The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205377

FR, MB91460, Flash RAM Copy

This application note describes the procedure to be used to generate code using the Softune Workbench that will be executed out of the RAM memory during the run time of the application. The following description is based on the use of the start91460.asm as provided together with the Softune Workbench Template Project for a given MCU out of the MB91460 series.

Contents

1 Introduction....1
2 Flash/ROM RAM Copy....2
2.1 Placing the RAM code in IRAM Section....2
2.2 Setup Linker....3
2.3 Enable Flash RAM Copy Procedure in start91460.asm....6
2.4 Cautions using Flash-RAM Copy....7
3 Additional Information....8
Document History....9

1 Introduction

This application note describes the procedure to be used to generate code using the Softune Workbench that will be executed out of the RAM memory during the run time of the application.

The following description is based on the use of the start91460.asm as provided together with the Softune Workbench Template Project for a given MCU out of the MB91460 series.

There are situations in microcontroller applications on MB91460 Flash MCUs when it is necessary to execute part of the application code out of the Instruction RAM of the controller because fetching data or code out of the internal Flash memory is currently not possible. Such situations are for example erasing/writing to the embedded Flash memory (Boot loader, EEPROM emulation) or while generating a CRC32 checksum over the Flash memory using this Flash Security feature on MB91460 series MCUs.

In that case code has to be copied from the Flash memory into the instruction RAM of the MCU and has to be executed from instruction RAM.

It is necessary to use the part of the RAM memory of the MB91460 series MCUs that is connected to the instruction bus of the MCU (for details on the architecture of MB91460 series please refer to the hardware manual of MB91460 series). This part of the RAM memory is in the following referred to as General Purpose RAM. For example on MB91F467D the instruction RAM is placed in the address range 0x30000-0x37FFF.

Since data/code placed in RAM memory is lost in case of power down of the MCU and the code in the RAM memory has to be initialized at start-up (after power-on) a copy of the RAM-code has to be available in the non-volatile memory of the MCU (Flash memory connected to the MCU).

The Softune Workbench is supporting this requirement in a way similar to handling RAM initialized RAM variables. Two sections IRAM and @IRAM are generated and the code is linked for the IRAM section placed in the instruction RAM area but in the final load module (abs-file or mhx-file) the code is place in the @IRAM section (in Flash area). The user has to take care to copy the code form @IRAM area to the IRAM area before calling the corresponding functions. The copy procedure is already implemented in the start91460.asm file and can be enabled by a corresponding switch in the start91460.asm file.

The procedure how to generate code to be executed out of the RAM area of the MB91460 series MCU and how to enable the copy process in the start91460.asm consists of the following steps:

- The code wanted to be executed out of RAM has to be placed into the section IRAM.

- Setup of the Softune Workbench linker for placing the IRAM (RAM area) and the @IRAM section (ROM area).
- The Flash RAM copy feature has to be enabled in the start91460.asm.

For a detailed description please see the following chapters of this application note.

2 Flash/ROM RAM Copy

This chapter describes the steps necessary to adapt in the code (C-module containing the code to be placed in RAM), the start-up code (enable the Flash-RAM-Copy feature) and the linker settings to be made to have given code available in RAM at run time.

2.1 Placing the RAM code in IRAM Section

Figure 1. Sample code for a C-module placing the generated code in the IRAM section. IRAM section will be placed in the General-Purpose-RAM of the MCU at run time (see the descriptions following).

```
/*
   The comlete code of the module ram_code.c will be placed in the RAM.
   The location of the RAM area will be controlled by the linker settings.
*/
#pragma section CODE=IRAM, attr=CODE

void RAM_MyRAMFunc1(void);
void RAM_MyRAMFunc2(void);

void RAM_MyRAMFunc1()
{
    RAM_MyRAMFunc2();
}

void RAM_MyRAMFunc2()
{
    /* This function is doing nothing */
}
```

The Softune Workbench supports generation of sections and placement of code, data or constants in these sections for the C language supported via the #pragma section or via the #pragma segment directives.

