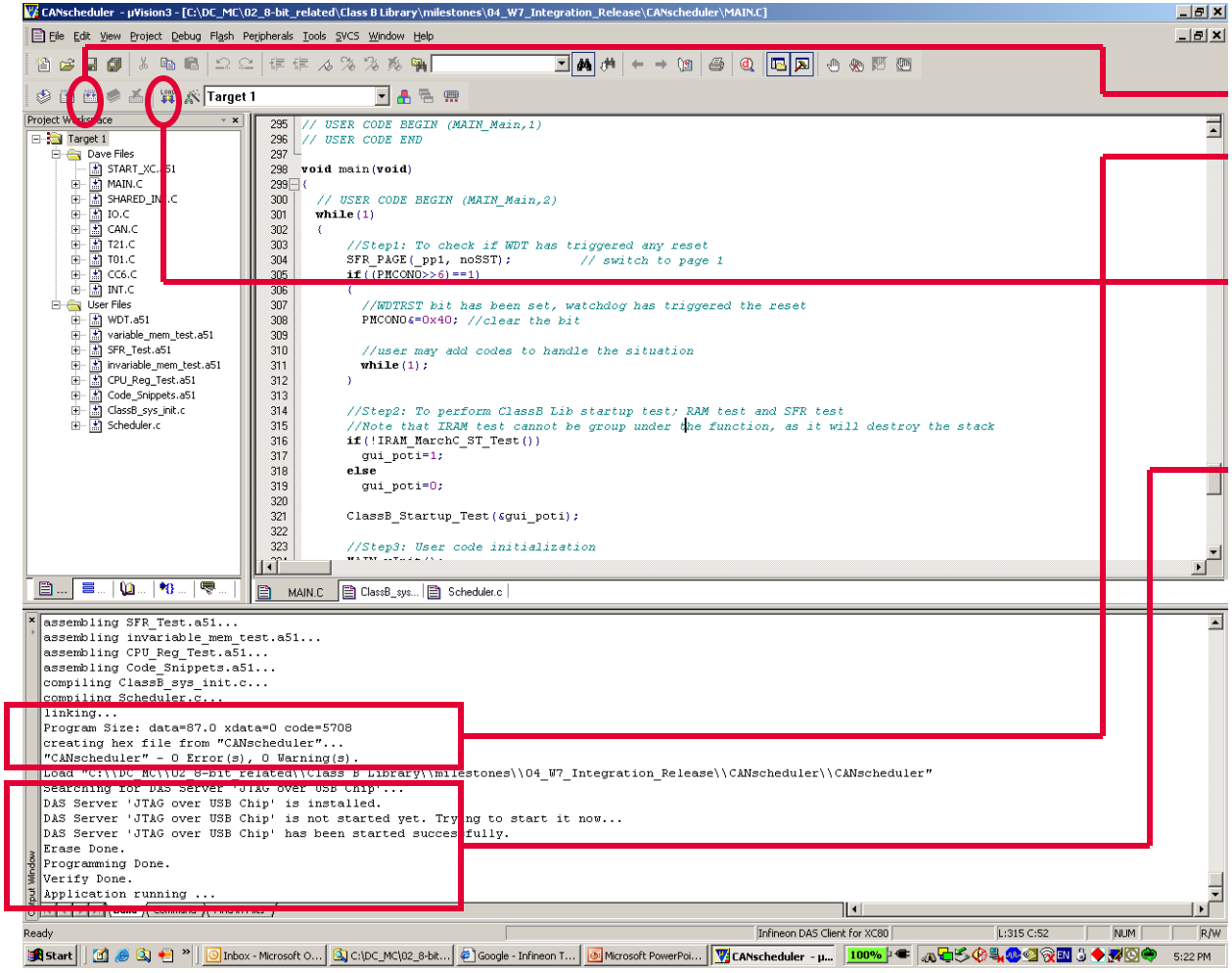
IEC60730 Class B Library

How to test the routines in XC878

Never stop thinking

Workspace compilation

- Open “CANscheduler.Uv2” with Keil uVision3 program (XC878 16Kbytes evaluation copy can be obtained from Infineon)



1. Press the compile button
2. Compilation is done successfully
3. Press the download button to flash the device
4. Download is done successfully

Copyright © Infineon Technologies 2008. All rights reserved.

Startup Tests

- At startup, all RAMs, flash ECC logic and specific peripherals will be tested
- All startup tests, except IRAM test, are grouped under

  void ClassB_Startup_Test(unsigned int disp_var):*
  - XRAM startup test using March C algorithm
  - PFlash & DFlash ECC Logic test
  - Timer0/1/2, GPIO, SSC and UART test

- IRAM and XRAM startup tests will be done using March C algorithm
  - IRAM startup test routine: *bit IRAM_MarchC_ST_Test(void)*
  - XRAM startup test routine: *bit XRAM_MarchC_ST_Test(void)*

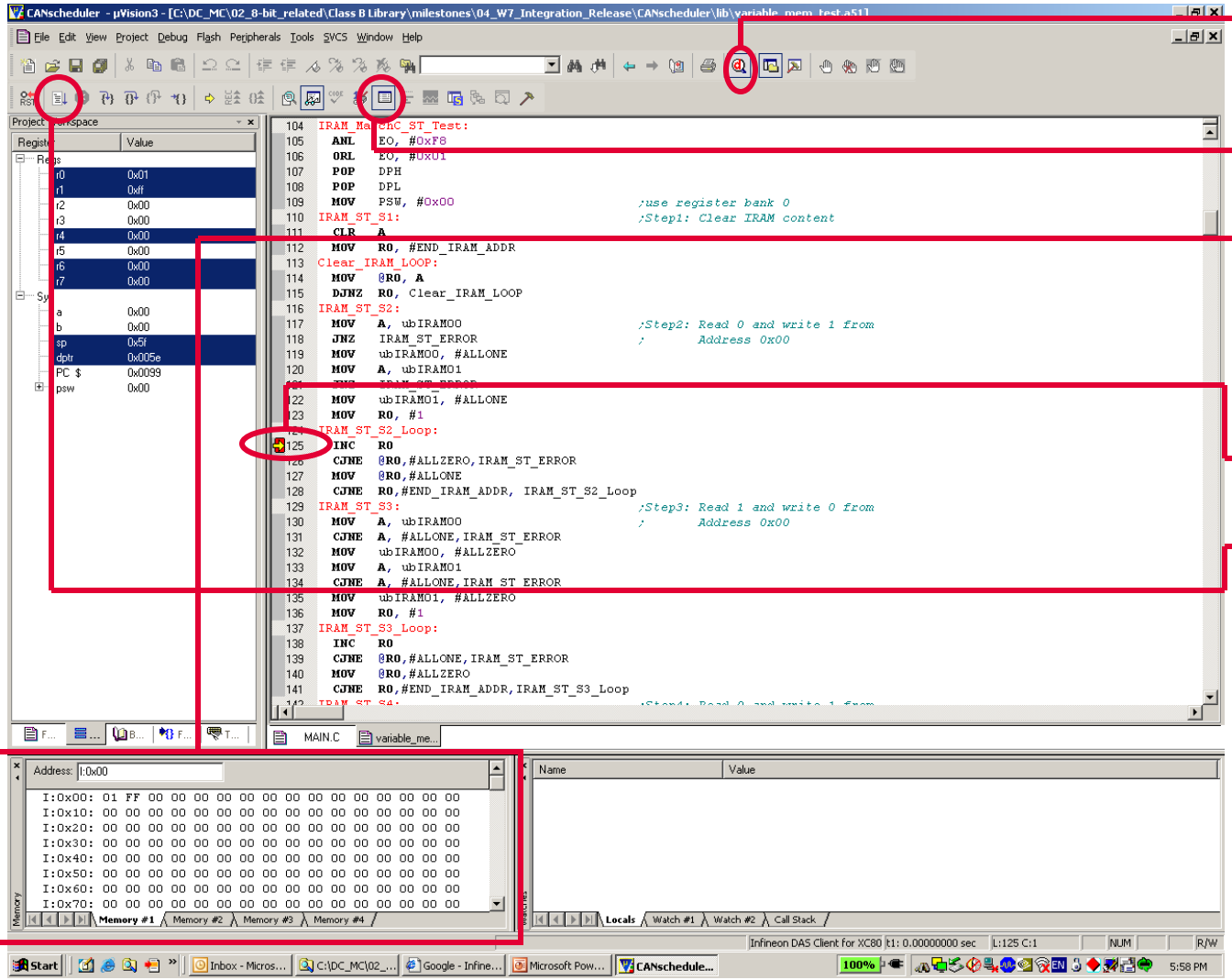
- Sequence of the startup tests shall be done in the following way:
  - IRAM test: *bit IRAM_MarchC_ST_Test(void)*
  - XRAM test
  - Peripherals test
  - Flash ECC logic test

  } *void ClassB_Startup_Test(unsigned int* disp_var)*

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C Startup Test

- To test the IRAM_MarchC_ST_Test() routine:

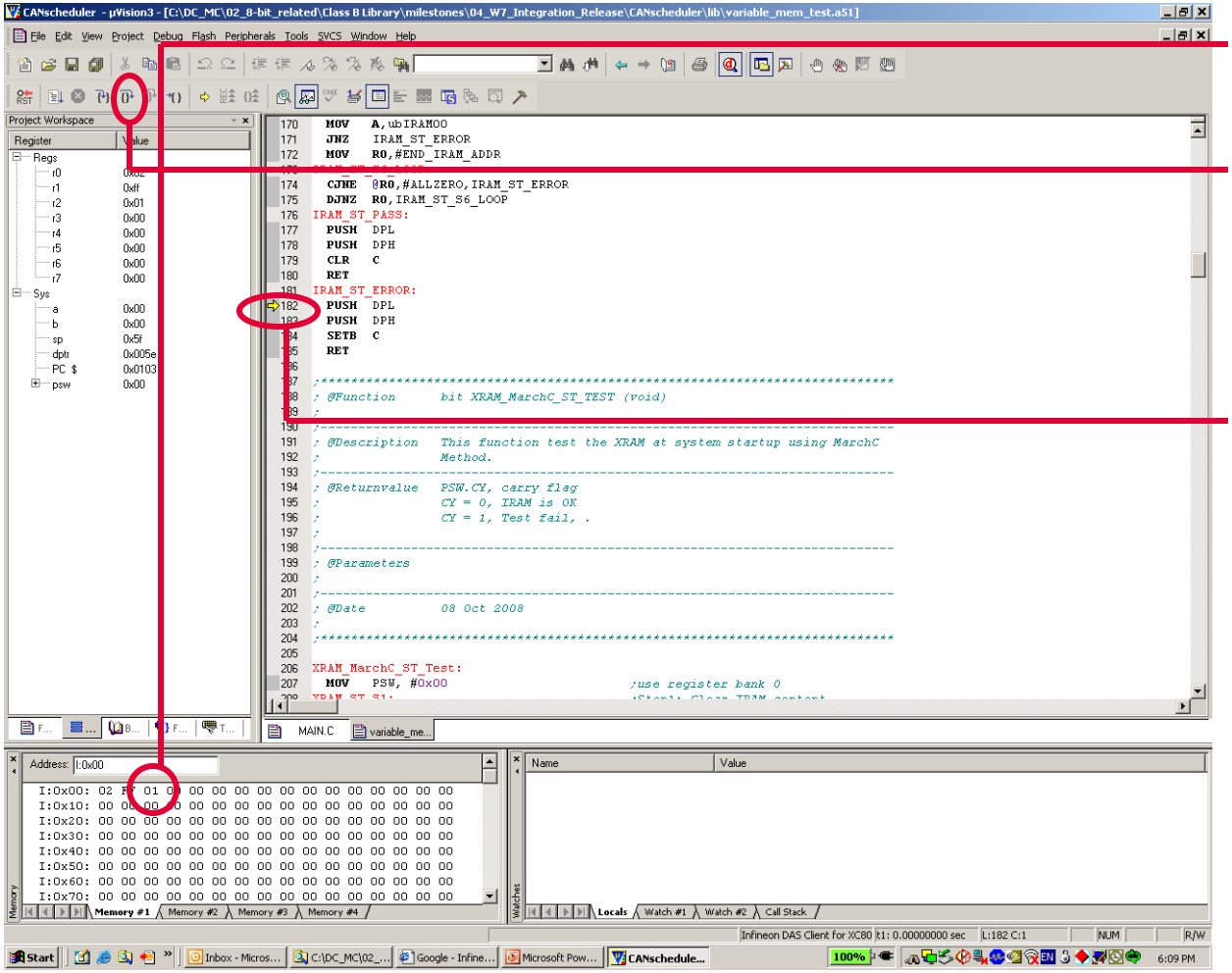


1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. Memory Window is shown for debugging purpose. Retrieve the IRAM window by typing "I:0x00" in the address column
4. Set breakpoint at line 125
5. Press "Run" button, the program shall stop at line 125 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C Startup Test

- At line 125 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error.

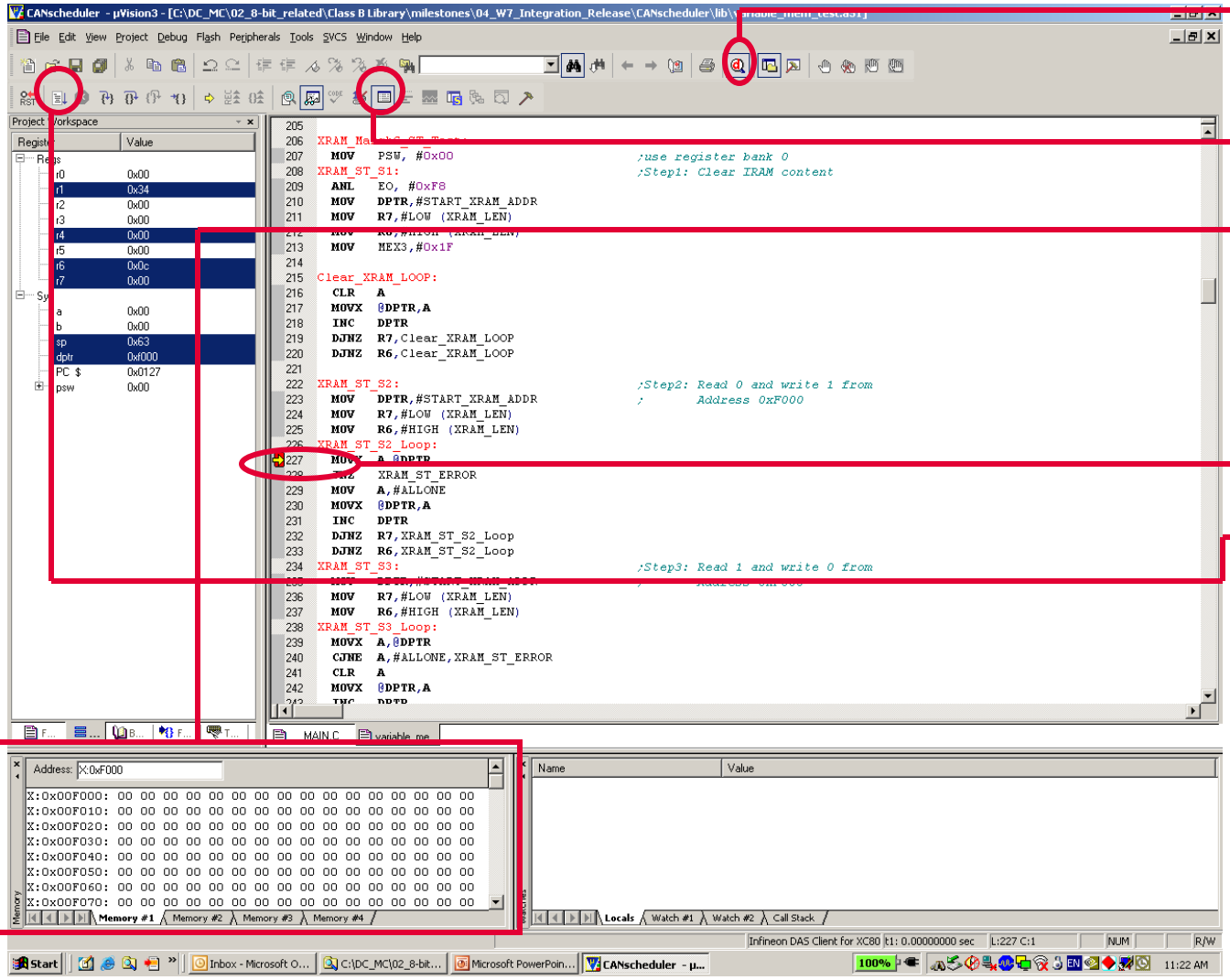


1. At address 0x2, double click the location and change the value to '1' as shown
2. Step through the code by clicking on the "Step Over" button. At line 126, it will jump to line 182 due to the value at address 0x2 is not '0' as expected
3. Line 182 onwards will push the return address back to stack, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C Startup Test

- To test XRAM_MarchC_ST_Test() routine:

