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.

AN205436

FR, MB91460, Accemic MDE 2008 Integration

In the following a description is given how to generally include the Accemic MDE 2008 into an application written for MB91460 Series using the language tools of the Softune Workbench. Here the focus is deliberately placed on how to integrate the MDE 2008 not using the Softune Workbench IDE but by directly using the command line tools (compiler, assembler and linker) as they might be used within an a makefile build environment.

Contents

1 Introduction 1
2 Basic MDE 2008 integration into an existing application 1
2.1 General remarks 1
2.2 Template Project 2
2.3 Preparing an application for use with Accemic MDE 3
3 MDE 2008 integration into an existing application 4
3.1 Example for a File/Directory Structure 4
3.2 Modifications of the Application for MDE2008 4
4 Impact of Including MDE 2008 into an Application 10
5 Appendix 11
5.1 Complete Sample Makefile for Building the Sample Application 11
5.2 Sample for removing the call to acc_InitKernel() from mhx-File 13
Document History 15

1 Introduction

In the following a description is given how to generally include the Accemic MDE 2008 into an application written for Cypress MB91460 Series using the language tools of the Cypress Softune Workbench. Here the focus is deliberately placed on how to integrate the MDE 2008 not using the Softune Workbench IDE but by directly using the command line tools (compiler, assembler and linker) as they might be used within an a make file build environment.

2 Basic MDE 2008 integration into an existing application

This chapter will over view on how to integrate the Accemic MDE 2008 debugger into an existing. The information here was kindly given by the company Accemic.

2.1 General remarks

In comparison to Accemic MDE 2006 with a precompiled monitor kernel, the kernel of Accemic MDE 2008 is linkable to the user project.

A template project is provided which demonstrates the implementation of the monitor kernel into an application.

After a reset your application has to call at least the **acc_InitKernel** function to initialize the monitor kernel. The best location is after the system stack initialization. You may call it after the clock initialization (for better download performance), but this should only be done if your clock initialization is running well. If you want to stop your application, you have to call **acc_CallKernel** or you application will continue to run.

The kernel will always be entered after an **acc_CallKernel** procedure, after a break or if it receives the stop command. It will be left again if the run command is received.

The **acc_CallKernel** function establishes a communication to the PC. The program memory content will be compared with the loaded *.abs file. If there is a difference, the flasher will be loaded and executed. After execution of the flasher a reset should be applied.

With the Debugger Configuration File memory regions can be excluded from the update of the flash memory.

2.1.1 Resources

To get detailed information on the used system resources, please refer to the *.mp1 file, that is generated by the Softune linker.

All sections that are used by the monitor start with ‘ACC_LIB’.

Strongly recommended literature:

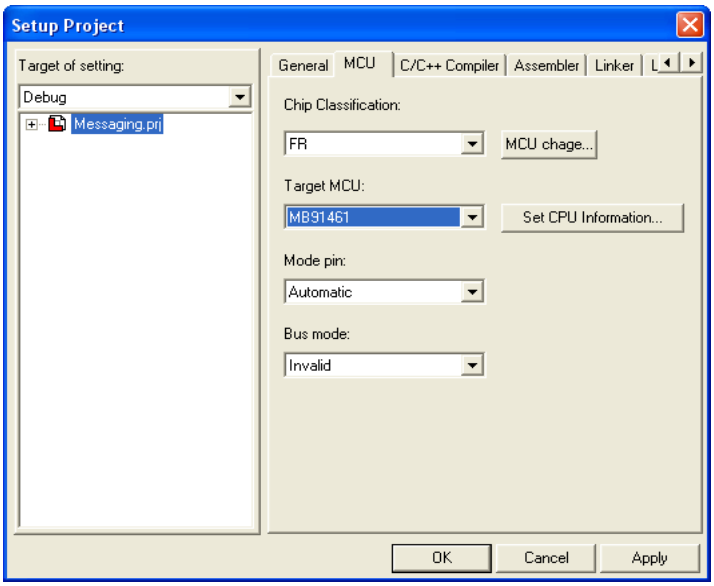
- FR FAMILY EMBEDDED C PROGRAMMING MANUAL
- FR FAMILY SOFTUNE C/C++ COMPILER MANUAL
- FR FAMILY SOFTUNE LINKAGE KIT MANUAL

2.2 Template Project

The monitor template is the starting point for each application development. It still contains the initialization of the monitor.

Based on this template an application with monitor support can be implemented easily.

1. Start the FR Family Softune Workbench and load the workspace: [Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\Templates\templates.wsp
2. Locate your MCU family and UART type project and set it to the active project (right click on project).
3. Go to Project -> Setup Project -> MCU and select the target CPU type.



