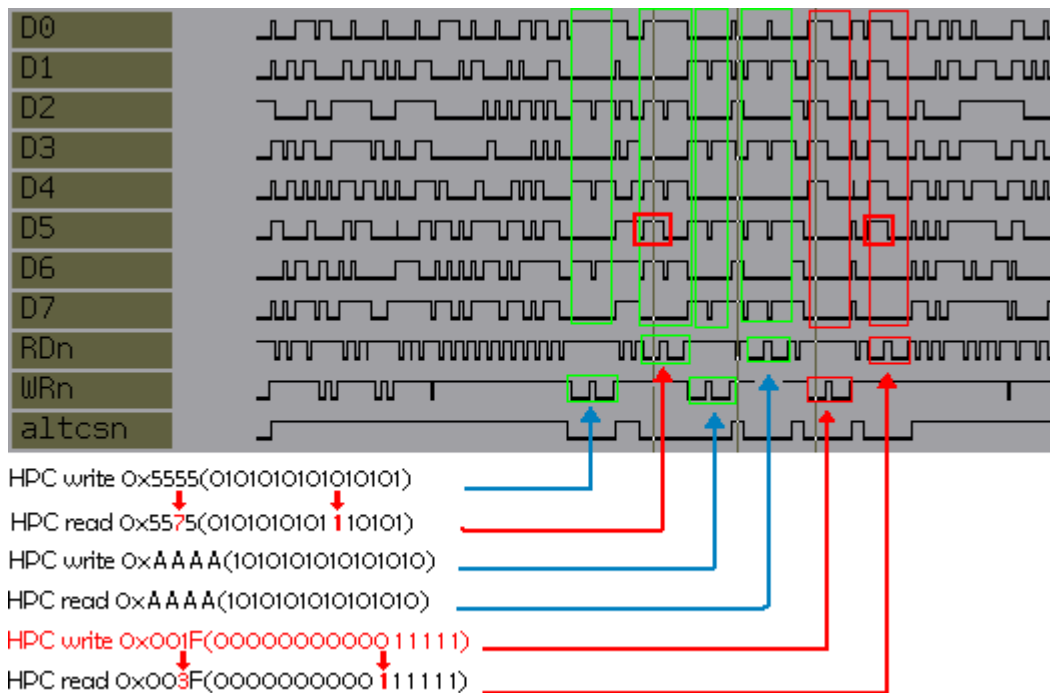
```

IFGT A, #0
JP      DPR_CHECK_END
DECSZ K
JP      CHECK_WORD
; Successful check: A=0, failed: A>0
DPR_CHECK_END:
$/
}

```

In our case we have failure during reading from DPRAM and as result failure in next read/write operations and calculations.

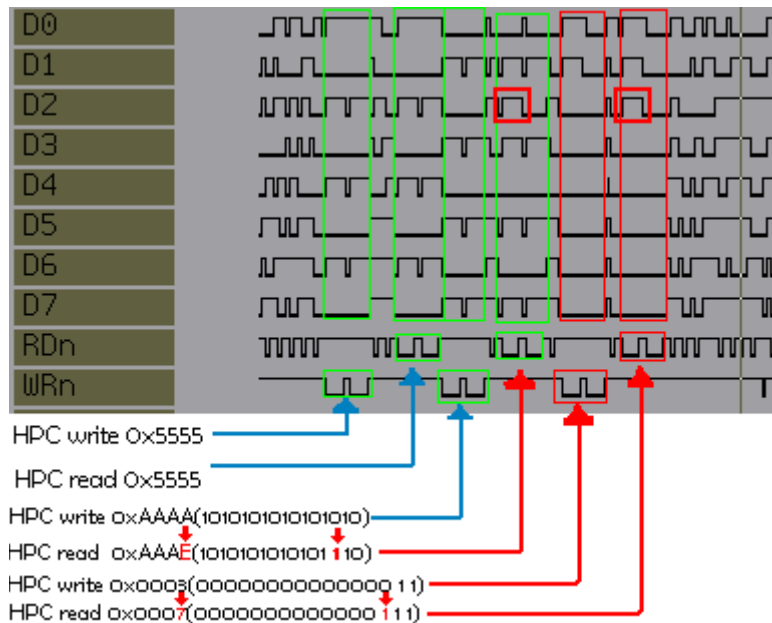
Example 1:



In this diagram we can see failure in bit 5.

HPC write 0x5555 to DPRAM and read back 0x5575 instead 0x5555 (bit 5 value is "1" instead "0"). Then HPC write/read 0xAAAA to/from DPRAM. After it HPC add current received value to previous received value (0x5575 + 0xAAAA = 0x1001F) and write calculated value (0x001F) to DPRAM. Then HPC read back value 0x003F with failure in bit 5. HPC complement received value and because of the calculated value greater than ZERO the function was stopping and returned this value. Of course DPRAM validity check failed because of returned value greater than ZERO.

Example 2:



In this diagram we can see failure in bit 2.