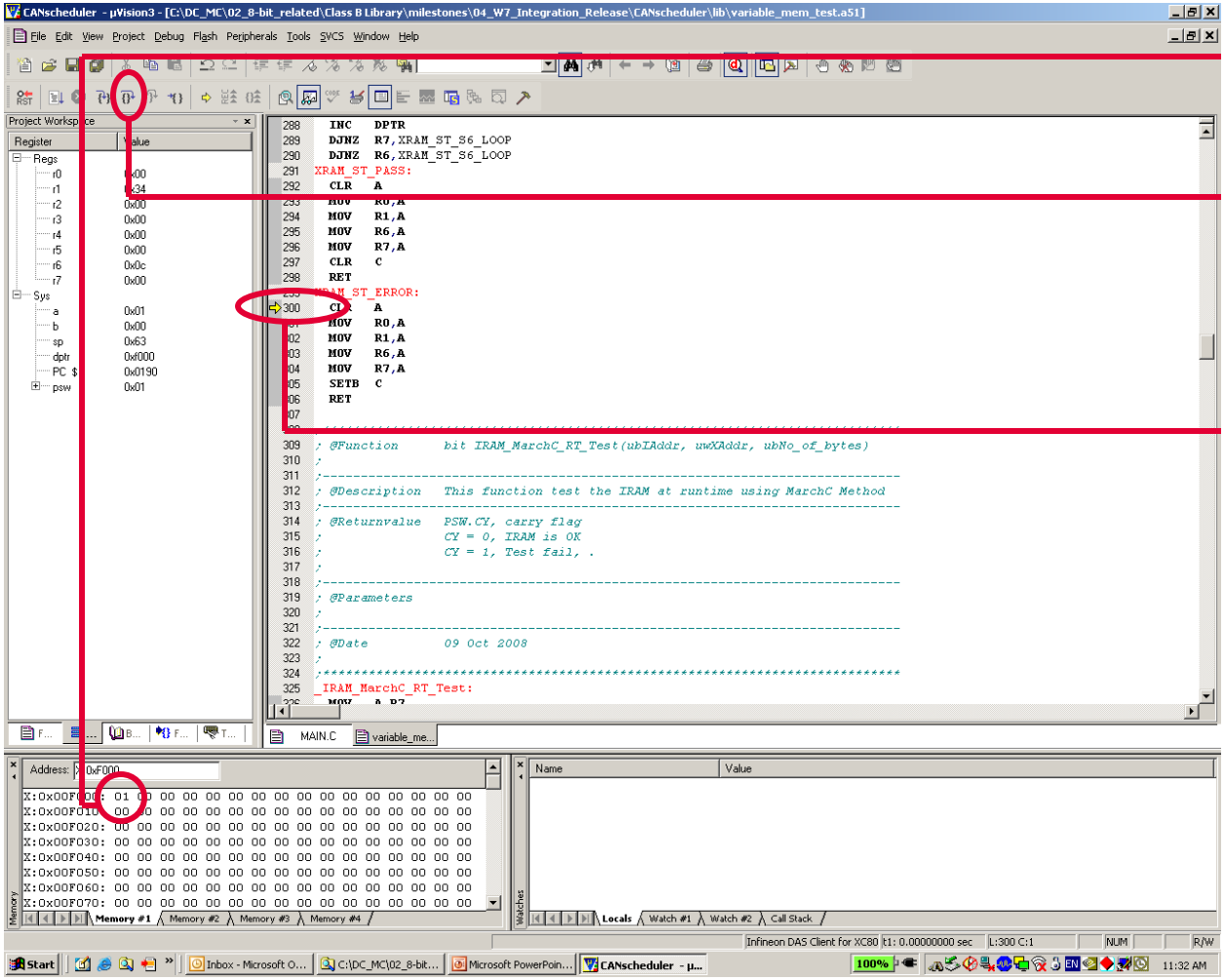


1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. Memory Window is shown for debugging purpose. Retrieve the XRAM window by typing "X:0xF000" in the address column
4. Set breakpoint at line 227
5. Press "Run" button, the program shall stop at line 227 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C Startup Test

- At line 227 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error and clear the registers value



1. At address 0xF000, double click the location and change the value to '1' as shown
2. Step through the code by clicking on the "Step Over" button. At line 228, it will jump to line 300 due to the value at address 0xF000 is not '0' as expected
3. Line 300 onwards will clear the registers, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

Flash ECC Logic Tests

- Both PFlash and DFlash ECC logic can be tested using the following routines:
  - PFlash: *bit PFlash_ECC_Logic_Test(uword uwAddr)*
  - DFlash: *bit DFlash_ECC_Logic_Test(uword uwAddr)*

- The routines can be tested for pass / fail condition without using the debugger to manipulate the routine's internal value. Details to be described in the subsequent slides.

- Both routines are incorporated into *ClassB_Startup_Test(void)*, which in turn will be called at the beginning of the main()

Copyright © Infineon Technologies 2008. All rights reserved.

Flash ECC Logic Test Conditions

- Target flash address will be required as the input parameter for the routines
- To test the routines:
  - PFlash: Reading an erased Pflash memory will trigger an ECC error, therefore the target address must be an erased sector
    - Pass condition:
      - Erase the target address first
      - Call the routine *PFlash_ECC_Logic_Test(target address)*
      - The routine shall return '0'. This shows that the ECC logic is triggered
    - Fail condition:
      - Program the target address with some data
      - Call the routine *PFlash_ECC_Logic_Test(target address)*
      - The routine fail shall return '1'. This shows that the ECC logic is not triggered.

Copyright © Infineon Technologies 2008. All rights reserved.

Flash ECC Logic Test Conditions

- DFlash: The routine will write to the target memory address, ADDR and ADDR+1 with valid. Location ADDR+1 will be written twice to corrupt the data content. Reading location ADDR+1 will therefore trigger the ECC error.
  - Pass condition:
    - Erase the target address first
    - Call the routine *DFlash_ECC_Logic_Test(target address)*
    - The routine shall return '0'. This shows that the ECC logic is triggered
  - Fail condition:
    - Erase the target address first
    - Program the target address with 0x8A
    - Call the routine *DFlash_ECC_Logic_Test(target address)*
    - The routine fail shall return '1'. This shows that the ECC logic is not triggered.

Copyright © Infineon Technologies 2008. All rights reserved.

Runtime Tests

- During runtime, all RAMs, CPU registers, internal data path and address decoder logic will be tested

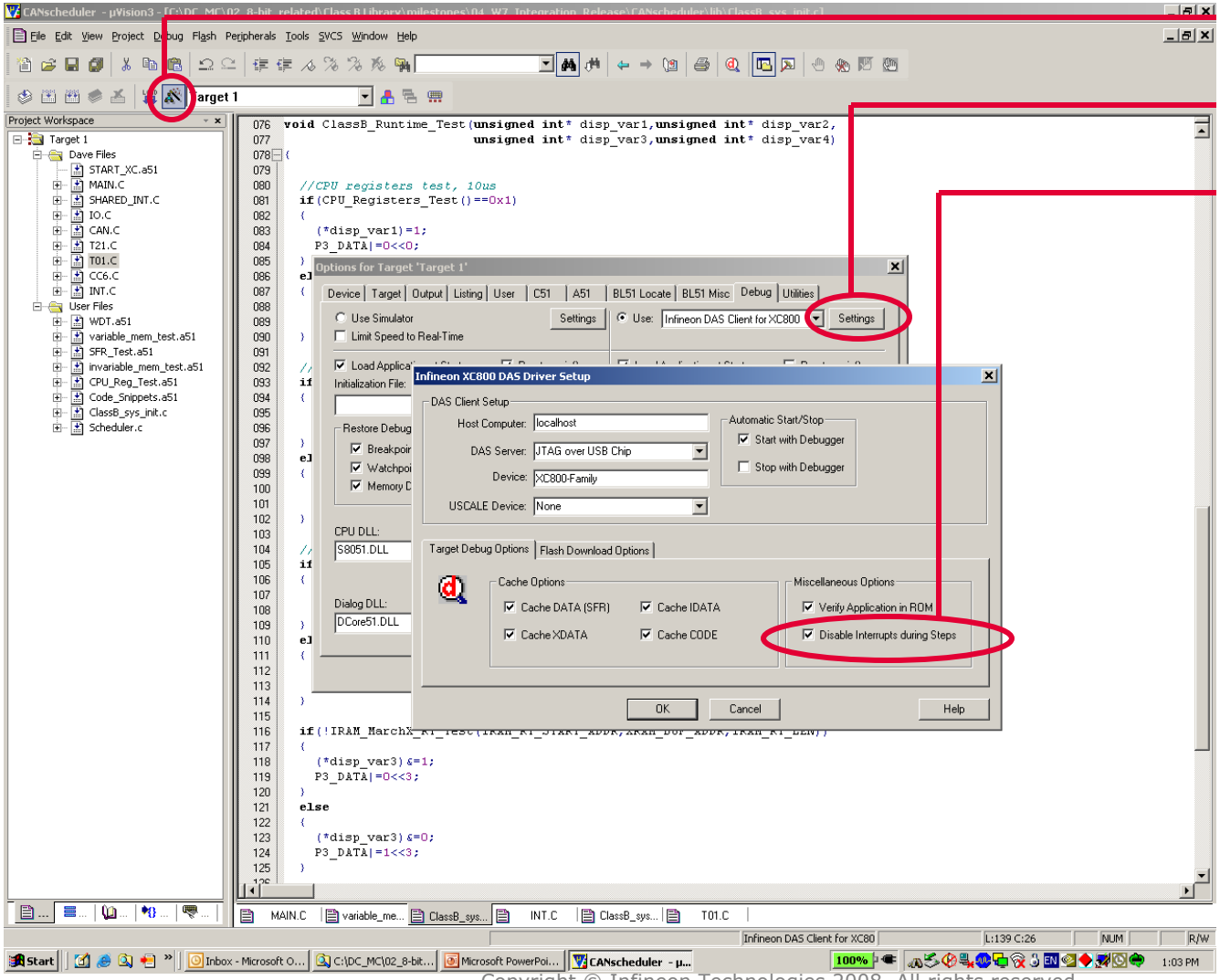
- All runtime tests are grouped under *void ClassB_Runtime_Test(….)*
  - IRAM runtime test using March C and March X algorithm
    - *bit IRAM_MarchC_RT_Test(….)*
    - *bit IRAM_MarchX_RT_Test(….)*
  - XRAM runtime test using March C and March X algorithm
    - *bit XRAM_MarchC_RT_Test(….)*
    - *bit XRAM_MarchX_RT_Test(….)*
  - CPU registers test: *ubyte CPU_Registers_Test(void)*
  - Internal data path and address decoder logic test: *bit ADDR_DATA_SA_Test(void)*

- In the CANscheduler framework, *ClassB_Runtime_Test()* will be called by Timer1 every 5ms

Copyright © Infineon Technologies 2008. All rights reserved.

Runtime Test Setup

- "Disabling the interrupt during steps" option needs to be checked in the debugger during code debugging



1. Click on "Option for Target" button
2. Click on "Settings" button
3. Put a check on "Disable Interrupts during Steps"

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C & March X Runtime Test

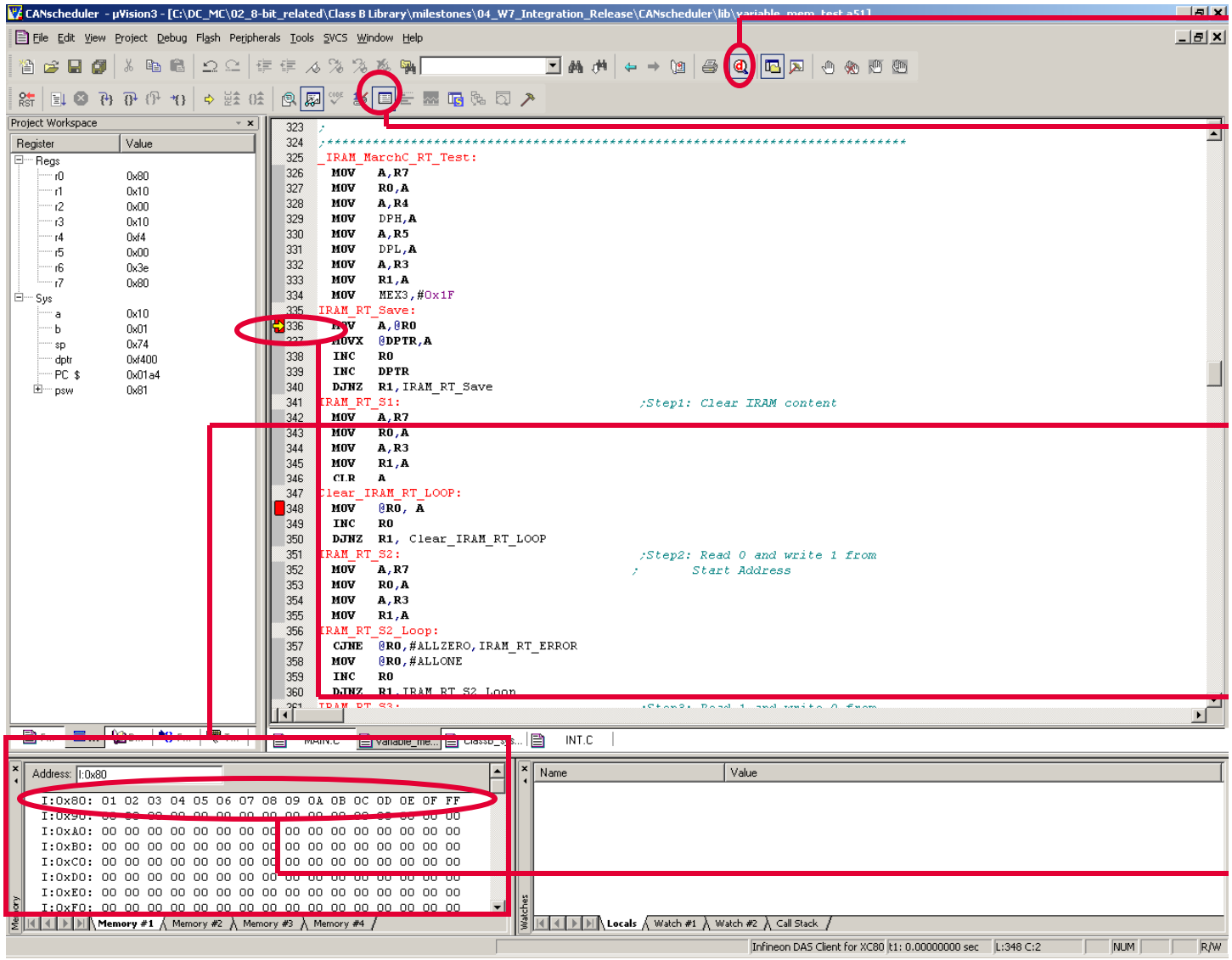
- Three input parameters are required in order to call *IRAM_MarchC_RT_Test()* and *IRAM_MarchX_RT_Test()* routine:
  - IRAM_RT_START_ADDR: IRAM area under test start address
  - XRAM_BUF_ADDR: XRAM start address to store the data of the IRAM area under test
  - IRAM_RT_LEN: Length of the IRAM area under test
  - The above pre-compile parameters can be configured in *ClassB_sys_init.h*

    Following values are configured in the CANscheduler workspace
    - *IRAM_RT_START_ADDR=0x80*
    - *XRAM_BUF_ADDR=0xF400*
    - *IRAM_RT_LEN=16*