4. When changing CPU type and setting the linker adjustment to default values the I-Cache area may be included to the data RAM area. Please exclude this area from data RAM in this case afterwards or preserve the linker settings of the template project.
5. In case of using the I-Cache as RAM area, please be sure to carefully read the corresponding sections in the hardware manual.
6. Open MDE_[Family name].c
7. Change the main oscillator frequency in acc_ComConfig
8. Change the sub oscillator frequency in acc_ComConfig
9. Build the project by clicking the “Build” button or by performing the command Project - Build in the Softune Workbench.
10. Add the generated file ‘[Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\Templates \[Family name]\Debug\ABS\template_XXX.abs’ to your project browser.
11. Drag & Drop the template_XXX.abs from the Project Navigator to the prepared processor in the Processor Navigator.
12. Press if you are not connected.

2.3 Preparing an application for use with Accemic MDE

1. Open your application.
2. Add [Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\LIB\MonitorCore.lib to your project
3. Add [Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\LIB\ MonitorCom_[Family name]_U[Bootloader UART].lib to your project
4. Copy [Accemic MDE Directory]\sources\Fujitsu\FR\SRC\MDE_[Family name].c to your project, open it and
   a. Change the access key in acc_KernelConfig
   b. Change the main oscillator frequency in acc_ComConfig
   c. Change the sub oscillator frequency in acc_ComConfig
   d. Insert your code for the watchdog checking into acc_CheckWatchdog
5. Recalculate your required stack size very carefully.
6. Please note the predefined stack size in “start.asm” may NOT be sufficient if you use Softune library functions. For more information on the required stack size, please refer to \\softune\lib\911*.stk. Stack overflow will cause a system crash. See also the chapter ‘System Resources’ for desired stack size.
7. Insert the following code snippet AFTER the system stack initialization of your application (Normally found in start91XXXX.asm)

```
.import _acc_InitKernel
.import _acc_CallKernel
LDI #_acc_InitKernel,R0
CALL        @R0
LDI #_acc_CallKernel,R0
CALL        @R0
```

   If you doesn’t execute the acc_Init_Kernel, the monitor debugger may not work.
8. You may remove the _acc_CallKernel part if you want to start your application directly. But this should only be done if you are absolutely sure that your application is running correctly. If you are sure that your clock initialization is running well, you can place the code above after the clock initialization to achieve faster download results during connection.
9. Locate the part in your code where the ISR register of your used UART is set. Remove this, because the ISR register is initialized by the monitor. If you change the monitor kernel ISR, the monitor debugger may not work.
10. Locate the part in your code where interrupt vectors are set. Set:
11. The RX interrupt of your used UART to acc_Receive_IRQ
    a. The TX interrupt of your used UART to acc_Transmit_IRQ
    b. All interrupts used by the debug unit to :
       - MB91V650: Interrupt 15 to acc_InstructionBreak_IRQ
       - MB91V460: Interrupt 10 to acc_InstructionBreak_IRQ
       - (Refer to the template project for detailed implementation)

If you don’t set the interrupt vectors, the monitor debugger may not work.

12. Verify that __EI () and __set_il(31) is called in your application. If you doesn’t enable the interrupts, the monitor debugger may not work.
13. It is advisable to disable all optimization options in the C Compiler preferences.
14. These options can be set under “Project →Setup Tool Options → C Compiler → Category: Optimize” in the FR Family Softune Workbench environment.
15. Build the project by clicking the “Build” button or by performing the command Project - Build in the Softune Workbench.
16. Add the generated file ‘[Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\Templates \[Family name]\Debug\ABS\template_XXX.abs’ to your project browser.

17. Drag & Drop the template_XXX.abs from the Project Navigator to the prepared processor in the Processor Navigator.
18. Press  if you are not connected.

3 MDE 2008 integration into an existing application

This chapter will describe the integration of Accemic MDE 2008 into an existing FR MB91460 MCU. To explain the necessary steps and the impact of the integration of MDE 2008 into an existing application an example is created based on the source files of a Softune Workbench Template project together with the GNU make tool.

3.1 Example for a File/Directory Structure

To have a base to start from it is assumed the existing application is placed in a file/directory structure like (the sources used in this sample are taken form a standard Softune Workbench template project fro MB91F467B):

Application_Dir -+

- inc -+
  - mb91467b.h
  - vectors.h
- src -+
  - main.c
  - mb91467b.asm
  - start91460.asm
  - vectors.c
- lib -+
- lst -+
- build -+
- makefile

The directory “lib” is planned to contain libraries of the project that might have to linked when generating the .asb-file. The “lst” directory is intended to hold the list files generated by the compiler, assembler and linker. Finally the “build” directory is storing the output files of linker and load module converter (when used). In **Error! Reference source not ound.** one can find a sample makefile to be used with the GNU make tool to build the above described sample project.