The difference between the two #pragma directives is that #pragma section will have influence on the complete C-module it is placed in. In contrast the #pragma segment directive only places the part following the #pragma segment directive into the corresponding defined sections. For details about these two #pragma directives please refer to the C/C++ manual of Softune Workbench V6.

The two directives have the following syntax:

#pragma section DEFSECT[=NEWNAME][,attr=SECTATTR][,locate=ADDR]

or

#pragma segment DEFSECT[=NEWNAME][,attr=SECTATTR][,locate=ADDR]

The code sample shown in Figure 1 uses the #pragma section directive to place the code of the complete C-module in the section called IRAM. The placement of the section IRAM can be controlled directly using the “locate” of the `#pragma section` directive together with a absolute address or using the linker settings. Using the linker settings for placing of sections is the recommended way because in that the linker has complete control about optimal placement of sections.

2.2 Setup Linker

2.2.1 2.2.1 Using Softune Workbench GUI to control section placement

Figure 2 shows the window of the Softune Workbench used to control the memory areas defined for the used controller. It is also possible to add/remove own memory areas in the ROM/RAM area of the used MCU. This is done using the Set-Button in the ROM/RAM Area List field and following the given instructions.

Please note the distinction between D_RAM area and ID_RAM area in the shown example. The MB91460 series supports two different RAM areas: One connected to the Data-bus (in this example called D_RAM) which is mainly used for giving fast access to data placed in RAM and in addition a general-purpose RAM (in the example called ID_RAM) that is connected to the instruction bus of the CPU. From this RAM area it is possible to access both instructions and data. It is recommended to place code into this area. For details about the addresses and sizes for Data-RAM and General-Purpose-RAM please refer to the datasheet of the used MB91460 series MCU.

To locate the IRAM section into the ID_RAM memory area please use Set Section button in the window shown in Figure 2. This will display an additional window shown in

Figure 3 where the ID_RAM memory area can be selected and sections can be added to the ID_RAM memory area.

Figure 2. Window of Softune Workbench for control of the different memory areas placed in RAM/ROM memory