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C Runtime Test

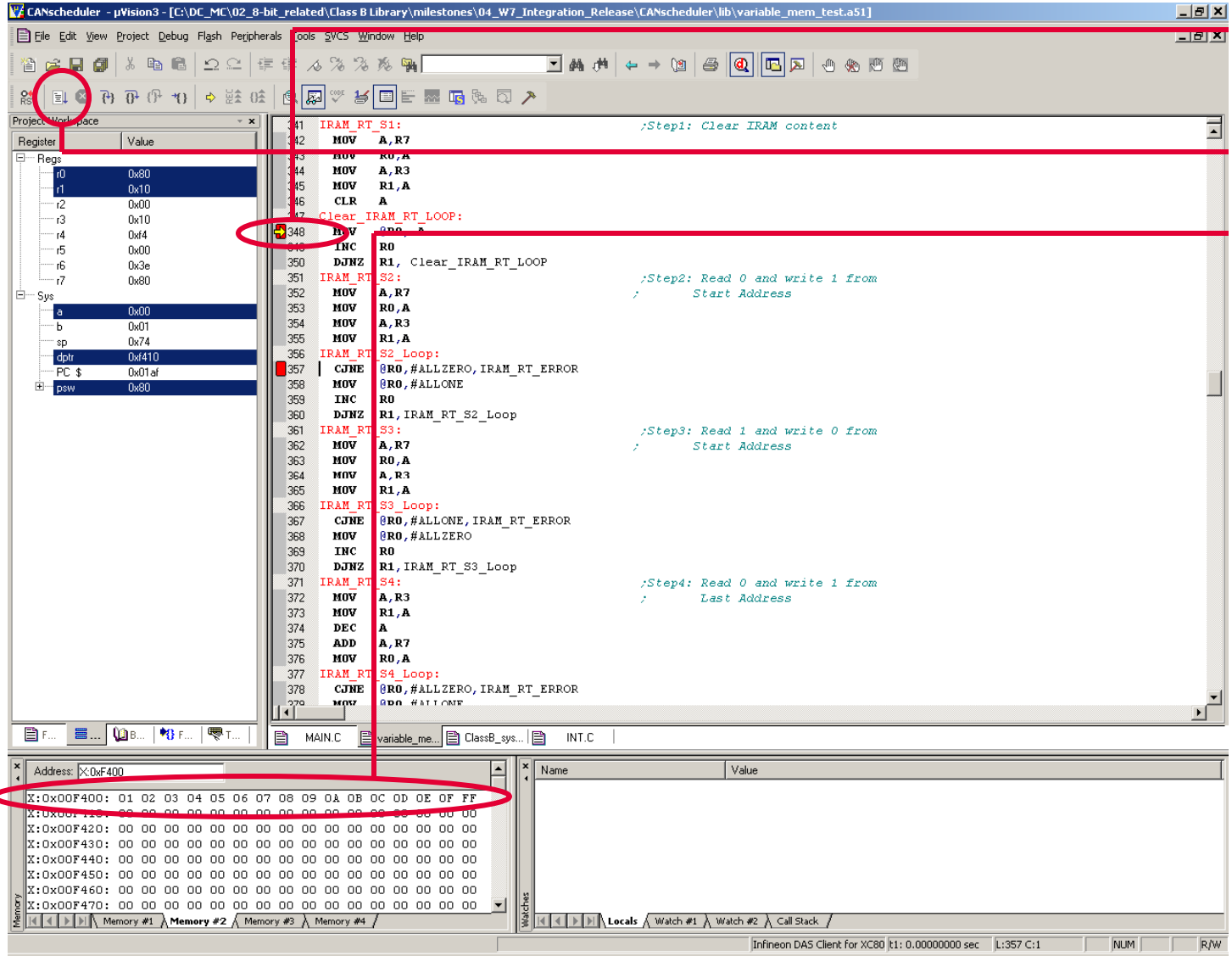
- To force an error in the *IRAM_MarchC_RT_Test()* routine:



1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. This is a non-destructive routine, therefore it will store the IRAM under test area value into designated XRAM area
4. Retrieve the IRAM window by typing "I:0x80" in the Memory #1 address column, and "X:0xF400" in the Memory #2 address column
5. Set breakpoint at line 336 & 348, then press "Run" button and the program shall stop at line 336
6. Set 16 bytes value start from address 0x80 in Memory #1 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C Runtime Test

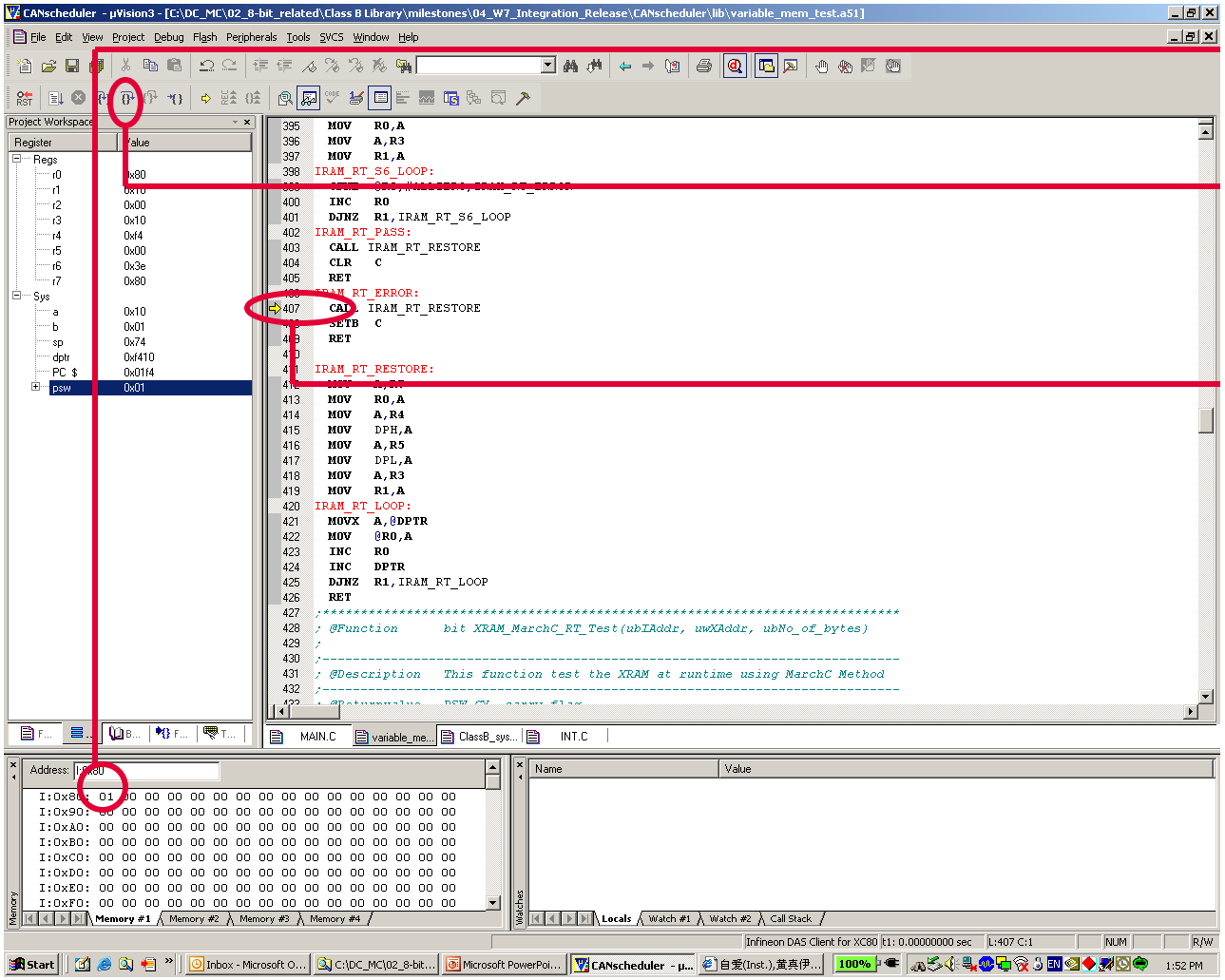


7. Remove breakpoint at line 336 and set a breakpoint at line 357
8. Press "Run" and it will stop at line 348
9. Open Memory #2 window, note that the IRAM value is now preserved in XRAM area
10. Remove breakpoint at line 348, press "Run" and it will stop at line 357

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March C Runtime Test

- At line 357 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error and restore the original value from XRAM area

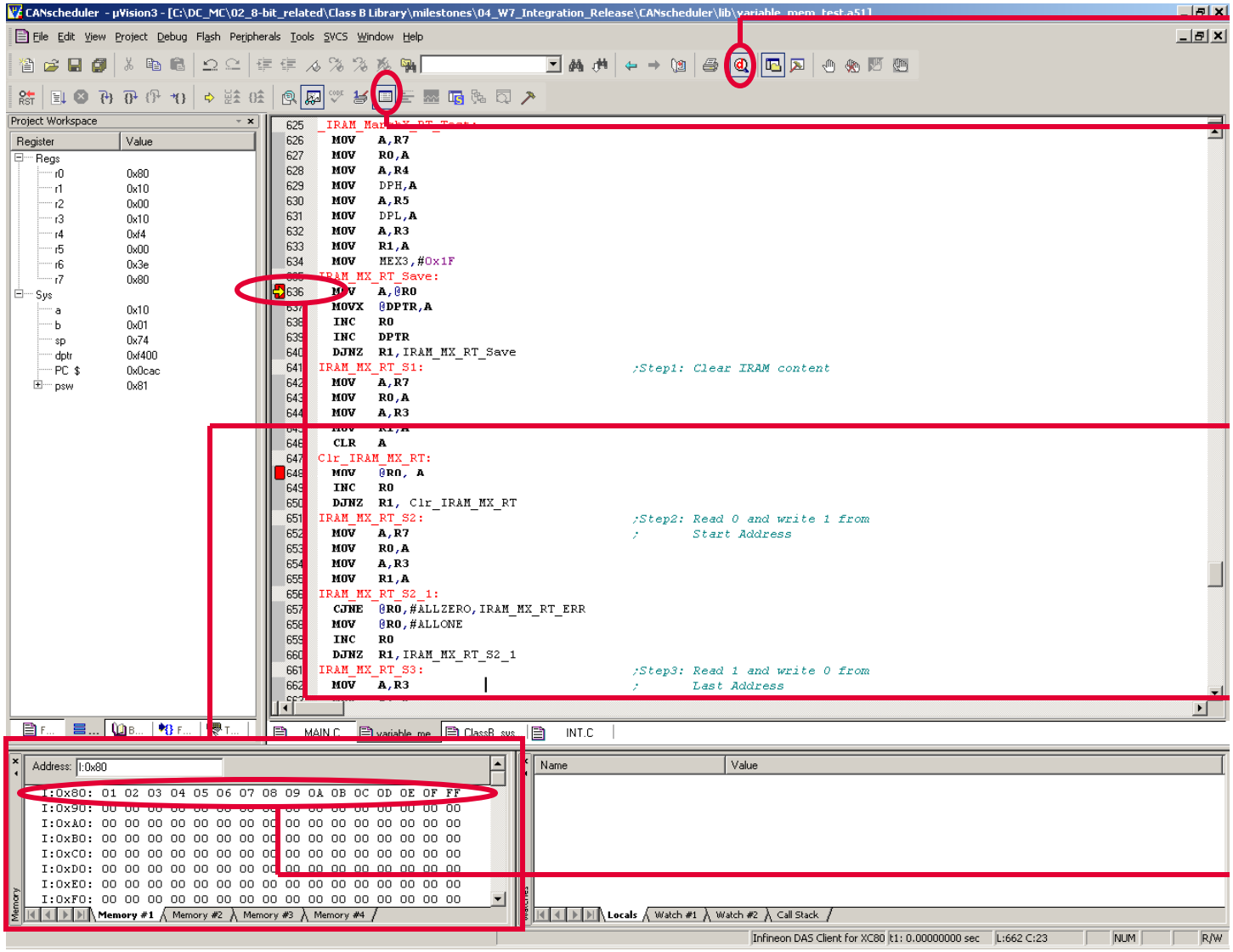


1. At address 0x80, double click the location and change the value to '1' as shown
2. Step through the code by clicking on the "Step Over" button. It will jump to line 407 due to the value at address 0x80 is not '0' as expected
3. Line 407 onwards will restore the original IRAM value, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March X Runtime Test

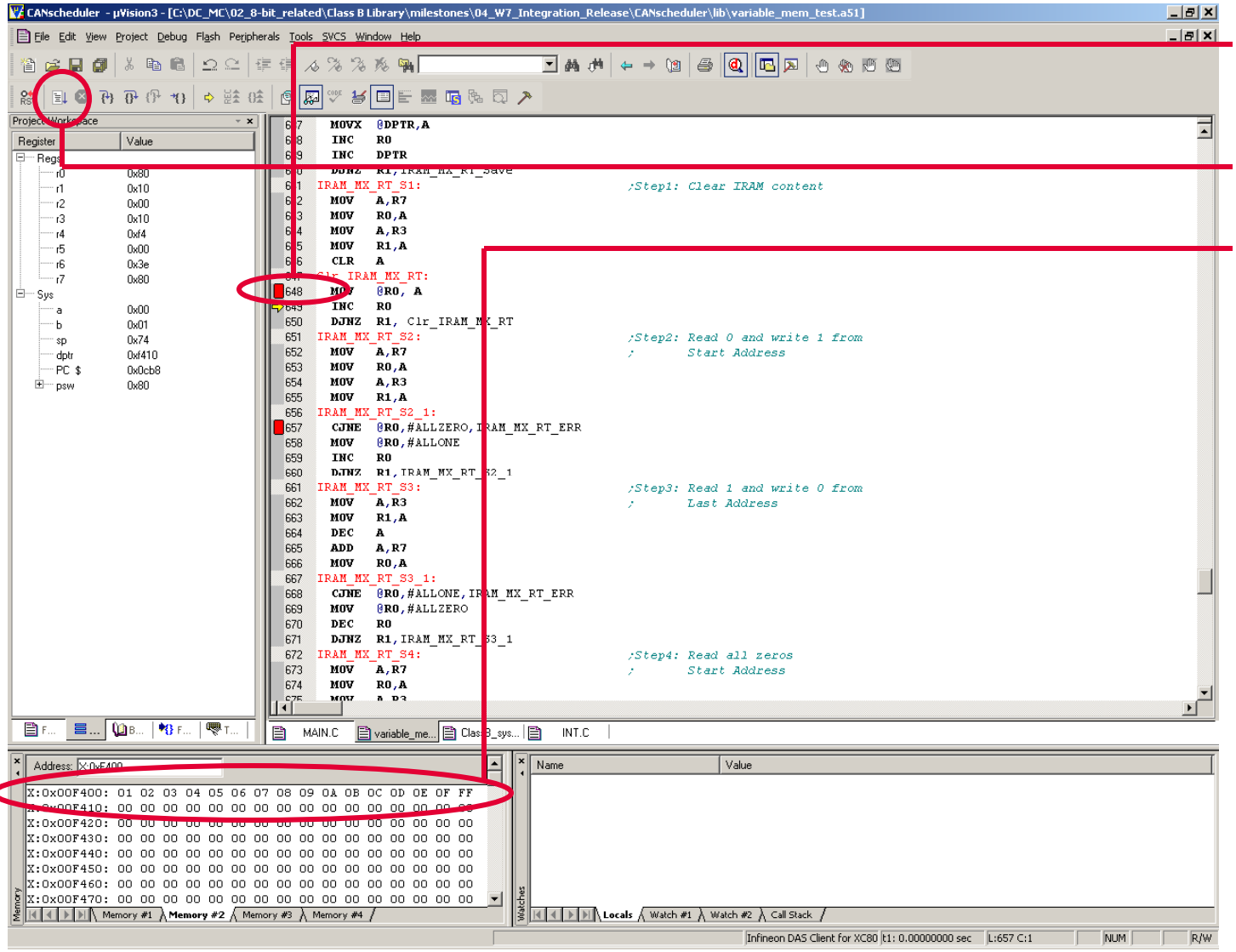
- To force an error in the *IRAM_MarchX_RT_Test()* routine:



1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. This is a non-destructive routine, therefore it will store the IRAM under test area value into designated XRAM area
4. Retrieve the IRAM window by typing "I:0x80" in the Memory #1 address column, and "X:0xF400" in the Memory #2 address column
5. Set breakpoint at line 636 & 648, then press "Run" button and the program shall stop at line 636
6. Set 16 bytes value start from address 0x80 in Memory #1 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March X Runtime Test

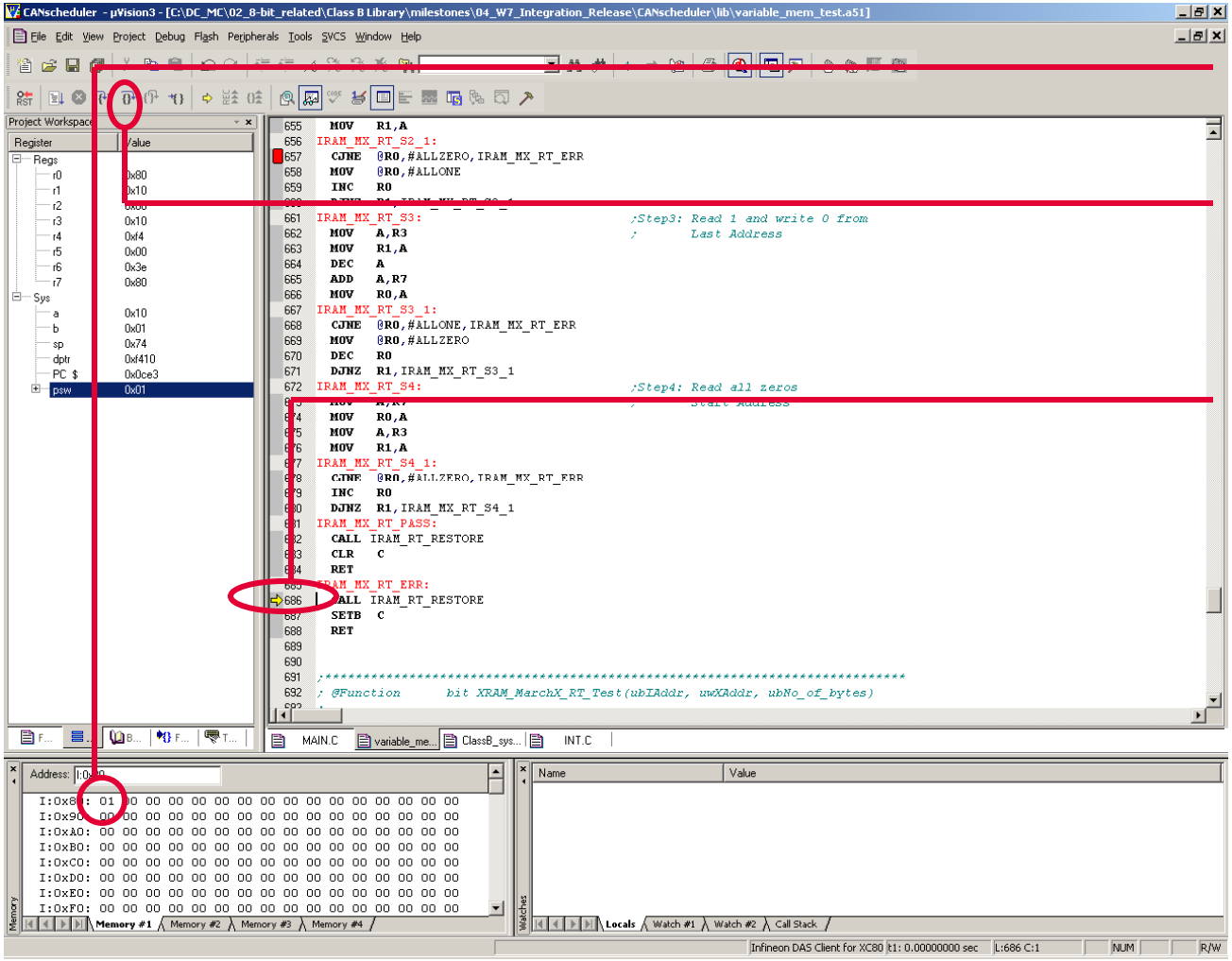


7. Remove breakpoint at line 636 and set a breakpoint at line 657
8. Press "Run" and it will stop at line 648
9. Open Memory #2 window, note that the IRAM value is now preserved in XRAM area
10. Remove breakpoint at line 648, press "Run" and it will stop at line 657

Copyright © Infineon Technologies 2008. All rights reserved.