3.2 Modifications of the Application for MDE2008

To add MDE 2008 to an already existing application one has to add a few source- and header-files to the list of files to be compiled/assembled during the build process. In addition two libraries are needed for completing the link process.

Since the vector table of the application has to include interrupt vectors for the USART communication and an interrupt vector for the instruction break IRQ the file containing the vector table definitions has to be modified (in this example in vectors.c). The Interrupt Control Register for the used USART has to be controlled only by the Accemic MDE 2008. For this reason the places in the original code changing the content of the corresponding ICRxx register has to be taken out of the original code.

Figure 1. Sample makefile for GNU make tool

```
# Sample make file
TOOLDIR = U:\bin

# Directory Structure
OBJECTDIR = ./obj
SOURCEDIR = ./src
INCLUDEDIR = ./inc
LIBDIR = ./lib
BUILDDIR = ./build
LISTDIR = ./lst

# Language Tools
CC = fcc911s.exe
ASM = fasm911s.exe
LINK = flnk911s.exe
CONVERT = f2ms.exe

# Options
CCOPT = -c -g -cpu MB91F467B -I $(INCLUDEDIR)
ASMOPT = -g -cpu MB91F467B -I $(INCLUDEDIR)
LINKOPT = -cpu MB91F467B -g -AL 2 -Xset_rora -ra D_RAM=0x0002A000/0x0002FFFF -ra          \
          ID_RAM=0x00030000/0x00033FFF -ro ROM_AREA=0x00040000/0x0014FFFF                 \
          -sc DATA/Data+INIT/Data+SSTACK/Data+USTACK/Data=D_RAM -sc IDRAM/Code=ID_RAM     \
          -sc CODE+@INIT+@IDRAM+CONST=ROM_AREA                                           \
          -sc CODE_START/Code=0x000F4000 -sc INTVECT/Const=0x000FFC00
CONVERTOPT = -S3

# List of Object Files
OBJECTS = $(OBJECTDIR)/main.obj $(OBJECTDIR)/vectors.obj $(OBJECTDIR)/mb91467b.obj        \
          $(OBJECTDIR)/start91460.obj

# Rules
output: $(OBJECTS)
        $(TOOLDIR)\$(LINK) $(LINKOPT) $(OBJECTS) -m $(LISTDIR)/output.mp1 -o
$(BUILDDIR)/output.abs
        $(TOOLDIR)\$(CONVERT) $(CONVERTOPT) -o $(BUILDDIR)/output.mhx $(BUILDDIR)/output.abs

$(OBJECTDIR)/%.obj: $(SOURCEDIR)/%.asm
        $(TOOLDIR)\$(ASM) $(ASMOPT) -o $@ $<

$(OBJECTDIR)/%.obj: $(SOURCEDIR)/%.c
        $(TOOLDIR)\$(CC) $(CCOPT) -o $@ $<

.PHONY: clean
clean:
        -rm -rf $(OBJECTDIR)/*.obj
        -rm -rf $(ACC_OBJECTDIR)/*.obj
        -rm -rf $(LISTDIR)/*.*
        -rm -rf $(BUILDDIR)/output.abs
```

The preparation of the Accemic MDE 2008 debugger kernel has to be done early in the startup phase of start91460.asm (it is recommended to do this right after initialization of the stacks). The initialization of the MDE 2008 debugger kernel is done by calling the function acc_InitKernel() (followed by acc_CallKernel() if one wants the MDE 2008 to halt right after initialization).

3.2.1 Adding the MDE 2008 Libraries, Source Files and Header Files to the Application

To have a clear separation of the MDE 2008 from the application one may add directories to the above described directory structure below the Application_Dir directory and add the needed source files, header files and libraries to these additional directories

Application_Dir -+

…

acc_src -+

MDE_MB91V460.c

acc_inc -+

cpu.h

acc_lib -+

MonitorCom_MB91V460_U4.lib

MonitorCore.lib

acc_obj -+

The corresponding source files, header files and libraries are distributed with the installation of Accemic MDE 2008.

3.2.2 Adapting the Compiler and Assembler Settings

The compiler settings of the application have to be extended by an option telling the compiler where to find the header files needed for the compilation of MDE_MB91V460.c. This is done by using the –I compiler option. For details on the Softune Workbench Compiler options please refer to the Softune Workbench Compiler manual. In this sample one has to add the option -I ./acc_inc to the compiler options.