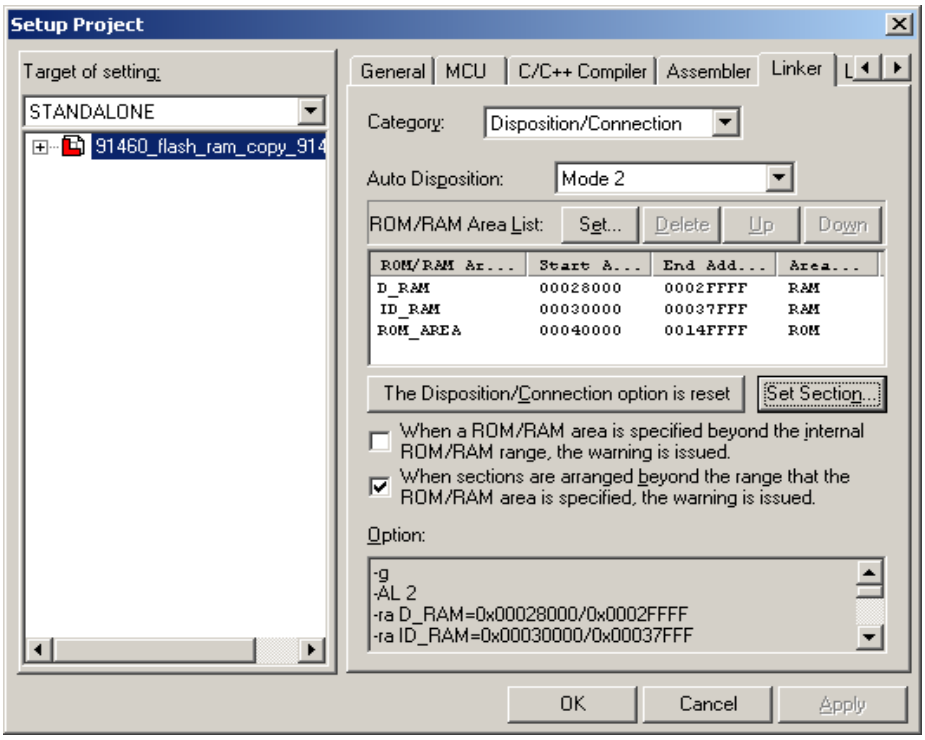
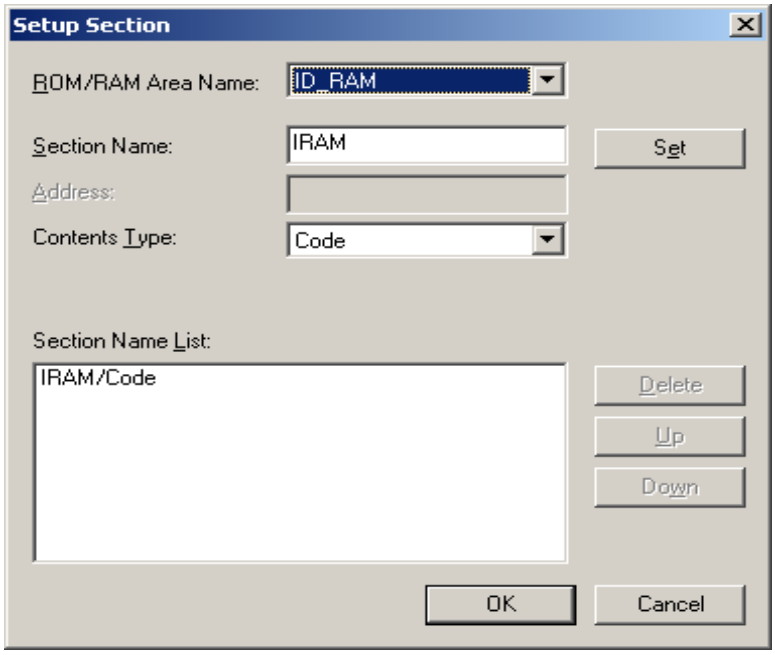
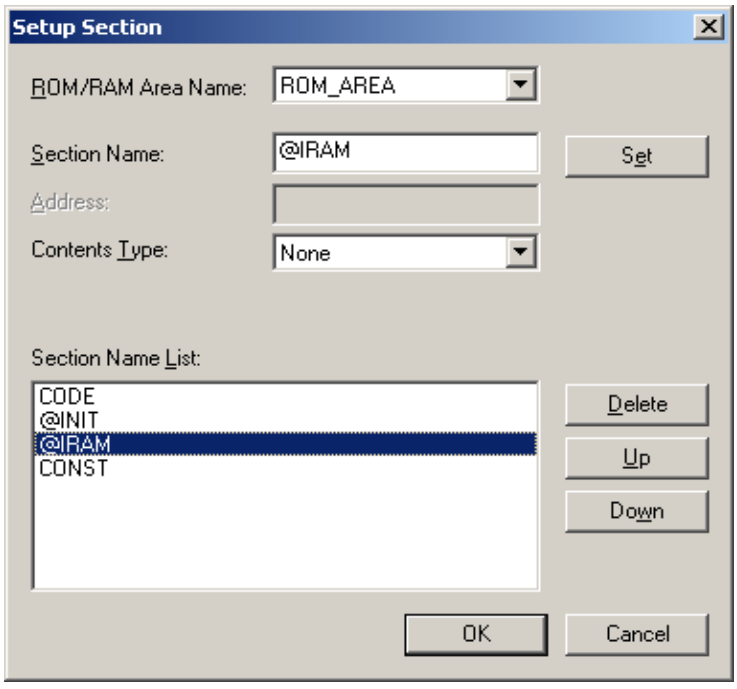


Figure 3. Add IRAM section to the ID_RAM Memory Area



In the next step, a section @IRAM has to be placed in the Flash/ROM memory to have the code available for copying it into the RAM on power-on. The procedure is similar to the procedure for placing the IRAM section itself. As shown in the example there is already a Flash/ROM memory area called ROM_AREA defined in the address range 0x40000 – 0x13FFFF. Here again using the Set Section Button opening the corresponding window has to be used to place the @IRAM section into the ROM_AREA (see Figure 4)

Figure 4. Add @IRAM section to the ROM_AREA.