IRAM March X Runtime Test

- At line 657 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error and restore the original value from XRAM area



11. At address 0x80, double click the location and change the value to '1' as shown
12. Step through the code by clicking on the "Step Over" button. It will jump to line 686 due to the value at address 0x80 is not '0' as expected
13. Line 686 onwards will restore the original IRAM value, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C & March X Runtime Test

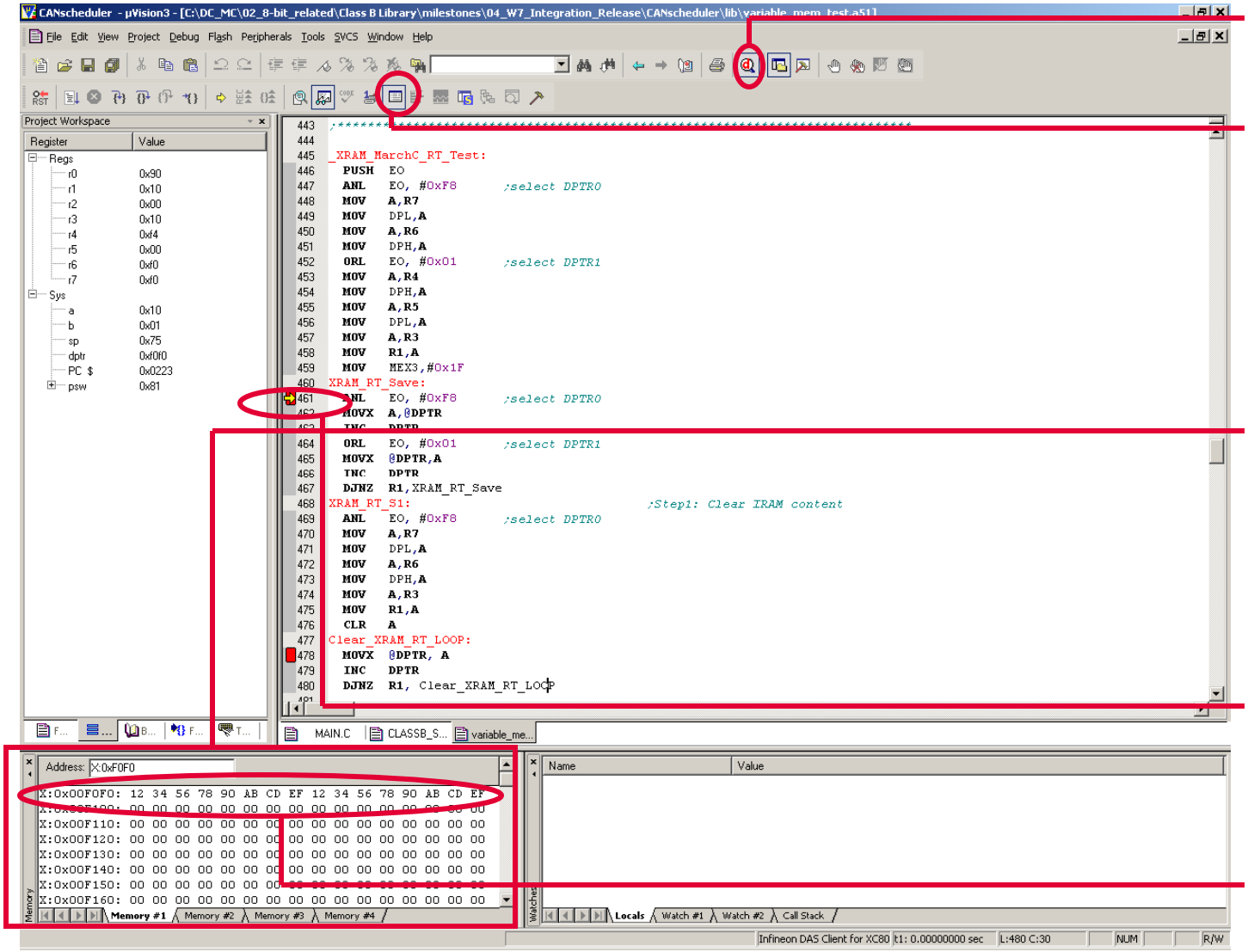
- Three input parameters are required in order to call *XRAM_MarchC_RT_Test()* and *XRAM_MarchX_RT_Test()* routine:
  - XRAM_RT_START_ADDR: XRAM are under test start address
  - XRAM_BUF_ADDR: XRAM start address to store the data of the XRAM area under test
  - XRAM_RT_LEN: Length of the XRAM area under test
  - The above pre-compile parameters can be configured in *ClassB_sys_init.h*

    Following values are configured in the CANscheduler workspace
    - *XRAM_RT_START_ADDR=0xF0F0*
    - *XRAM_BUF_ADDR=0xF400*
    - *XRAM_RT_LEN=16*

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C Runtime Test

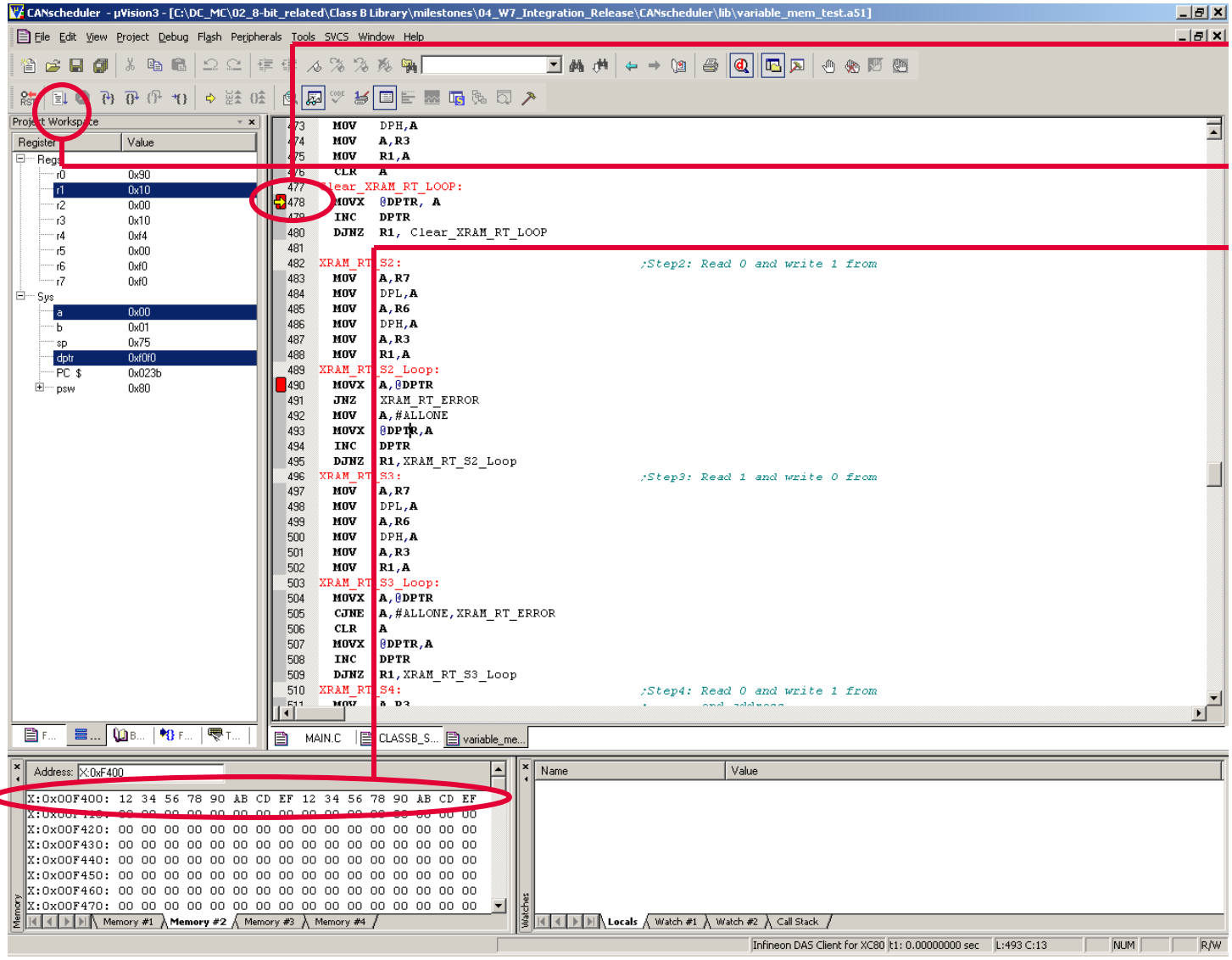
- To force an error in the *XRAM_MarchC_RT_Test()* routine:



1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. This is a non-destructive routine, therefore it will store the XRAM under test area value into designated XRAM area
4. Retrieve the XRAM window by typing "X:0xF0F0" in the Memory #1 address column, and "X:0xF400" in the Memory #2 address column
5. Set breakpoint at line 461 & 478, then press "Run" button and the program shall stop at line 461
6. Set 16 bytes value start from address 0xF0F0 in Memory #1 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C Runtime Test

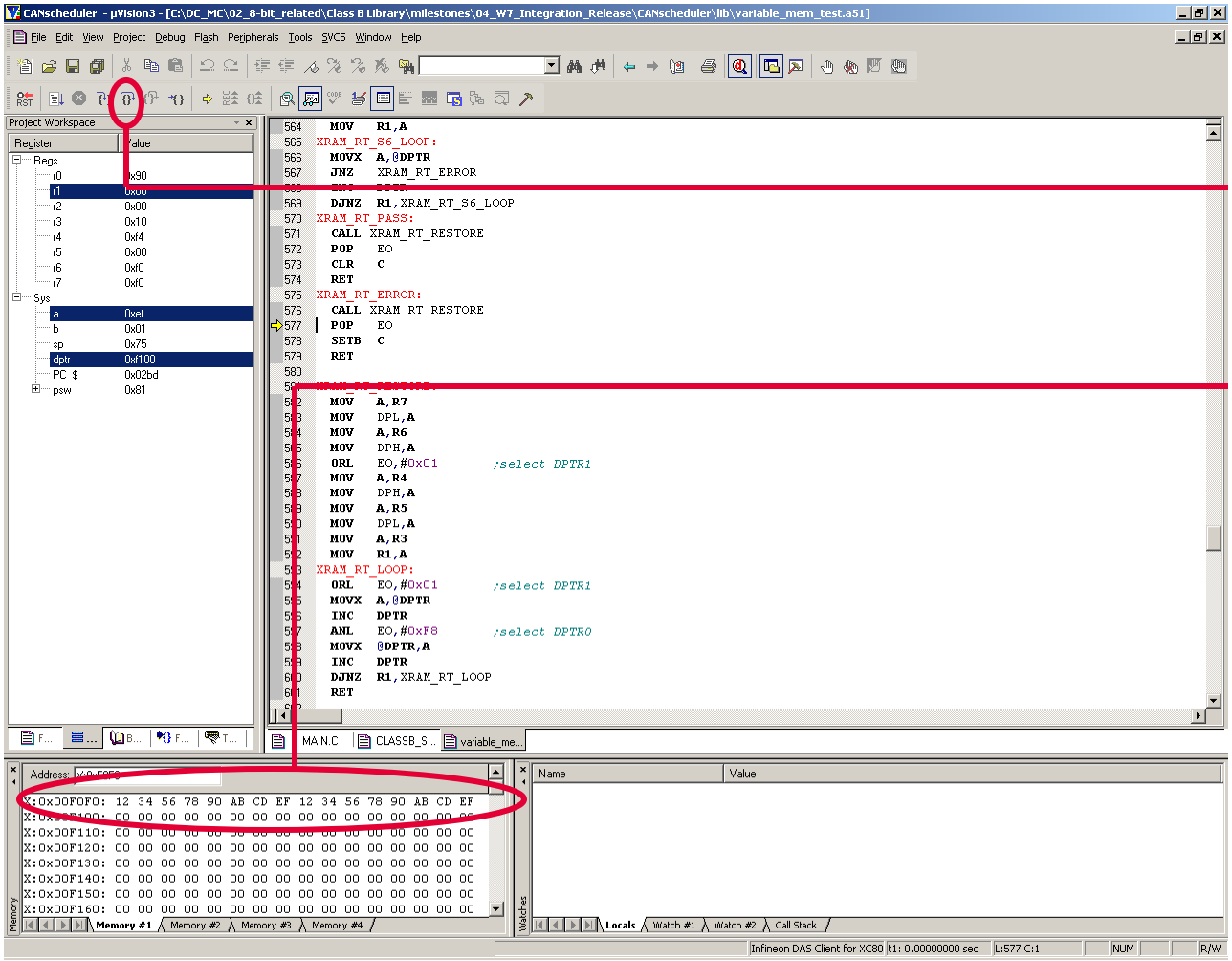


7. Remove breakpoint at line 461 and set a breakpoint at line 490
8. Press "Run" and it will stop at line 478
9. Open Memory #2 window, note that the IRAM value is now preserved in XRAM area
10. Remove breakpoint at line 478, press "Run" and it will stop at line 490

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March C Runtime Test

- At line 490 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error and restore the original value from XRAM area

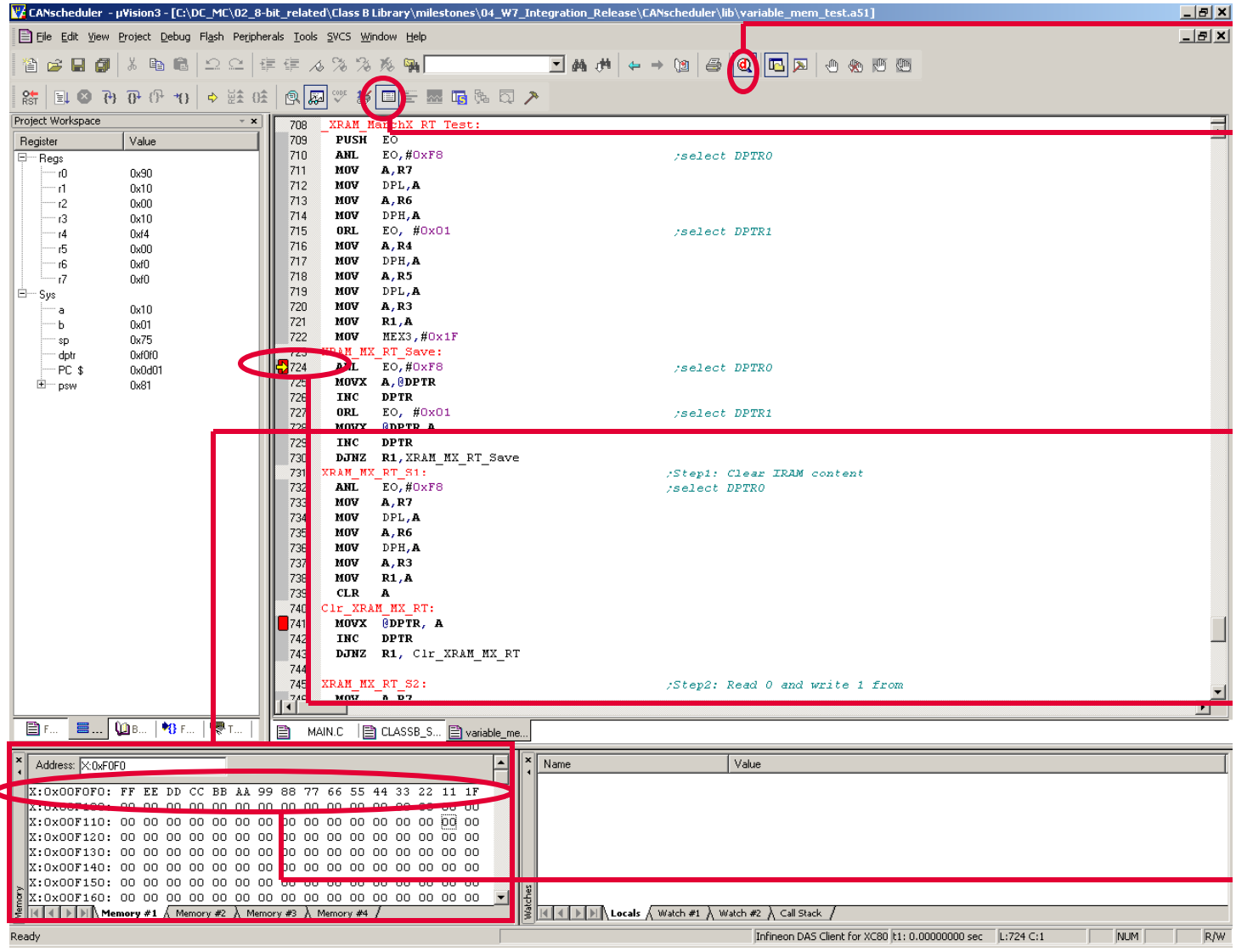


11. At address 0xF0F0, double click the location and change the value to '1'
12. Step through the code by clicking on the "Step Over" button. It will jump to line 576 due to the value at address 0xF0F0 is not '0' as expected
13. Line 576 onwards will restore the original XRAM value as shown, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March X Runtime Test