3.2.3 Initialization of ACCEMIC MDE 2008 Kernel

As mentioned before the MDE 2008 debugger kernel has to be initialized to be able to establish the communication to the PC software when required. It is recommended to include the call the initialization function `acc_InitKernel()` into the startup code right after the initialization of the stacks. But the user is free to place the corresponding function call `acc_InitKernel()` anywhere into the existing application. The only requirement is that stack has to available when `acc_InitKernel()` is called. One should keep in mind that the application code executed before the call of `acc_InitKernel()` can not be debugged. Please refer to **Error! Reference source not found.** for an example ow to include the MDE 2008 debugger kernel initialization into `start91460.asm`.

To ease the change between having the debugger included into the application or not the call of `acc_InitKernel()` is surrounded by pre-processor directives to select between the call to `acc_InitKernel()` or an corresponding number or `NOP` instructions (the `NOP` instructions will keep the size of the start91460.asm code as same as when the acc_InitKernel call is included). The replacement by NOP instructions is one way to ensure that the memory layout of the module start91460.asm is not changing when the application is build with or without including the MDE 2008 debugger. An possible alternative way is to directly change the mhx-file (in this sample `output.mhx`) which can be generated from the linker output output.abs by using the Softune Workbench tool f2ms.exe. For details on the Softune Workbench load module file conversion tools please refer to the manual of the Softune Workbench Linker Kit. The idea here is to replace the assembly opcodes for the assembly instructions used for calling the MDE2008 initialization by a corresponding number of `NOP` instructions. For information on the FR assembly language instructions and their corresponding opcodes please refer to the FR Programming Manual. Please be aware that one has to have a clear understanding of the structure of the used output-file formats (e.g. mhx-file format) and the area where the corresponding addresses where the targeted instruction are to be found. It is strongly recommended to perform intensive re-test of the application after such change has to be done on the load module file to make sure this change has no undesired side effects. **Please be aware that Cypress can not taking any liability when the load module file was changed in this way.** An example of changing the mhx-file for the part initializing the MDE 2008 kernel in start91460.asm can be found in the appendix of this document.

Figure 2. Code added to start91460.asm after stack initialization to initialize MDE 2008 debugger kernel.

```
;=============================================================================================
; 7.1       Initialise Stack Pointer and Table Base Register
;=============================================================================================
#if STACKUSE == SYSSTACK
        ORCCR           #0x20
        LDI             #__userstack_top, SP    ; initialize SP
        ANDCCR          #0xDF
        LDI             #__systemstack_top, SP  ; initialize SP
#endif

#if STACKUSE == USRSTACK
        ANDCCR          #0xDF
        LDI             #__systemstack_top, SP  ; initialize SP
        ORCCR           #0x20
        LDI             #__userstack_top, SP    ; initialize SP
#endif

;=============================================================================================
; 7.1a       Initialise Accemic MDE 2008 Kernel
;=============================================================================================

#ifdef ACC_DEBUGGER
        .import _acc_InitKernel
        .import _acc_CallKernel
        LDI:32          #_acc_InitKernel, R0
        CALL            @R0
;       LDI:32          #_acc_CallKernel, R0
;       CALL            @R0
#else
        NOP
        NOP
        NOP
        NOP
;       NOP
;       NOP
;       NOP
;       NOP
#endif
```