The generation of memory areas will be controlled by the linker options –ra (definition of RAM memory area) –ro (definition of Flash/ROM memory area) and –sc (section allocation). For details about the Softune Workbench Linker options please refer to the Softune Linkage Kit V6 Manual.

2.3 Enable Flash RAM Copy Procedure in start91460.asm

The routines that copy the content of the @IRAM section (placed in the Flash/ROM memory of the MCU) into the IRAM section (placed in the RAM memory of the MCU) are included in the start91460.asm start-up code as provided in the Softune Workbench template projects which are available at the Fujitsu web-page (the link to the Fujitsu web-page is given in the appendix of this application note). The Flash-RAM-Copy feature is disabled as default in the start91460.asm. Please make sure to have the feature enabled when the project is build. The sample code (parts extracted from the start91460.asm) in shows the parts related to the Flash-RAM-copy feature.

When the Flash-RAM-Copy feature is enabled by setting the switch under point 4.4 of start91460.asm to ON the addresses _RAM_IRAM and _ROM_IRAM that are generated by the Softune Workbench are imported and entered as parameters into the Flash-RAM-Copy-routine.

If the application is not using start91460.asm for start-up code the start-up code should contain a own Flash-RAM-Copy-routine using _RAM_IRAM and _ROM_IRAM as start addresses for IRAM section and @IRAM section since this information is generated from the Softune Workbench language tools.

Figure 5. Sample code related to the Flash-RAM-Copy feature as extracted from start91460.asm (remark: "..." is symbol for the rest of start91460.asm not used herein).

```
;================================================================================
; 4.4  Copy code from Flash to I-RAM
;================================================================================
; #set    I_RAM       ON        ; <<< select  if  code  in  section IRAM
;                                     should be copied
;; If this option is activated code located in the  section IRAM is copied
; during startup from ROM to the instruction-RAM. The code is linked
; for the instruction-RAM.
#if I_RAM
        .import _RAM_IRAM
        .import _ROM_IRAM
#endif
;================================================================================
; 7.8  Copy code from Flash to I-RAM
;================================================================================
#if I_RAM == ON
        LDI             #_RAM_IRAM, R0
        LDI             #_ROM_IRAM, R1
        LDI             #sizeof(IRAM), R13
        CMP             #0, R13
        BEQ             copy_iram_end
copy_iram1:
        ADD             #-1, R13
        LDUB            @(R13, R1), R12
        BNE:D           copy_iram1
        STB             R12, @(R13, R0)
copy_iram_end:
        ClearRCwatchdog
#endif
```

2.4 Cautions using Flash-RAM Copy

Do not use functions located in the RAM area of the MCU before the corresponding area was initialized by the Flash RAM copy feature since this RAM area has no defined content after reset and behavior of the MCU cannot be predicted.

As an example, please see the following two screen shots taken from the Softune Workbench Simulator showing the behavior of the MCU before (Figure 6) and after (Figure 7) passing through the Flash-RAM-Copy-Routine in start91460.asm.

Figure 6. Screen shot of Softune Workbench Simulator. Break point at start of Flash-RAM-Copy-Routine hit. The RAM memory where the RAM-code is expected is empty/ uninitialized (here filled with 0x0000).