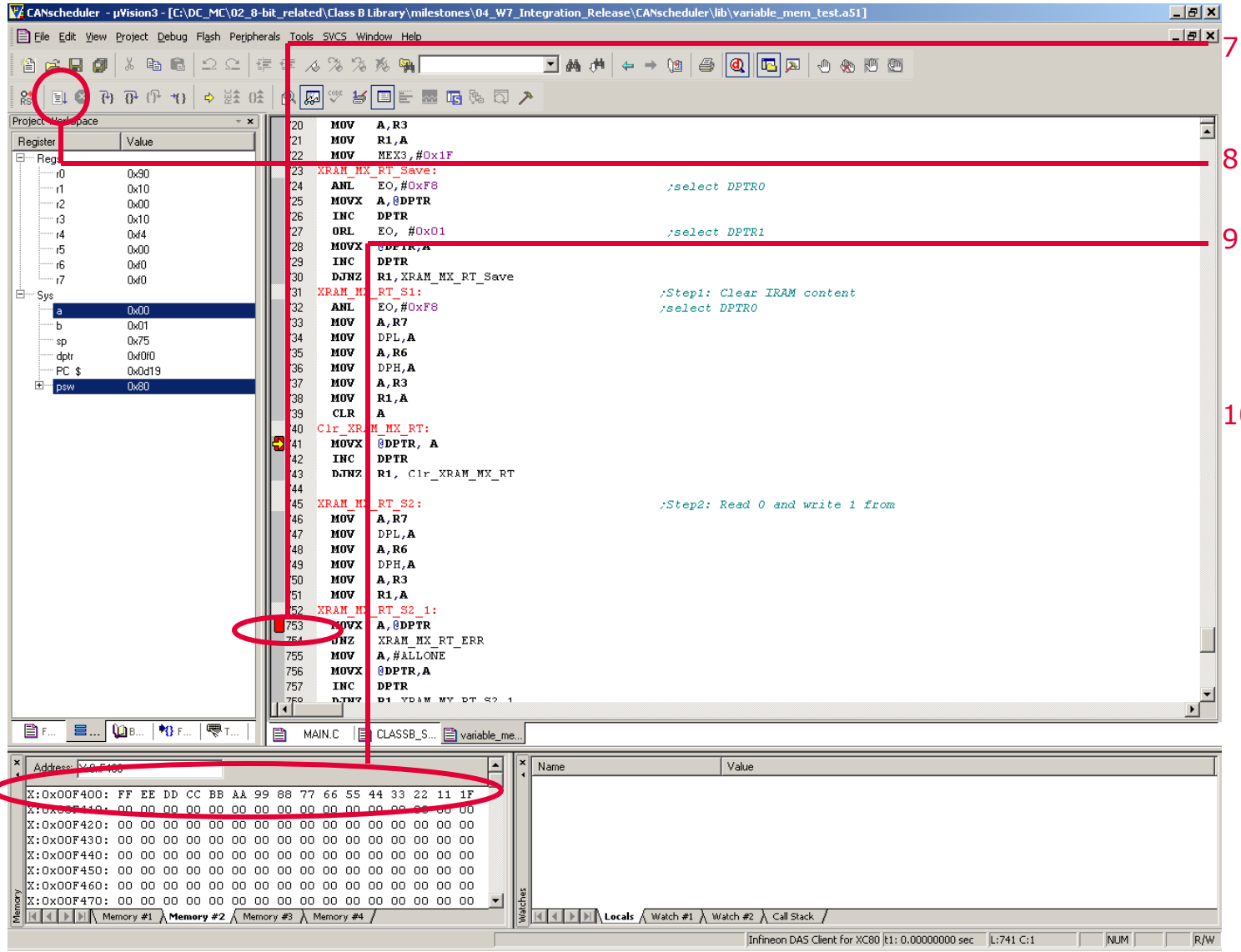
- To force an error in the *XRAM_MarchX_RT_Test()* routine:



1. Press the "Debug" button to go into debug mode
2. Turn on the "Memory Window" button.
3. This is a non-destructive routine, therefore it will store the XRAM under test area value into designated XRAM area
4. Retrieve the XRAM window by typing "X:0xF0F0" in the Memory #1 address column, and "X:0xF400" in the Memory #2 address column
5. Set breakpoint at line 724 & 741, then press "Run" button and the program shall stop at line 724
6. Set 16 bytes value start from address 0xF0F0 in Memory #1 as shown

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March X Runtime Test

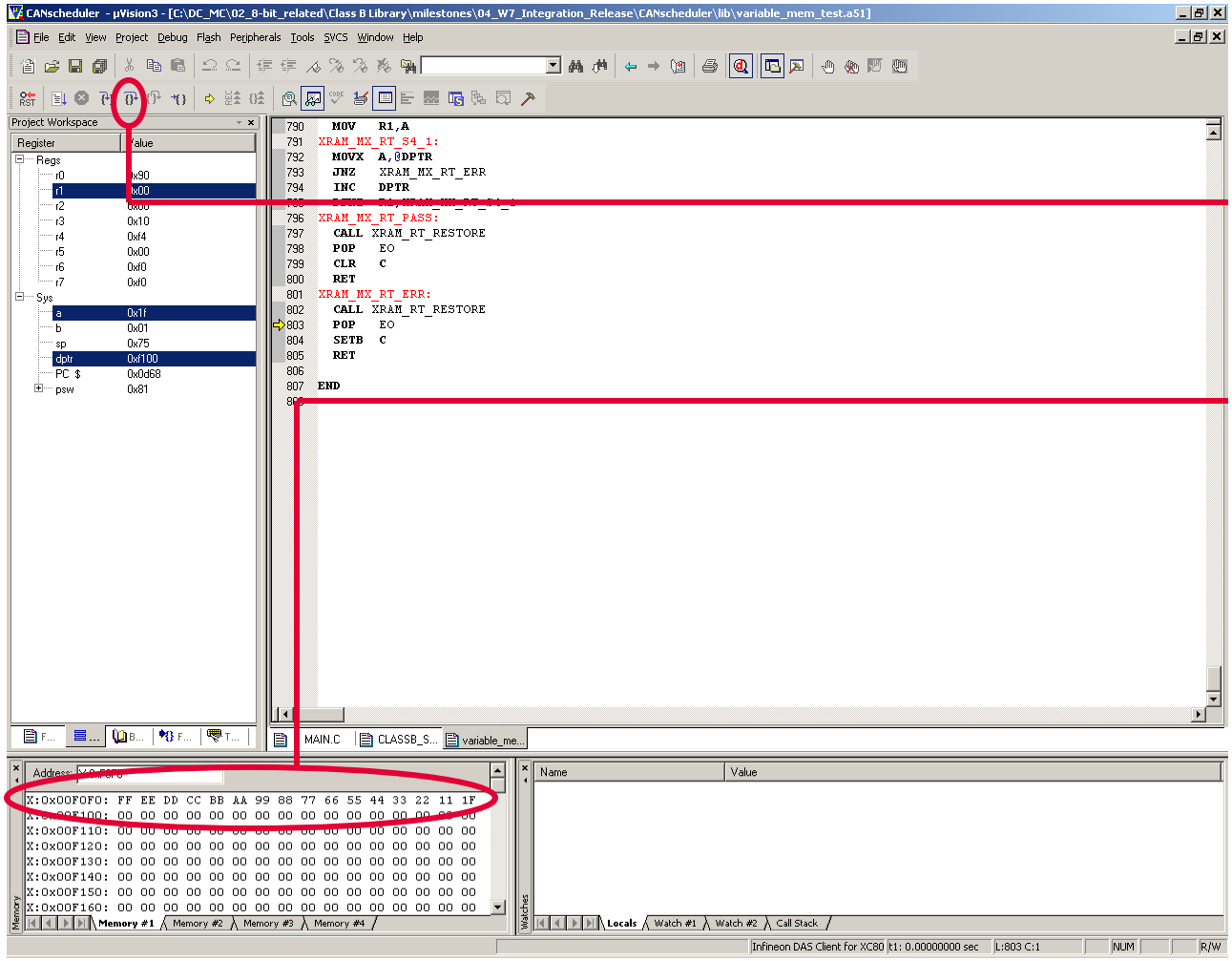


7. Remove breakpoint at line 724 and set a breakpoint at line 753
8. Press "Run" and it will stop at line 741
9. Open Memory #2 window, note that the XRAM value is now preserved in XRAM area
10. Remove breakpoint at line 741, press "Run" and it will stop at line 753

Copyright © Infineon Technologies 2008. All rights reserved.

XRAM March X Runtime Test

- At line 753 onwards, it is expected to read '0' from the start address. If the return value is as expected, it will write '1' and proceed with next location. Otherwise, it will return as error and restore the original value from XRAM area

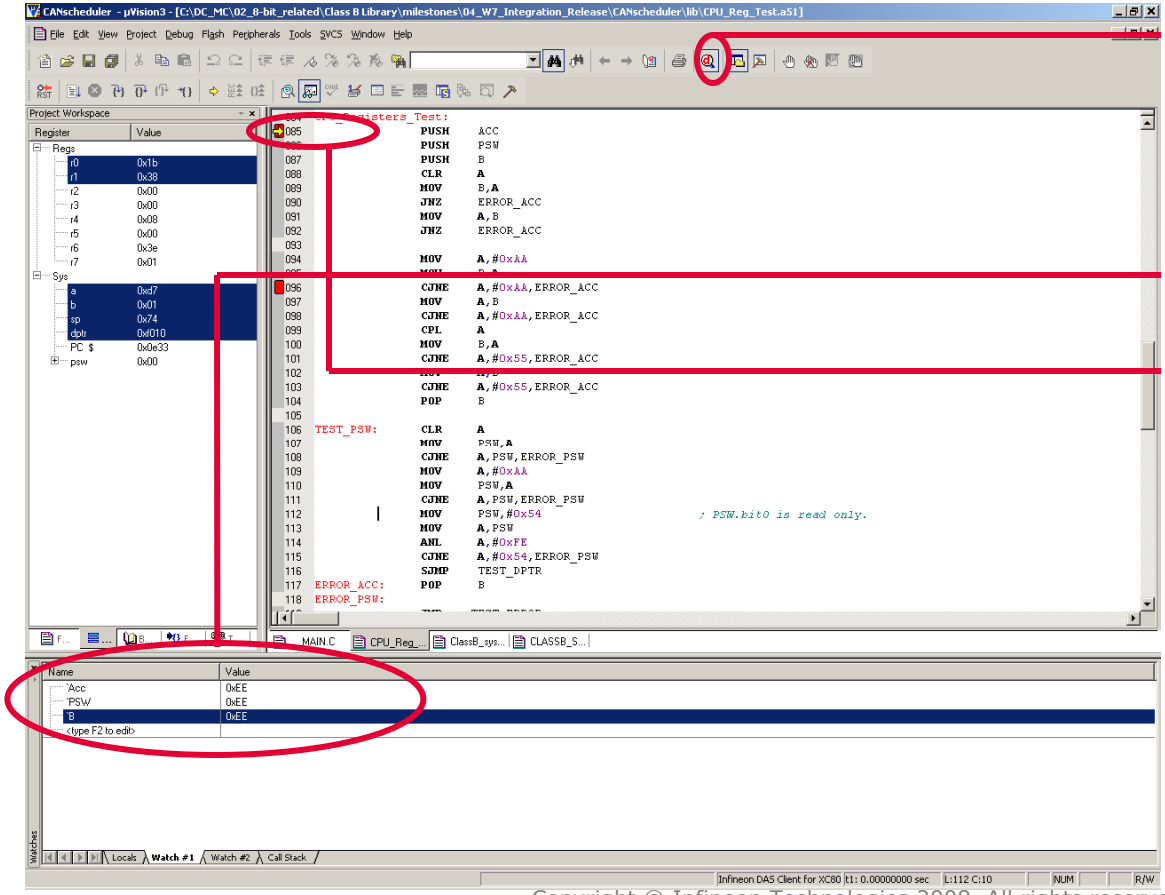


11. At address 0xF0F0, double click the location and change the value to '1'
12. Step through the code by clicking on the "Step Over" button. It will jump to line 802 due to the value at address 0xF0F0 is not '0' as expected
13. Line 802 onwards will restore the original XRAM value as shown, set the carry bit and return to the main application

Copyright © Infineon Technologies 2008. All rights reserved.

CPU Registers Test

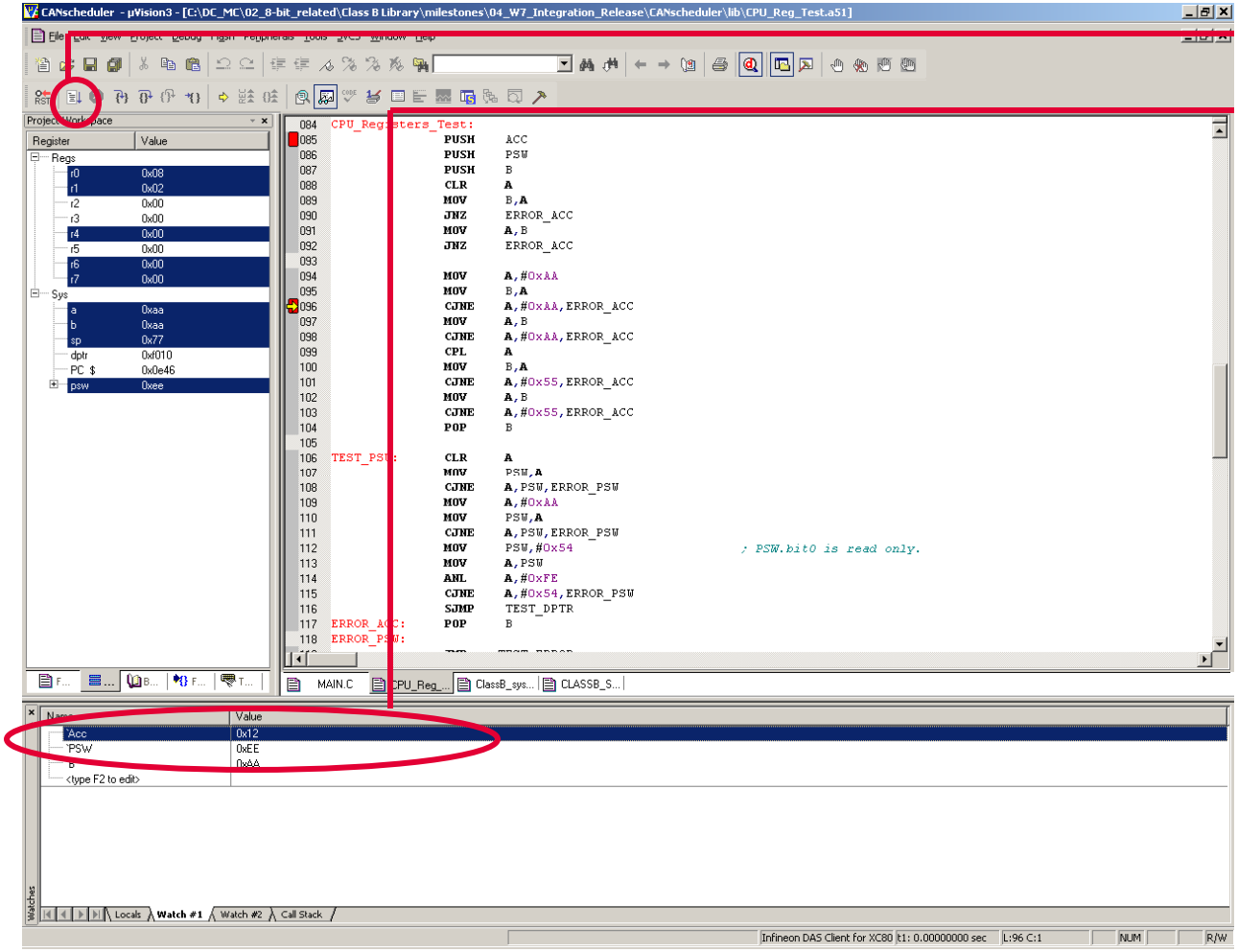
- CPU register Acc, B, DPTR and PSW will be tested by the routine *CPU_Registers_Test()*
- The routine is non-destructive, i.e. the values in the registers will be restored after the test
- To test register Acc:



1. Press the "Debug" button to go into debug mode
2. This is a non-destructive routine, therefore it will push the register Acc value into the stack before testing
3. Set breakpoint at line 85 & 96, then press "Run" button and the program shall stop at line 85
4. In Watch #1 window, add registers Acc, B and PSW with value 0xEE

Copyright © Infineon Technologies 2008. All rights reserved.