3.2.4 Adding the needed entries for MDE 2008 to the Application Vector Table

For the communication between the Accemic MDE 2008 kernel and the PC debugger software a USART of the MB91460 MCU is used (USART4 is chosen in this sample). On the MCU the USART is interrupt controlled. In addition the MDE 2008 kernel makes use of the Instruction Break interrupt (interrupt number 10) of the MB91460 MCU. The corresponding interrupt vector addresses have to be included into the vector table definition of the application. To ensure the interrupt priority of the USART used to the MDE 2008 serial communication is only controlled by the MDE 2008 one has to exclude the code from the current application that is setting up the interrupt control register ICRxx responsible for the chosen USART. Shows for this example how these two requirements are fulfilled in the template project (vectors.c). The corresponding interrupt handler functions acc_Receive_IRQ(), acc_Transmit_IRQ() and acc_InstructionBreak_IRQ() are part of the MonitorCom_MB91V460_U4.lib. To ease the use of the files in both cases (with and without Accemic MDE 2008) the corresponding changes are surrounded by pre-processor directives.

```
void InitIrqLevels(void)
{
    /*  ICRxx */
    /* Softune Workbench Monitor Debugger is using ext int0 for abort function */
…
    ICR24 = 31;    /* System Reserved              */
                   /* System Reserved              */
#ifndef ACC_DEBUGGER
    ICR25 = 31;    /* USART (LIN, FIFO) 4 RX       */
                   /* USART (LIN, FIFO) 4 TX       */
#endif
    ICR26 = 31;    /* USART (LIN, FIFO) 5 RX       */
                   /* USART (LIN, FIFO) 5 TX       */
…
}
/*---------------------------------------------------------------------
  Prototypes

  Add your own prototypes here. Each vector definition needs is proto-
  type. Either do it here or include a header file containing them.
---------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler (void);
#ifdef ACC_DEBUGGER
__interrupt void acc_Receive_IRQ(void);
__interrupt void acc_Transmit_IRQ(void);
__interrupt void acc_InstructionBreak_IRQ(void);
#endif
/*---------------------------------------------------------------------
   Vector definiton

   Use following statements to define vectors. All resource related
   vectors are predefined. Remaining software interrupts can be added here
   as well.
---------------------------------------------------------------------*/
#pragma intvect 0xBFF8            0     /* (fixed) reset vector         */
#pragma intvect 0x06000000        1     /* (fixed) Mode Byte            */

#ifdef ACC_DEBUGGER
#pragma intvect acc_InstructionBreak_IRQ 10
#endif

#pragma intvect DefaultIRQHandler 65    /* System Reserved              */
…
#ifdef ACC_DEBUGGER
#pragma intvect acc_Receive_IRQ   66    /* USART (LIN, FIFO) 4 RX       */
#pragma intvect acc_Transmit_IRQ  67    /* USART (LIN, FIFO) 4 TX       */
#else
#pragma intvect DefaultIRQHandler 66    /* USART (LIN, FIFO) 4 RX       */
#pragma intvect DefaultIRQHandler 67    /* USART (LIN, FIFO) 4 TX       */
#endif

#pragma intvect DefaultIRQHandler 68    /* USART (LIN, FIFO) 5 RX       */
…
```

3.2.5 Adapting the Linker Settings (Changes in the Makefile)

Target of this example is to have minimum impact of the MDE 2008 into the linkage of the original application. To fulfil this requirement one can reduce the RAM/ROM area provided to the application (as long as this is possible). The freed parts of ROM and RAM will be reserved for the Accemic MDE 2008. When the MDE 2008 is not used in the application these memory areas will remain unused.

The definition of memory areas used by the linker is controlled by the linker options –ra (RAM area) and –ro (ROM area). For details on the Softune Workbench Linker options please refer to the Softune Workbench Linker Kit manual. In the current example (MB91F467B) the data RAM area is defined in the address range 0x2A000 to 0x2FFFF (linker option -ra D_RAM=0x0002A000/0x0002FFFF). This area will be reduced to the address range 0x2A000 to 0x2FEFF. The now remaining 256 bytes will be used to generate a new RAM area ACC_RAM_AREA in the address range 0x2FF00 to 0x2FFFF. The RAM area definitions now are:

-ra D_RAM=0x0002A000/0x0002FEFF

-ra ACC_RAM_AREA=0x0002FF00/0x2FFFF

The same is done with the ROM area. The existing ROM area ROM_AREA in the address range 0x40000 to 0x14FFFF is reduced and the then remaining ROM memory is used for an ACC_ROM_AREA. The linker options for this are:

-ro ROM_AREA=0x00040000/0x0014BFFF

-ro ACC_ROM_AREA=0x0014C000/0x0014FFFF

The Accemic MDE 2008 uses own section to place the code provided by the Accemic MDE 2008 debugger. These sections are:

acc_LIB_code

acc_LIB_const

acc_LIB_stack

Acc_LIB_data

The acc_LIB_code and acc_LIB_const section are to be placed in the ACC_ROM_AREA and acc_LIB_data and acc_LIB_stack have to be placed in the ACC_RAM_AREA. The placement of sections in given memory areas (defined by –ra and –ro option) is controlled by the linker option –sc. From this it follows that the section placement linker options needed here are:

-sc acc_LIB_data+acc_LIB_stack=ACC_RAM_AREA

-sc acc_LIB_code+acc_LIB_const=ACC_ROM_AREA

Finally the linker has to know where to find the libraries containing the code for the MDE 2008 debugger kernel and the code for the USART communication. These libraries are MonitorCOM_MB1V460_U4.lib and MonitorCore.lib. The linker option –L is used to tell the linker where to search for libraries and the option –l tells the linker which libraries are to be used. One has to add the following linker options (for the sample directory structure) to the linker settings:

