

**FR FAMILY**  
32-BIT MICROCONTROLLER  
**MB91460**

---

**ACCEMIC MDE 2008**  
**INTEGRATION**

APPLICATION NOTE



## Revision History

Date	Issue
2009-09-03	V1.0 RSCHUM First draft
2009-10-07	V1.1 RSCHUM <ul style="list-style-type: none"> <li>- including ACCMIC description of how to integrate MDE 2008 into an existing application</li> <li>- including information about alternative way to remove MDE 2008 debugger (avoid re-build of application)</li> </ul>

This document contains 22 pages.

## Warranty and Disclaimer

The use of **the deliverables** (e.g. software, application examples, target boards, evaluation boards, starter kits, schematics, engineering samples of IC's etc.) is subject to the conditions of Fujitsu Microelectronics Europe GmbH ("FME") as set out in (i) the terms of the License Agreement and/or the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials.

Please note that the deliverables are intended for and must only be used in an evaluation laboratory environment.

The software deliverables are provided without charge and therefore provided on an as-is basis. The software deliverables are to be used exclusively in connection with FME products.

Regarding hardware deliverables, FME warrants that they will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

Should a hardware deliverable turn out to be defect, FME's entire liability and the customer's exclusive remedy shall be, at FME's sole discretion, either return of the purchase price and the license fee, or replacement of the hardware deliverable or parts thereof, if the deliverable is returned to FME in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to FME, or abuse or misapplication attributable to the customer or any other third party not relating to FME or to unauthorised decompiling and/or reverse engineering and/or disassembling.

FME does not warrant that the deliverables does not infringe any third party intellectual property right (IPR). In the event that the deliverables infringe a third party IPR it is the sole responsibility of the customer to obtain necessary licenses to continue the usage of the deliverable.

In the event the software deliverables include the use of open source components, the provisions of the governing open source license agreement shall apply with respect to such software deliverables.

To the maximum extent permitted by applicable law FME disclaims all other warranties, whether express or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the deliverables are not designated.

To the maximum extent permitted by applicable law, FME's liability is restricted to intention and gross negligence. FME is not liable for consequential damages.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

The contents of this document are subject to change without a prior notice, thus contact FME about the latest one.

## Contents

<b>REVISION HISTORY .....</b>	<b>2</b>
<b>WARRANTY AND DISCLAIMER .....</b>	<b>3</b>
<b>CONTENTS .....</b>	<b>4</b>
<b>1 INTRODUCTION .....</b>	<b>5</b>
<b>2 BASIC MDE 2008 INTEGRATION INTO AN EXISTING APPLICATION .....</b>	<b>6</b>
2.1 General remarks.....	6
2.1.1 Ressources.....	6
2.2 Template Project .....	6
2.3 Preparing an application for use with Accemic MDE.....	7
<b>3 MDE 2008 INTEGRATION INTO AN EXISTING APPLICATION .....</b>	<b>10</b>
3.1 Example for a File/Directory Structure .....	10
3.2 Modifications of the Application for MDE2008 .....	10
3.2.1 Adding the MDE 2008 Libraries, Source Files and Header Files to the Application.....	12
3.2.2 Adapting the Compiler and Assembler Settings.....	12
3.2.3 Initialization of ACCEMIC MDE 2008 Kernel.....	12
3.2.4 Adding the needed entries for MDE 2008 to the Application Vector Table	13
3.2.5 Adapting the Linker Settings (Changes in the Makefile).....	16
<b>4 IMPACT OF INCLUDING MDE 2008 INTO AN APPLICATION .....</b>	<b>18</b>
<b>5 APPENDIX.....</b>	<b>20</b>
5.1 Complete Sample Makefile for Building the Sample Application .....	20
5.2 Sample for removing the call to acc_InitKernel() from mhX-File .....	21

# 1 Introduction

In the following a description is given how to generally include the Accemic MDE 2008 into an application written for Fujitsu MB91460 Series using the language tools of the Fujitsu 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.

## 2 Basic MDE 2008 integration into an existing application

---

This chapter will overview 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 your 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

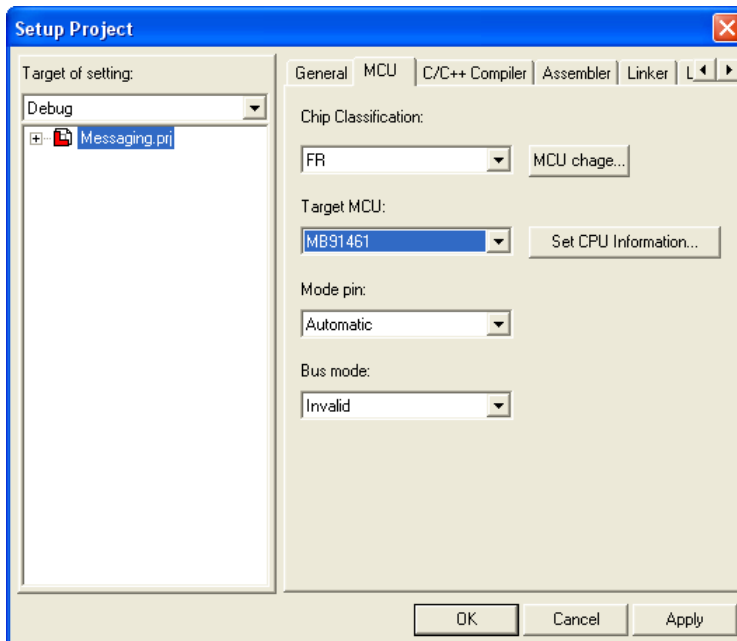
### 2.2 Template Project


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
  1. Change the access key in acc\_KernelConfig
  2. Change the main oscillator frequency in acc\_ComConfig
  3. Change the sub oscillator frequency in acc\_ComConfig
  4. 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:
    1. The RX interrupt of your used UART to acc\_Receive\_IRQ
    2. The TX interrupt of your used UART to acc\_Transmit\_IRQ
    3. 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.
  11. 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.
  12. It is advisable to disable all optimization options in the C Compiler preferences.
  13. These options can be set under “Project → Setup Tool Options → C Compiler → Category: Optimize” in the FR Family Softune Workbench environment.
  14. Build the project by clicking the “Build” button or by performing the command Project - Build in the Softune Workbench.
  15. Add  the generated file '[Accemic MDE Directory]\sources\Fujitsu\FR\Monitor\Templates \[Family name]\Debug\ABS\template\_XXX.abs' to your project browser.
  16. Drag & Drop the template\_XXX.abs from the Project Navigator to the prepared processor in the Processor Navigator.
  17. 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 Fujitsu 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 from a standard Softune Workbench template project from 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 be 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 Figure 1 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.

```
# 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.mpl -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
        -rm -rf $(BUILDDIR)/output.mhx
```

Figure 1: Sample makefile for GNU make tool.

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 be 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 Figure 2 for an example how 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 Fujitsu 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.

```

;=====
; 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
    
```

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

### 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          */
#ifdef 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
000FFC00-000FFFFFF 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
000FFC00-000FFFFFF 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. **Fujitsu 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
```

-- END --