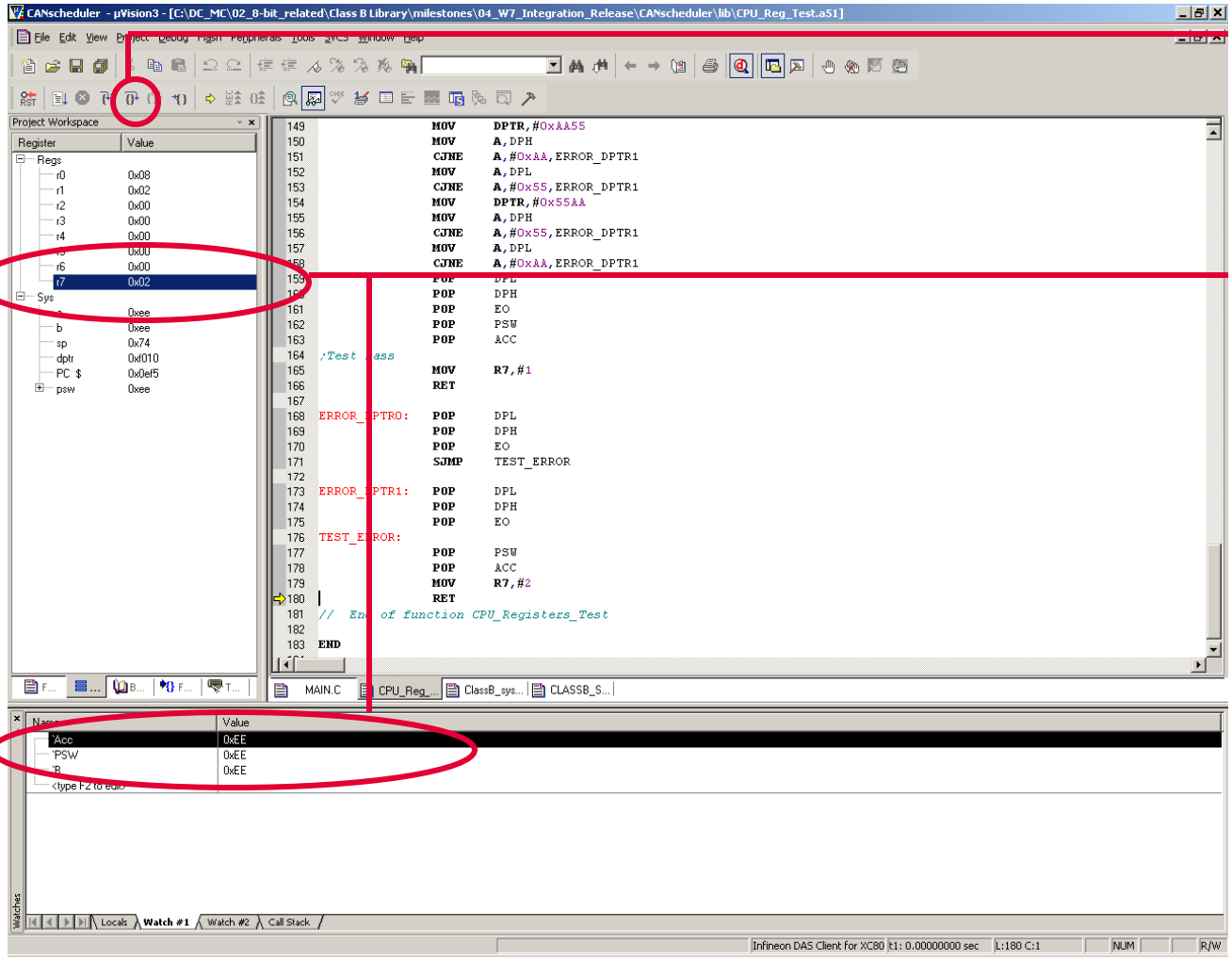
CPU Registers Test –Register Acc



5. Press “Run” button, it will stop at line 96
6. In Watch #1 window, change Acc value from 0xAA to 0x12

Copyright © Infineon Technologies 2008. All rights reserved.

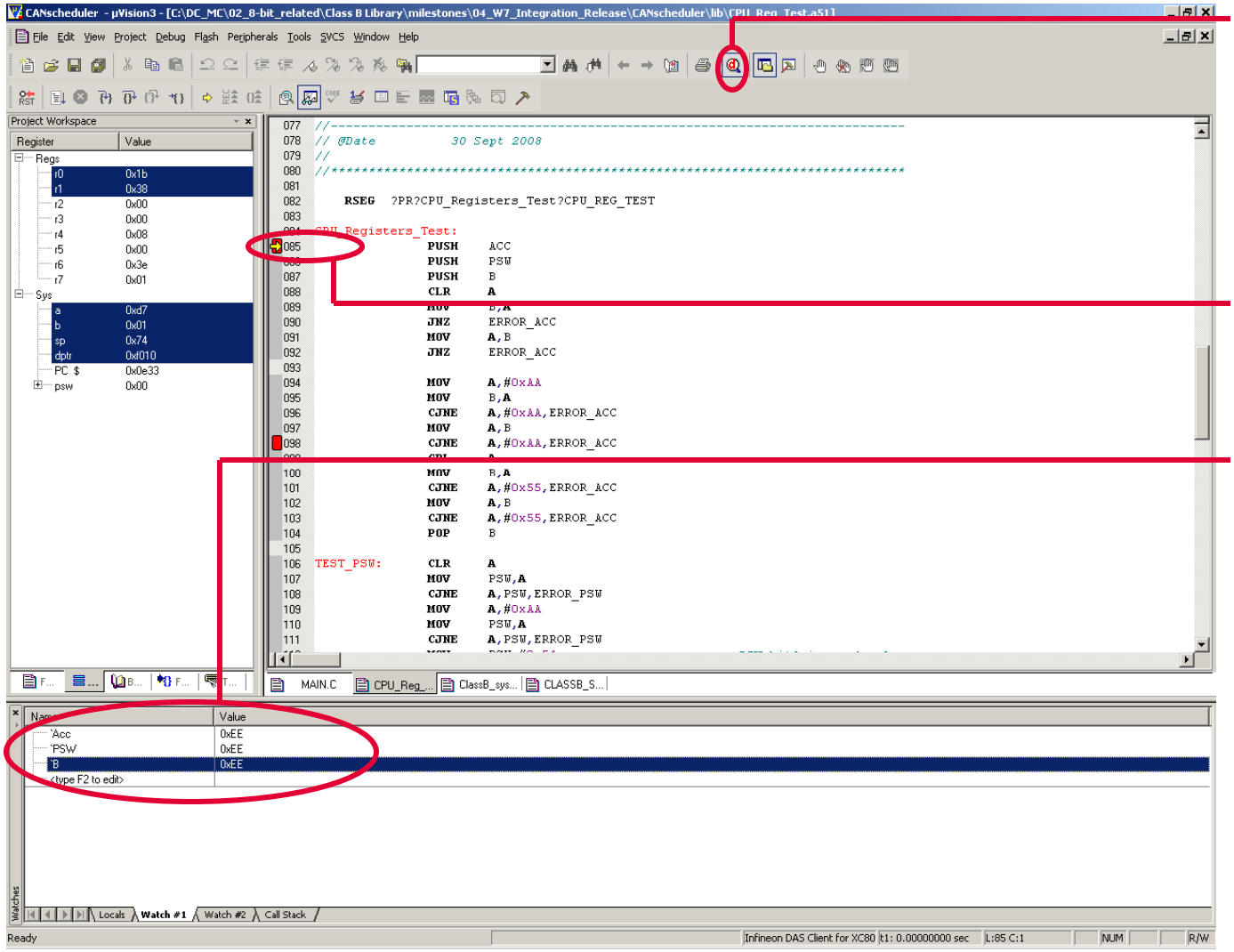
CPU Registers Test –Register Acc



7. Press “Step Over” button, eventually it will come to line 180 as the expected value in register Acc is not 0xAA and it will jump into error branch
8. Note that in Watch #1 window, registers Acc, B and PSW are restored to the original value 0xEE. The function will return 0x2 as shown in register R7

Copyright © Infineon Technologies 2008. All rights reserved.

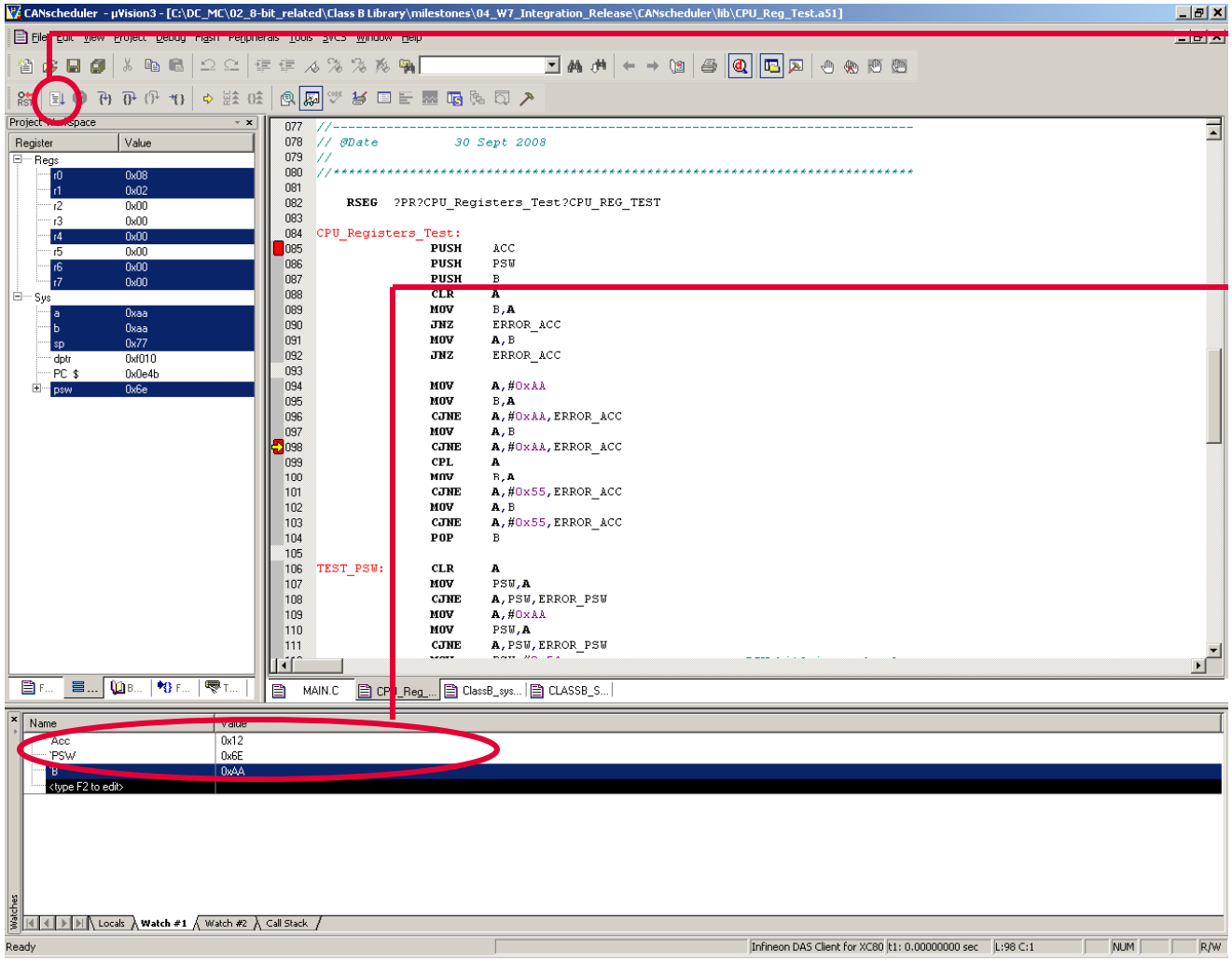
CPU Registers Test –Register B



1. Press the "Debug" button to go into debug mode
2. This is a non-destructive routine, therefore it will push the register B value into the stack before testing
3. Set breakpoint at line 85 & 98, then press "Run" button and the program shall stop at line 85
4. In Watch #1 window, add registers Acc, B and PSW with value 0xEE

Copyright © Infineon Technologies 2008. All rights reserved.

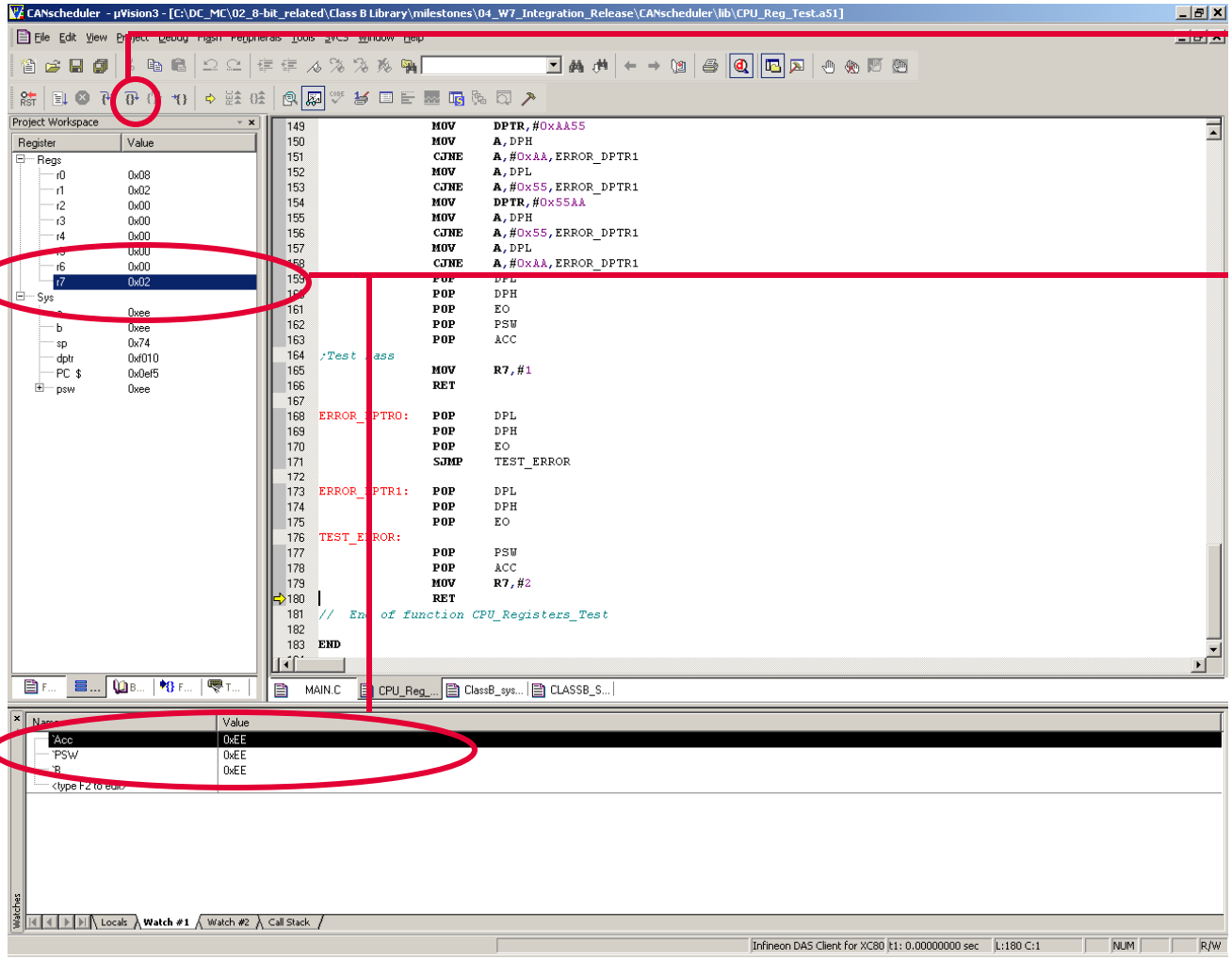
CPU Registers Test –Register B



5. Press “Run” button, it will stop at line 98
6. Register B value will be copied to Acc first, before the comparison can be made
7. In Watch #1 window, change Acc value from 0xAA to 0x12

Copyright © Infineon Technologies 2008. All rights reserved.