-L ./acc_lib

-l MonitorCore.lib

-l MonitorCom_MB91V460_U4.lib

Please find in the Appendix 5.1 a complete sample makefile including the above described linker settings. This makefile can build the sample project for both cases (with or without the Accemic MDE 2008 debugger. The decision to include the debugger or not is done by setting the variable DEBUGGER=ACC_DEBUGGER when calling the make tool for including the MDE 2008 or calling the make tool without setting the variable DEBUGGER. See the two below examples (command line):

make –f makefile DEBUGGER=ACC_DEBUGGER -> Build Application with MDE 2008

make –f makefile -> Build Application without MDE 2008

4 Impact of Including MDE 2008 into an Application

This chapter will show the influence of the MDE 2008 onto the placement of the application in the memories of the MB91460 MCU. For this comparison the template project described in chapter 3 is used.

Please compare the extracts of the .mp1-file (output by the linker) generated for the sample application when not including the MDE 2008 and when the MDE 2008 debugger is included into the application

1. Debugger not included

```
0002A000-0002A3FB  000003FC  STACK P RW-- 04 REL  SSTACK
0002A000-........  00000000  DATA  P RW-- 04 REL  DATA
0002A000-........  00000000  DATA  P RW-- 04 REL  INIT
0002A3FC-0002A3FD  00000002  STACK P RW-- 04 REL  USTACK
00030000-........  00000000  CODE  P RWXI 04 REL  IDRAM
00040000-0004023B  0000023C  CODE  P R-XI 02 REL  CODE
0004023C-........  00000000  CONST P R--I 04 REL  CONST
0004023C-........  00000000  DATA  P R--- 04 REL #INIT
0004023C-........  00000000  CODE  P R-XI 04 REL #IDRAM
000F4000-000F421F  00000220  CODE  P R-XI 04 REL  CODE_START
000FFC00-000FFFFF  00000400  CONST P R--I 04 REL  INTVECT
00148000-0014800F  00000010  CODE  N R-XI 00 ABS  SECURITY_VECTORS
```

2. MDE 2008 Debugger included

```
0002A000-0002A3FB  000003FC  STACK P RW-- 04 REL  SSTACK
0002A000-........  00000000  DATA  P RW-- 04 REL  DATA
0002A000-........  00000000  DATA  P RW-- 04 REL  INIT
0002A3FC-0002A3FD  00000002  STACK P RW-- 04 REL  USTACK
0002FF00-0002FF1E  0000001F  DATA  P RW-- 04 REL  ACC_LIB_DATA
0002FF20-0002FFD7  000000B8  STACK P RW-- 04 REL  ACC_LIB_STACK
00030000-........  00000000  CODE  P RWXI 04 REL  IDRAM
00040000-000402D7  000002D8  CODE  P R-XI 02 REL  CODE
000402D8-........  00000000  CONST P R--I 04 REL  CONST
000402D8-........  00000000  DATA  P R--- 04 REL #INIT
000402D8-........  00000000  CODE  P R-XI 04 REL #IDRAM
000F4000-000F421F  00000220  CODE  P R-XI 04 REL  CODE_START
000FFC00-000FFFFF  00000400  CONST P R--I 04 REL  INTVECT
00148000-0014800F  00000010  CODE  N R-XI 00 ABS  SECURITY_VECTORS
0014C000-0014C6E7  000006E8  CONST P R--I 04 REL  ACC_LIB_CONST
0014C6E8-0014D2C9  00000BE2  CODE  P R-XI 02 REL  ACC_LIB_CODE
```

The differences of the mappings are shown in red in the map list for the debugger included. One finds the sections acc_LIB_xxx placed in the corresponding RAM/ROM areas (this has no impact on the mapping of the application). Linking of standard library functions

__STD_LIB__divi

__STD_LIB__udivi

__STD_LIB_abs

needed by the MDE 2008 will increase the size of the code section and this has indeed influence on the linkage of the application.