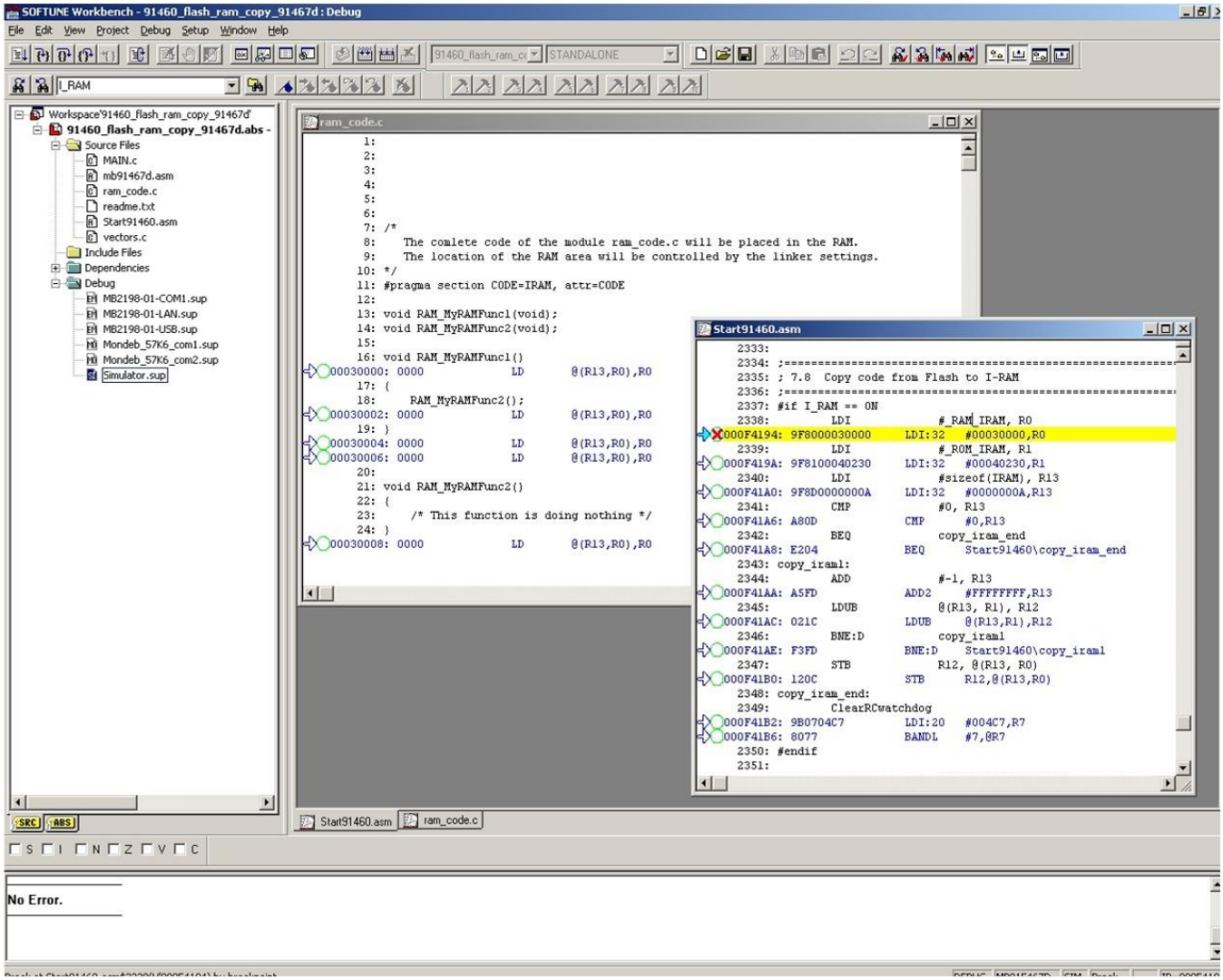
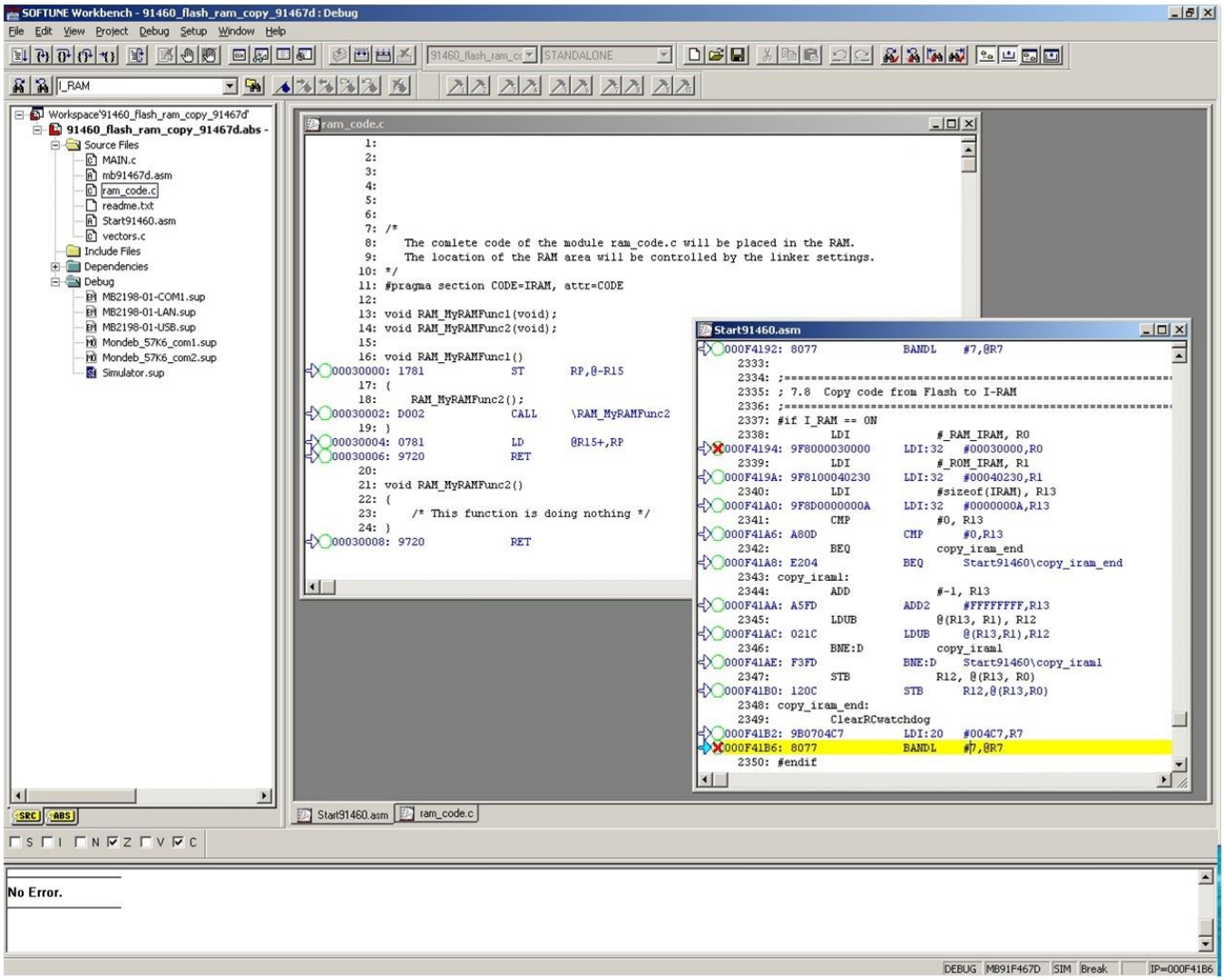


Figure 7. Screen shot of Softune Workbench Simulator. Break point at end of Flash-RAM-Copy-Routine hit. The RAM memory where the RAM-code is filled with the corresponding machine code as expected.



3 Additional Information

Link to Cypress Webpage

For downloading Softune Workbench samples, Application Note, Documentation on MCUs from the Cypress web-page please use the following links:

http://www.cypress.com/cypress-microcontrollers

http://www.cypress.com/cypress-mcu-product-softwareexamples

Document History

Document Title: AN205377 - FR, MB91460, Flash RAM Copy

Document Number: 002-05377

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	02/07/2008	V1.0 RSchum First Version
*A	5133810	NOFL	02/11/2016	Converted Spansion Application Note “MCU-AN-300087-E-V10” to Cypress format
*B	6070279	NOFL	02/13/2018	Sunset Review Migrated to new template Updated links

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/go/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2008-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.
Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.