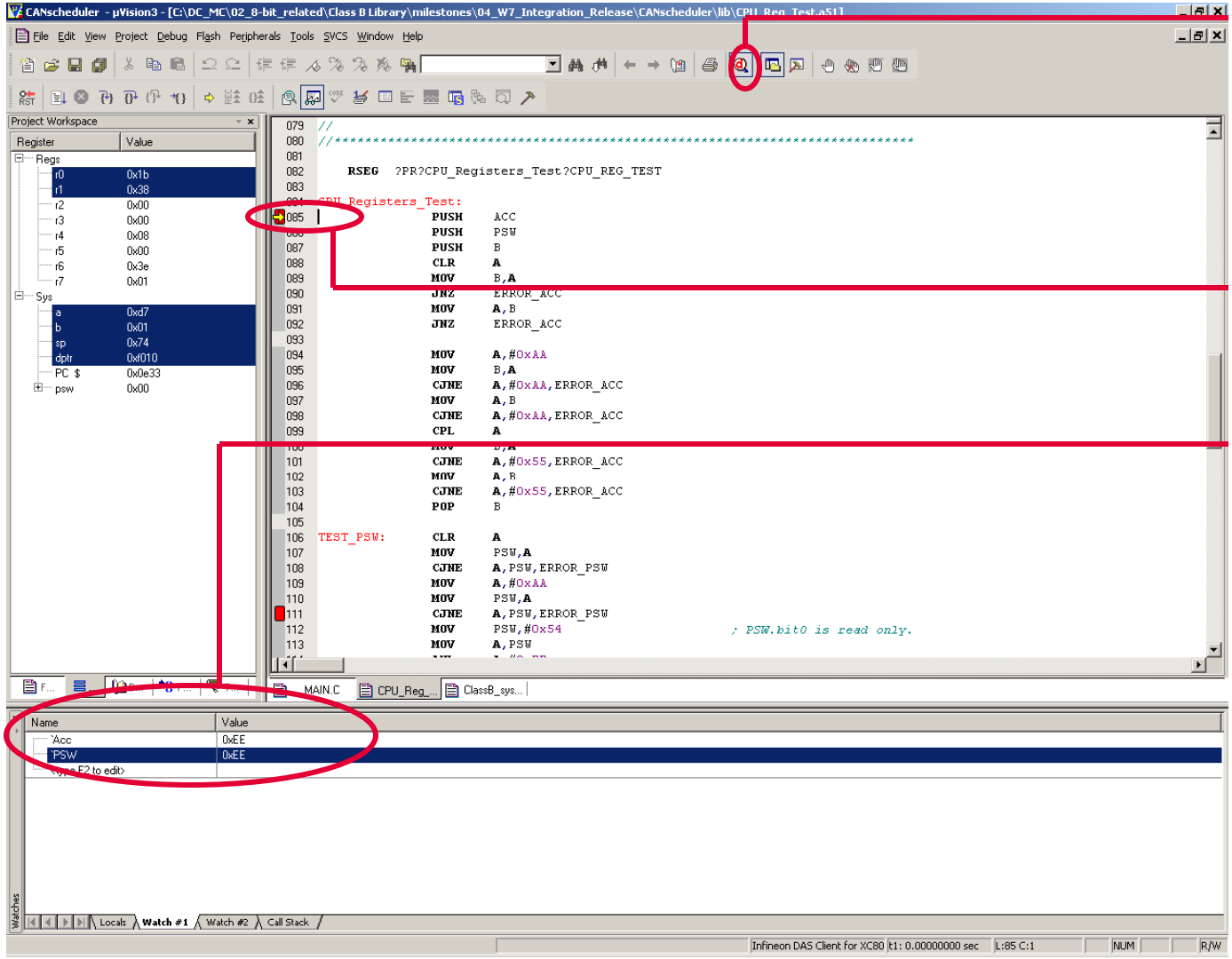
CPU Registers Test –Register B



8. Press “Step Over” button, eventually it will come to line 180 as the expected value in register Acc is not 0xAA and it will jump into error branch
9. Note that in Watch #1 window, registers Acc, B and PSW are restored to the original value 0xEE. The function will return 0x2 as shown in register R7

Copyright © Infineon Technologies 2008. All rights reserved.

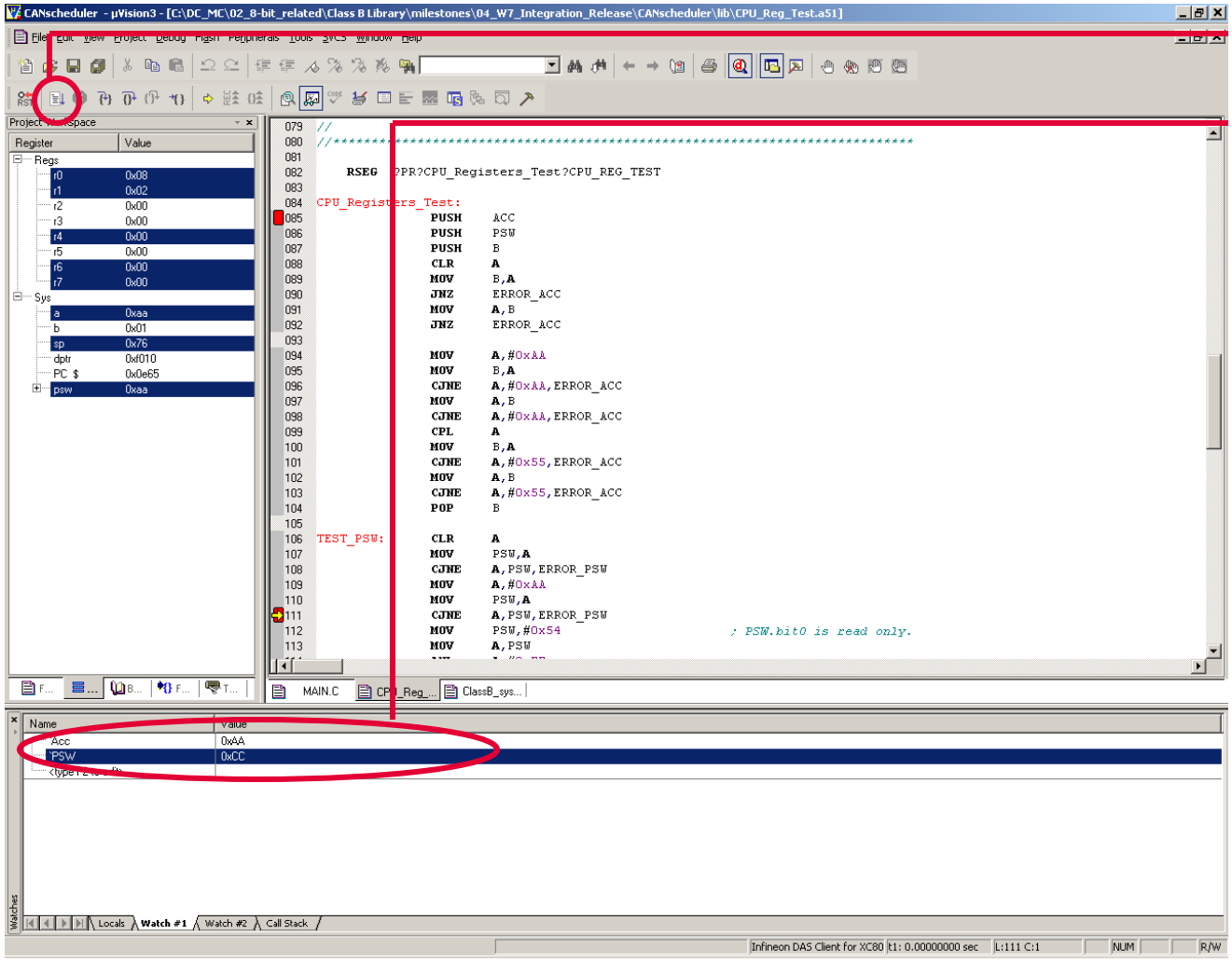
CPU Registers Test –Register PSW



1. Press the "Debug" button to go into debug mode
2. This is a non-destructive routine, therefore it will push the register PSW value into the stack before testing
3. Set breakpoint at line 85 & 111, then press "Run" button and the program shall stop at line 85
4. In Watch #1 window, add registers Acc and PSW with value 0xEE

Copyright © Infineon Technologies 2008. All rights reserved.

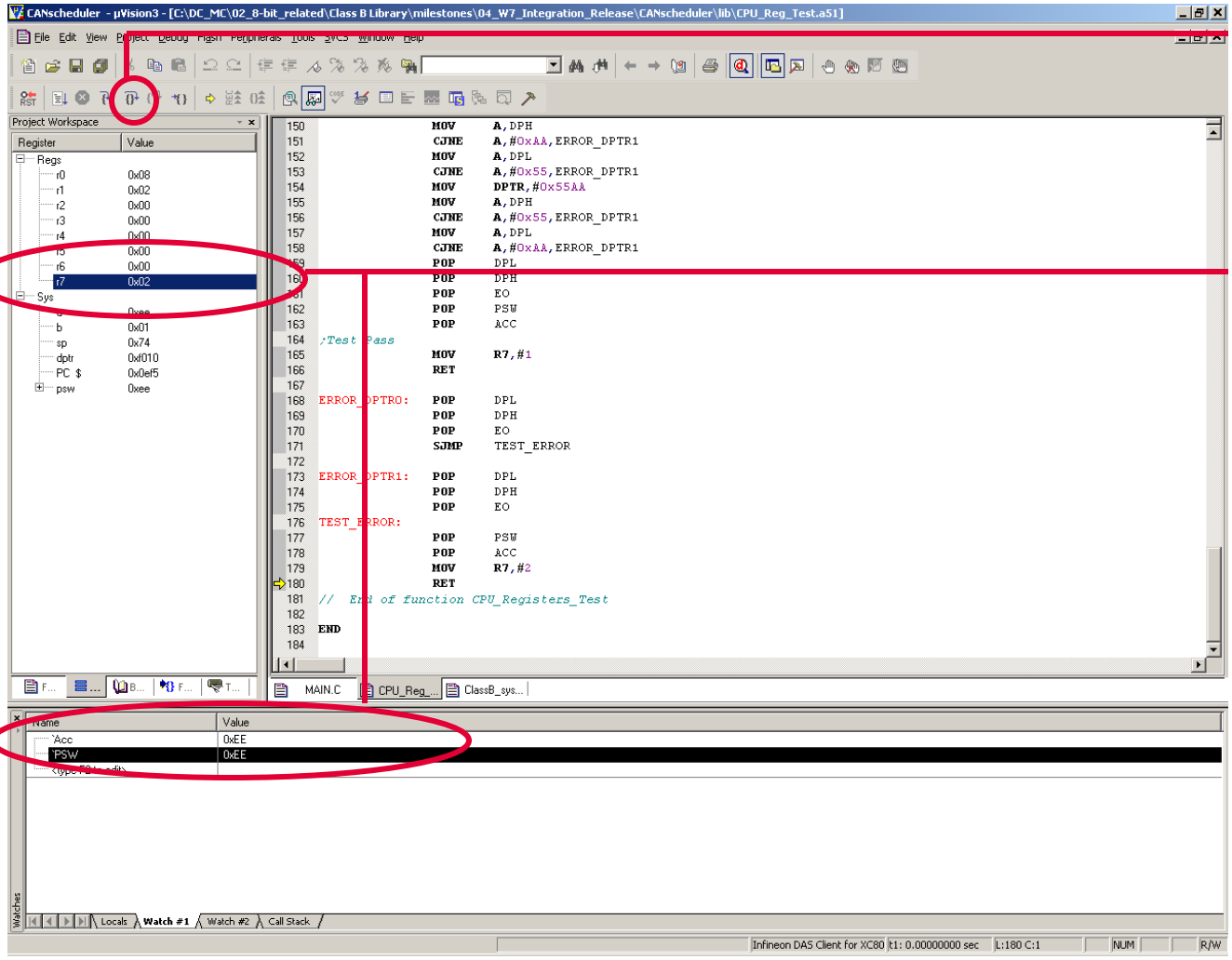
CPU Registers Test –Register PSW



5. Press “Run” button, it will stop at line 111
6. In Watch #1 window, change PSW value from 0xAA to 0xCC

Copyright © Infineon Technologies 2008. All rights reserved.

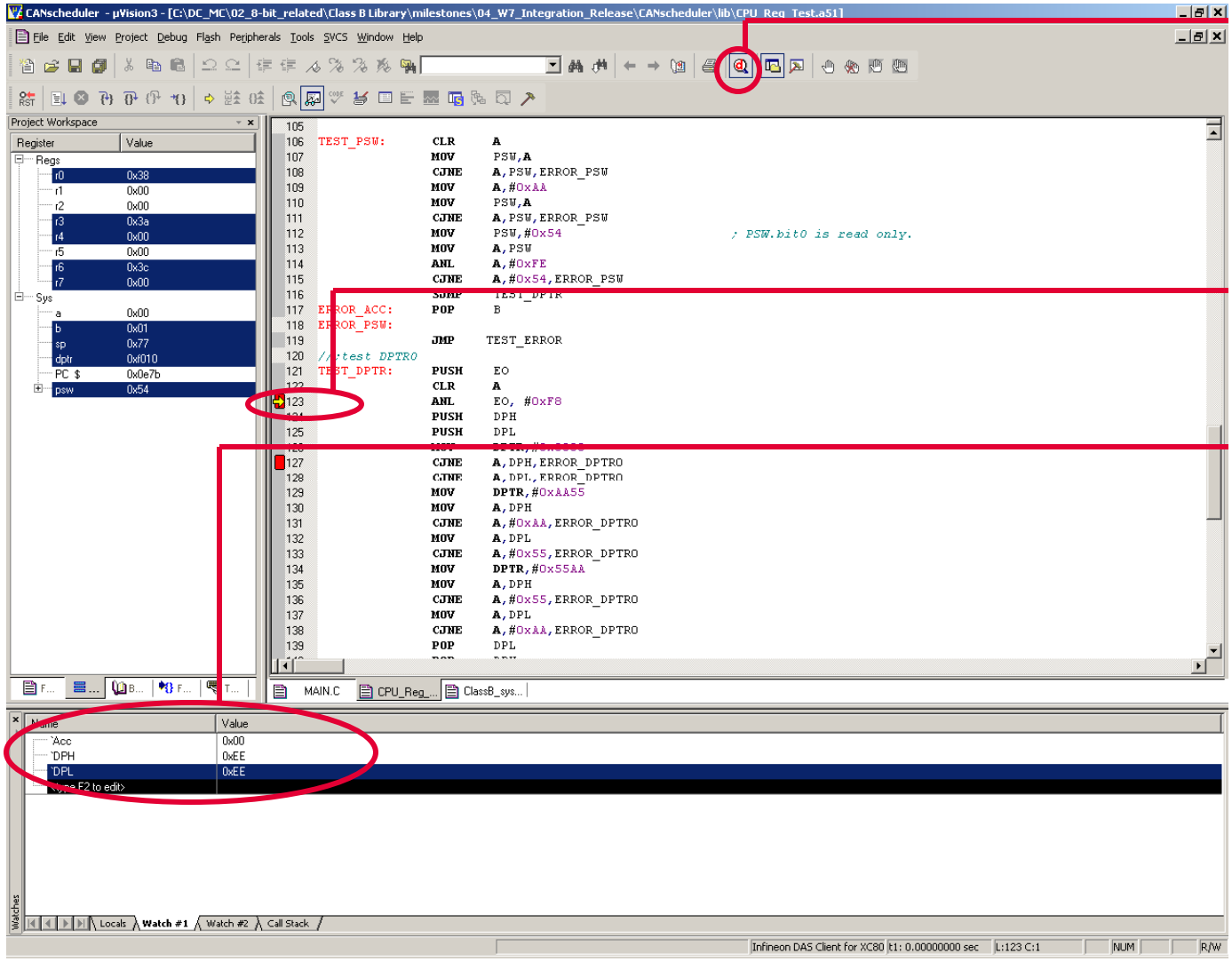
CPU Registers Test –Register PSW



7. Press “Step Over” button, eventually it will come to line 180 as the expected value in register PSW is not 0xAA and it will jump into error branch
8. Note that in Watch #1 window, registers Acc and PSW are restored to the original value 0xEE. The function will return 0x2 as shown in register R7

Copyright © Infineon Technologies 2008. All rights reserved.