5 Appendix

5.1 Complete Sample Makefile for Building the Sample Application

```
# Sample make file
TOOLDIR = U:\bin

# Directory Structure
OBJECTDIR = ./obj
ACC_OBJECTDIR = ./acc_obj
SOURCEDIR = ./src
ACC_SOURCEDIR = ./acc_src
INCLUDEDIR = ./inc
ACC_INCLUDEDIR = ./acc_inc
LIBDIR = ./lib
ACC_LIBDIR = ./acc_lib
BUILDDIR = ./build
LISTDIR = ./lst

# Language Tools
CC = fcc911s.exe
ASM = fasm911s.exe
LINK = flnk911s.exe
CONVERT = f2ms.exe

# Options
ifeq ($(DEBUGGER),ACC_DEBUGGER)
CCOPT = -c -g -cpu MB91F467B -I $(INCLUDEDIR) -I $(ACC_INCLUDEDIR) -DACC_DEBUGGER
ASMOPT = -g -cpu MB91F467B -I $(INCLUDEDIR) -D$(DEBUGGER)
LINKOPT = -cpu MB91F467B -g -AL 2 -Xset_rora -ra D_RAM=0x0002A000/0x0002FEFF          \
          -ra ID_RAM=0x00030000/0x00033FFF -ro ROM_AREA=0x00040000/0x0014BFFF         \
          -sc DATA/Data+INIT/Data+SSTACK/Data+USTACK/Data=D_RAM -sc IDRAM/Code=ID_RAM \
          -sc CODE+@INIT+@IDRAM+CONST=ROM_AREA -sc CODE_START/Code=0x000F4000         \
          -sc INTVECT/Const=0x000FFC00 -ro ACC_ROM_AREA=0x0014C000/0x0014FFFF        \
          -ra ACC_RAM_AREA=0x0002FF00/0x0002FFFF                                     \
          -sc ACC_LIB_CONST+ACC_LIB_CODE=ACC_ROM_AREA                                \
          -sc ACC_LIB_DATA+ACC_LIB_STACK=ACC_RAM_AREA -L $(ACC_LIBDIR)               \
          -l MonitorCore.lib -l MonitorCom_MB91V460_U4.lib
else
CCOPT = -c -g -cpu MB91F467B -I $(INCLUDEDIR)
ASMOPT = -g -cpu MB91F467B -I $(INCLUDEDIR)
LINKOPT = -cpu MB91F467B -g -AL 2 -Xset_rora -ra D_RAM=0x0002A000/0x0002FEFF          \
          -ra ID_RAM=0x00030000/0x00033FFF -ro ROM_AREA=0x00040000/0x0014BFFF         \
```

```
            -sc DATA/Data+INIT/Data+SSTACK/Data+USTACK/Data=D_RAM -sc IDRAM/Code=ID_RAM \
            -sc CODE+@INIT+@IDRAM+CONST=ROM_AREA                                       \
            -sc CODE_START/Code=0x000F4000 -sc INTVECT/Const=0x000FFC00
endif

CONVERTOPT = -S3

# List of Object Files

ifeq ($(DEBUGGER),ACC_DEBUGGER)
ACC_OBJECTS = $(ACC_OBJECTDIR)/MDE_MB91V460.obj
else
ACC_OBJECTS =
endif


OBJECTS = $(OBJECTDIR)/main.obj $(OBJECTDIR)/vectors.obj $(OBJECTDIR)/mb91467b.obj
$(OBJECTDIR)/start91460.obj $(ACC_OBJECTS)



# Rules
output: $(OBJECTS)
        $(TOOLDIR)\$(LINK) $(LINKOPT) $(OBJECTS) -m $(LISTDIR)/output.mp1 \
         -o $(BUILDDIR)/output.abs
        $(TOOLDIR)\$(CONVERT) $(CONVERTOPT) -o $(BUILDDIR)/output.mhx $(BUILDDIR)/output.abs

$(OBJECTDIR)/%.obj: $(SOURCEDIR)/%.asm
        $(TOOLDIR)\$(ASM) $(ASMOPT) -o $@ $<

$(OBJECTDIR)/%.obj: $(SOURCEDIR)/%.c
        $(TOOLDIR)\$(CC) $(CCOPT) -o $@ $<

$(ACC_OBJECTDIR)/%.obj: $(ACC_SOURCEDIR)/%.c
        $(TOOLDIR)\$(CC) $(CCOPT) -o $@ $<

.PHONY: clean
clean:
        -rm -rf $(OBJECTDIR)/*.obj
        -rm -rf $(ACC_OBJECTDIR)/*.obj
        -rm -rf $(LISTDIR)/*.*
        -rm -rf $(BUILDDIR)/output.abs
        -rm -rf $(BUILDDIR)/output.mhx
```

5.2 Sample for removing the call to acc_InitKernel() from mhx-File

As discussed in 3.2.3 an alternative way to remove the MDE 2008 debugger kernel from the final release of an application is to directly change the load module file (e.g. the mhx-file) of the application including the MDE 2008 debugger. Again as already mentioned in 3.2.3 using such a procedure to remove the MDE 2008 debugger kernel the user needs to have a clear understanding on the structure of the load module file and the memory mapping of the application. **Cypress will not take any liability on the resulting code**.