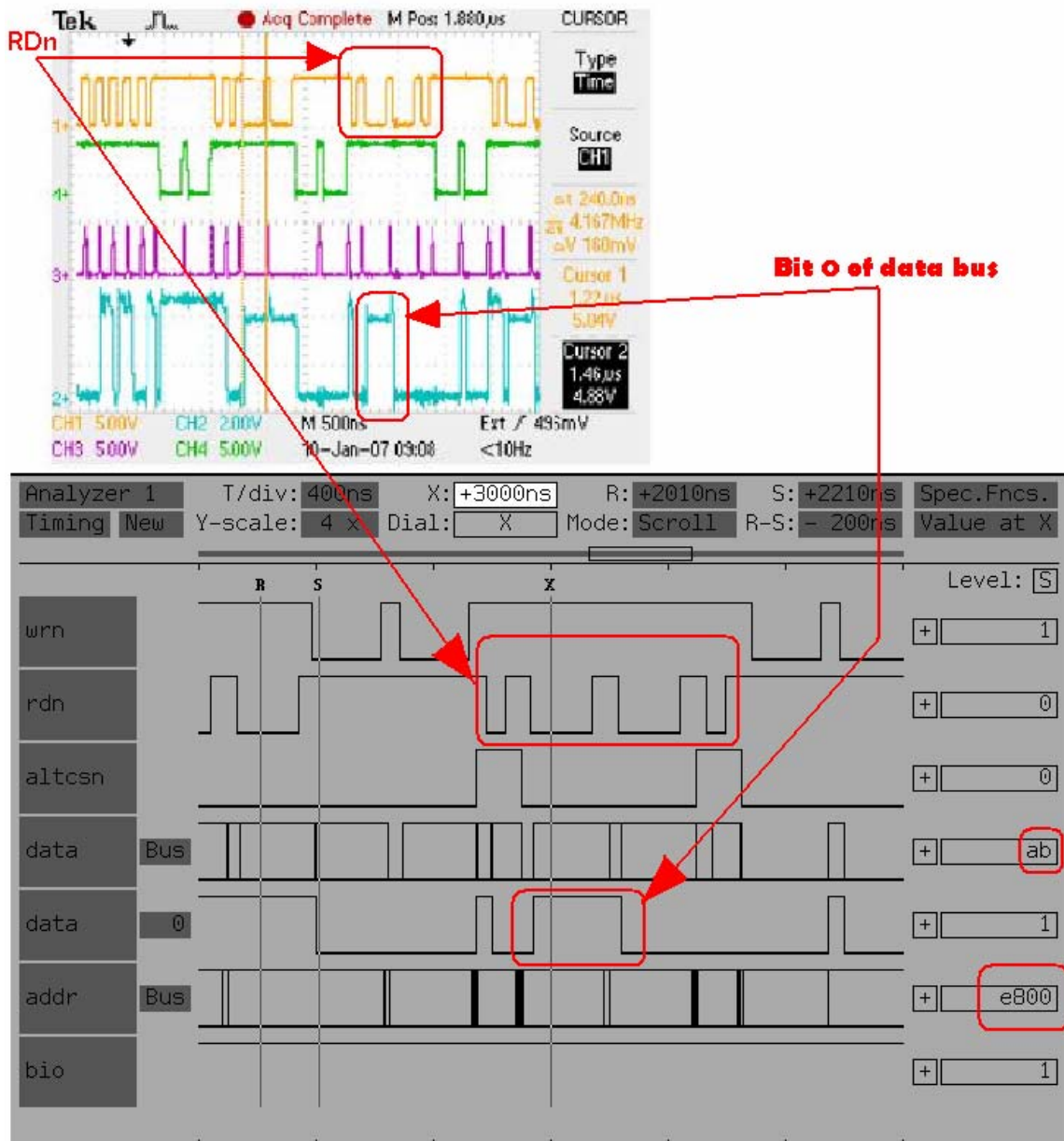
HPC write/read 0x5555 to/from DPRAM. Then HPC write 0xAAAA to DPRAM and read back 0xAAAE instead 0xAAAA (bit 2 value is "1" instead "0"). HPC add current received value to previous received value ($0x5555 + 0xAAAE = 0x10003$) and write calculated value (0x0003) to DPRAM. Then HPC read back value 0x0007 with failure in bit 2. HPC complement received value and because of the calculated value greater than ZERO the function was stopping and returned this value. Of course DPRAM validity check failed because of returned value greater than ZERO.

This phenomenon occurs with:

1. Altera A021a5 version only when Encoder/Sine Encoder configuration in use . This version included into UCB versions 7.0.1 – 7.4.3.
2. Altera A020a4 version only when Resolver configuration in use . This version included into UCB versions 6.0.8 – 6.1.0G.

This configuration code compiled by MAXPLUSII software. But when this code compiled by QUARTUSII software the failure disappears.

According to Ari's supposition took place Flash ([AM29F800](#)) timing issue because 70ns flash was assembled instead 55ns as planned. But after measurements by Scope&LogicAnalyzer this supposition doesn't true. We can see that signal clear, without collision and stand in RDn time. In bottom picture we can see BIT0 of data bus. This bit create failure in this case.



According to Ari and Boris comments took place Altera's internal timing issue.

Boris comments:

"Quartus II software from Altera corporation is new tool that comes instead of old one MaxPlus II . The difference between them is significant. Here is some features of the Quartus II:

- Better user interface.
- Better VHDL analysis and synthesis.
- Requires an average of 5% fewer device resources for a given MAX design.
- Better place & route algorithms etc.

As the result of all that we have better fitting of the same design.

For example, UCB2 Altera design version A21a5 compiled by MAXPLUS II uses 1698 Logic Cells(LC), the same design compiled by QUARTUS II uses 1647 LC. "