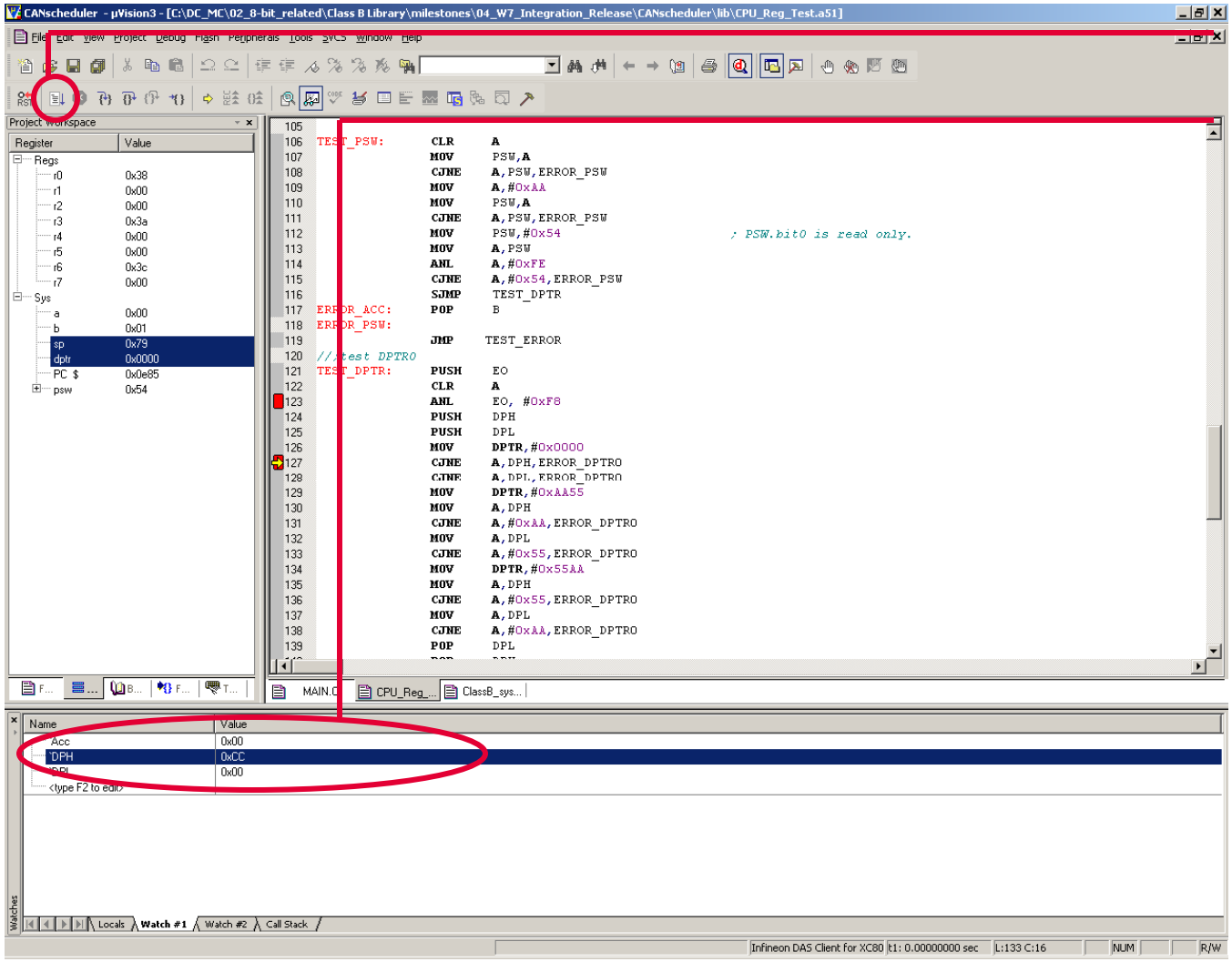
CPU Registers Test –Register DPTR



1. Press the "Debug" button to go into debug mode
2. This is a non-destructive routine, therefore it will push the register DPTR value into the stack before testing
3. Set breakpoint at line 123 & 127, then press "Run" button and the program shall stop at line 123
4. In Watch #1 window, add registers Acc, DPH and DPL. Set DPH and DPL with value 0xEE

Copyright © Infineon Technologies 2008. All rights reserved.

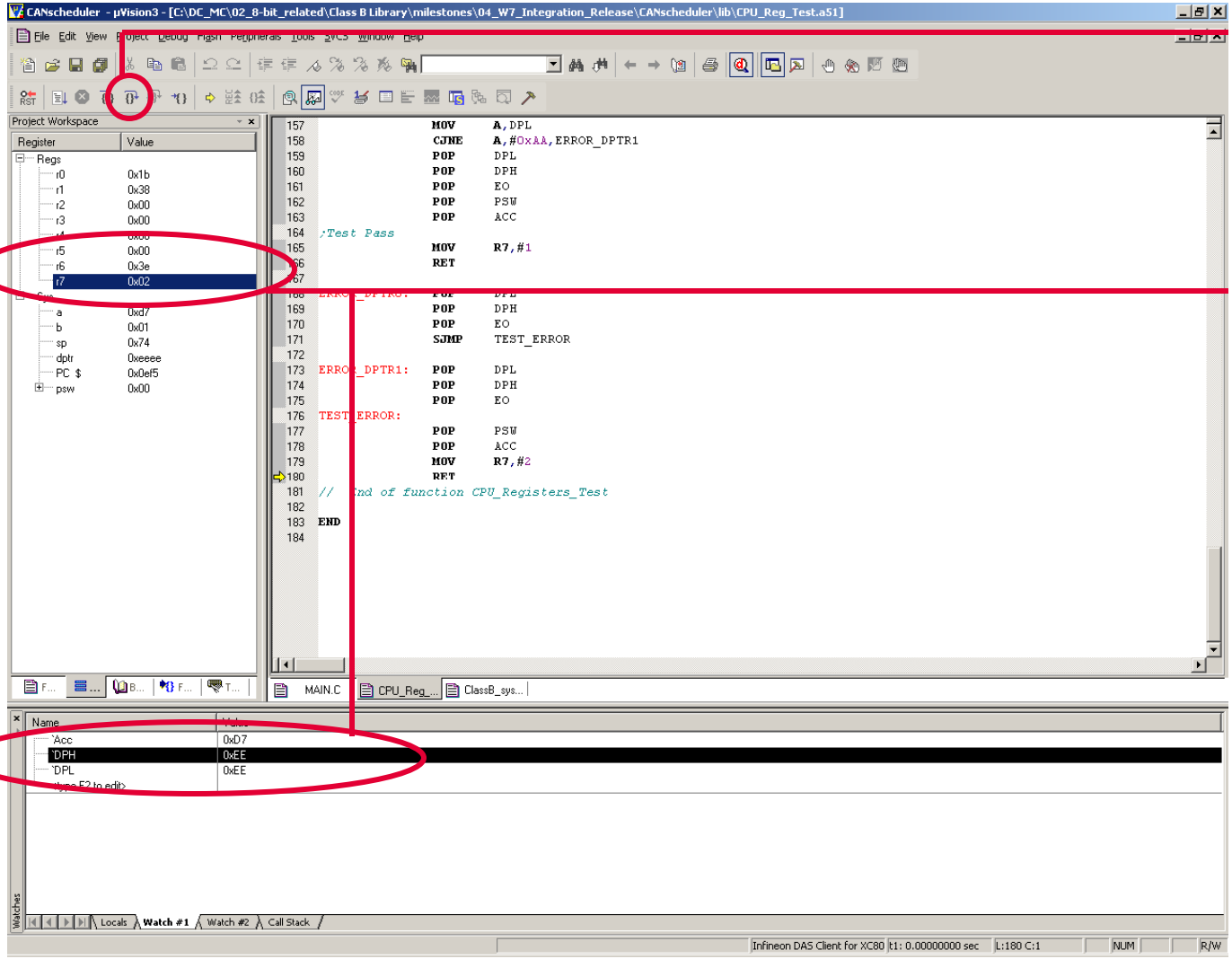
CPU Registers Test –Register DPTR



5. Press “Run” button, it will stop at line 127
6. In Watch #1 window, change DPH value from 0x00 to 0xCC

Copyright © Infineon Technologies 2008. All rights reserved.

CPU Registers Test –Register DPTR



7. Press “Step Over” button, eventually it will come to line 180 as the expected value in register DPH is not 0x00 and it will jump into error branch
8. Note that in Watch #1 window, registers DPH and DPL are restored to the original value 0xEE. The function will return 0x2 as shown in register R7

Copyright © Infineon Technologies 2008. All rights reserved.

Internal Data Path & Address Decoder Logic Test

- The above logic test can be done by calling the routine

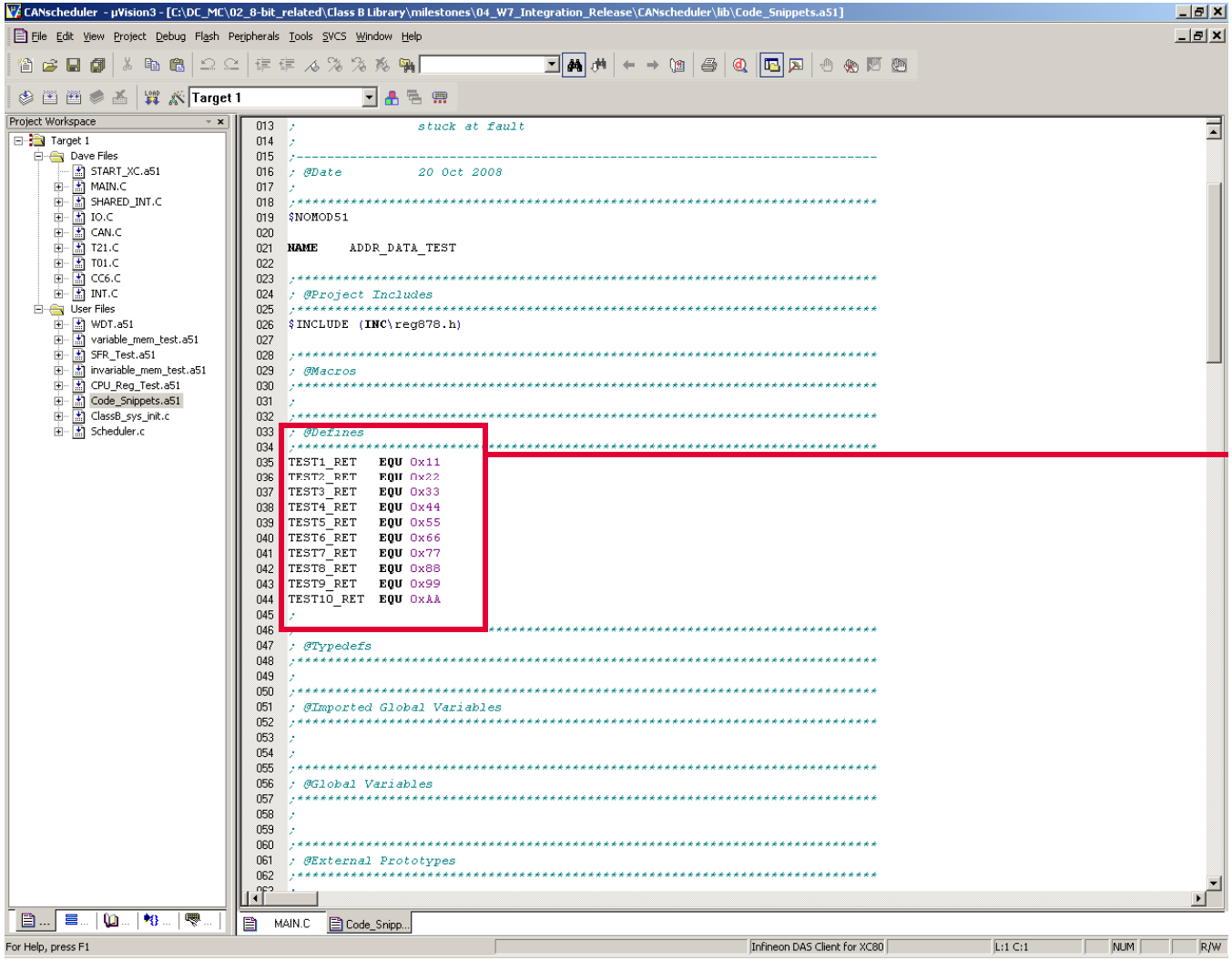
  bit ADDR_DATA_SA_Test(void)

- The routine consists of 10 small routines within, and they will be called and check for the return value individually
- The small routine's address and its return value can be configured by user
  - Return value can be configured in code_snippets.a51 file
  - Address of the individual small routine can be configured from the linker settings in Keil uVision3 workspace

Copyright © Infineon Technologies 2008. All rights reserved.

Internal Data Path & Address Decoder Logic Test

- Return value of the routines in Code_Snippets.a51 file

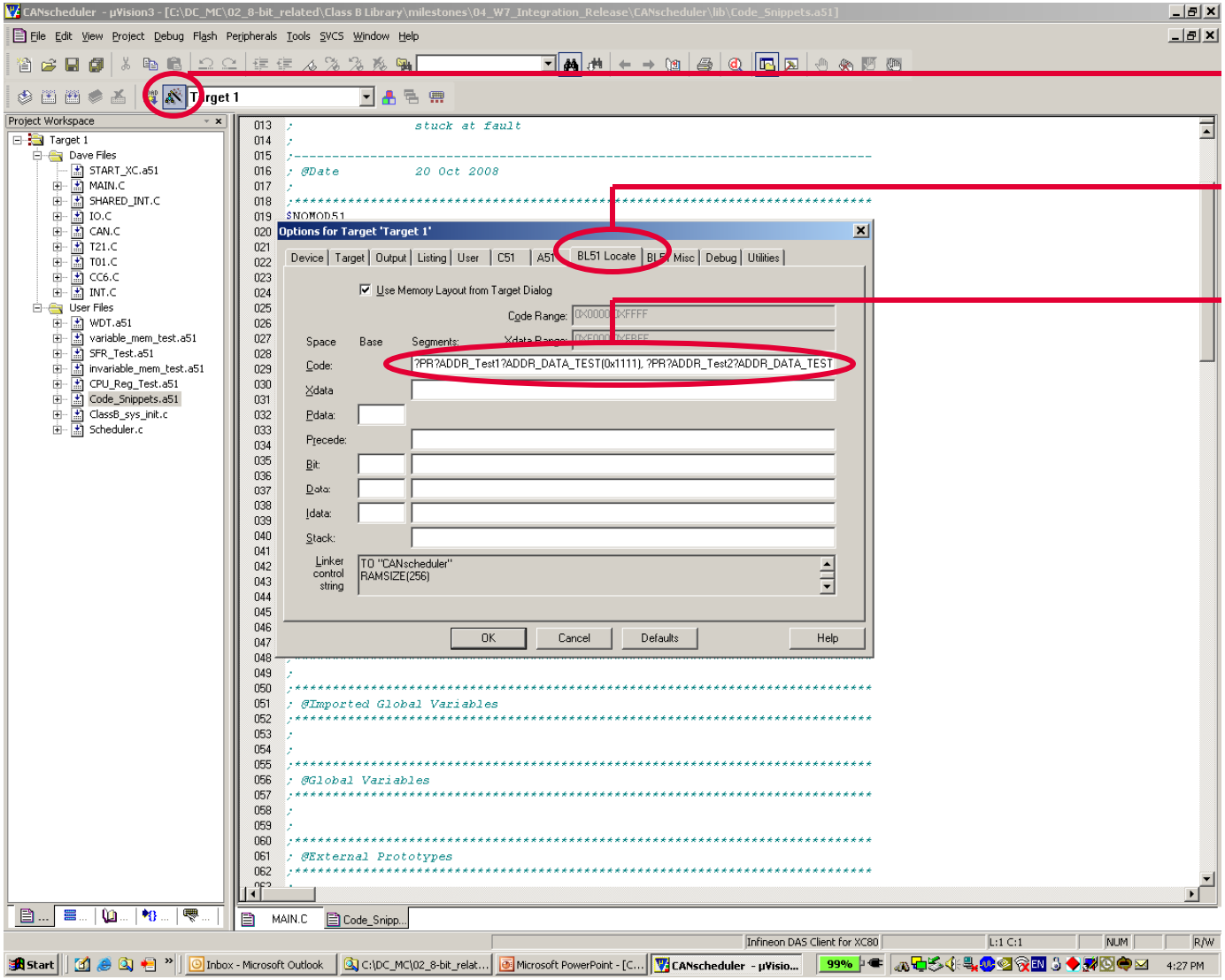


- Individual routine pre-compile return value

Copyright © Infineon Technologies 2008. All rights reserved.

Internal Data Path & Address Decoder Logic Test

- Configuration of the small routine's address in Keil uVision3 workspace



- Click "Options for Target" button
- Select "BL51 Locate" tab
- Enter the routine's address in this column. E.g. ADDR_Test1 routine will be located to address 0x1111

Copyright © Infineon Technologies 2008. All rights reserved.



Never stop thinking