To explain the procedure please find in the following as an example an extract from output.mhx as generated when the call to acc_InitKernel() is included. The memory locations of the symbol _start (entry point to application) and acc_InitKernel can be found in the linker output .mp1-file:

```
000F4000(ABS)     Addr.  OM/LM @__start
                    _start
0014C744(ABS)     Addr.  LIB    _acc_InitKernel
                    acc_InitKernel
```

Figure shows the changes to be done in the mhx-file to replace the two instructions LDI:32 #acc_InitKernel, R0 and CALL @R0 by a corresponding number of NOP instructions. For details on the instruction opcodes please refer to the FR Programming Manual and for details on the structure of mhx S-Records please refer to Softune Workbench Linker Kit Manual.

Similar changes might need to be done to remove the code from the sections reserved for the Accemic MDE 2008 debugger (memory locations of `acc_LIB_code` and `acc_LIB_const)`. Important is to change also the vector table (INTVECT section). Here the interrupt vectors no. 10 (EDSU Instruction Break IRQ), no. 66 (USART4 RX IRQ) and no. 67 (USART TX) have to be changed to point to the default (not handled) IRQ handler to ensure correct behaviour of the application even when the MDE 2008 kernel is removed.

Please ensure that the resulting application (MDE 2008 Kernel removed) is extensively tested to avoid any possible side effect caused by removing the MDE 2008 Kernel without re-building of the application.

```
Extract from mhx-file in the address range 0xF4000 - 0xF402F
including the call to acc_InitKernel (0x14C744):

S315000F40009FA083EF871F9B0704C7807793209F8FFF
S315000F40100002A3FE83DF9F8F0002A3FC9F80001484
S315000F4020C74497109F80000FFC00B3009B0004ADA0

resulting memory layout:
------------------------

000F4000: 9FA0             NOP
000F4002: 83EF             ANDCCR   #EF
000F4004: 871F             STILM    #1F
000F4006: 9B0704C7         LDI:20   #004C7,R7
000F400A: 8077             BANDL    #7,@R7
000F400C: 9320             ORCCR    #20
000F400E: 9F8F0002A3FE     LDI:32   #0002A3FE,R15
000F4014: 83DF             ANDCCR   #DF
000F4016: 9F8F0002A3FC     LDI:32   #0002A3FC,R15
000F401C: 9F800014C744     LDI:32   #0014C744,R0
000F4022: 9710             CALL     @R0
000F4024: 9F80000FFC00     LDI:32   #000FFC00,R0
000F402A: B300             MOV      R0,TBR
000F402C: 9B0004AD         LDI:20   #004AD,R0


Changing the mhx-file at the corresponding address range
(0xF401C - 0xF4023):

S315000F40009FA083EF871F9B0704C7807793209F8FFF
S315000F40100002A3FE83DF9F8F0002A3FC9FA09FA039
S315000F40209FA09FA09F80000FFC00B3009B0004ADD4

resulting memory layout:
------------------------

000F4000: 9FA0             NOP
000F4002: 83EF             ANDCCR   #EF
000F4004: 871F             STILM    #1F
000F4006: 9B0704C7         LDI:20   #004C7,R7
000F400A: 8077             BANDL    #7,@R7
000F400C: 9320             ORCCR    #20
000F400E: 9F8F0002A3FE     LDI:32   #0002A3FE,R15
000F4014: 83DF             ANDCCR   #DF
000F4016: 9F8F0002A3FC     LDI:32   #0002A3FC,R15
000F401C: 9FA0             NOP
000F401E: 9FA0             NOP
000F4020: 9FA0             NOP
000F4022: 9FA0             NOP
000F4024: 9F80000FFC00     LDI:32   #000FFC00,R0
000F402A: B300             MOV      R0,TBR
000F402C: 9B0004AD         LDI:20   #004AD,R0
```

Document History

Document Title: AN205436 - FR, MB91460, Accemic MDE 2008 Integration

Document Number:002-05436

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	09/03/2009	V1.0 RSCHUM First draft
			10/07/2009	V1.1 RSCHUM -including ACCMIC description of how to integrate MDE 2008 into an existing application -including information about alternative way to remove MDE 2008 debugger (avoid re-build of application)
*A	5137517	NOFL	02/15/2016	Converted Spansion Application Note “MCU-AN-300114-E-V11” to Cypress format
*B	5870078	AESATP12	09/01/2017	Updated logo and copyright.

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

Cypress Developer Community

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

Technical Support

cypress.com/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, 2009-2017. 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. 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.