Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec No: 002-05258

Spec Title: AN205258 - F2MC - 8FX FAMILY, MB951XX, SOFTUNE WORKBENCH MONITOR DEBUGGER FOR 8FX

Replaced By: NONE

AN205258

F²MC - 8FX Family, MB951XX, SOFTUNE Workbench Monitor Debugger for 8FX

This application note describes the Softune Workbench Monitor Debugger for the 8FX family. The Softune Workbench Monitor Debugger gives the possibility to control the program execution and view internal registers and memory areas directly on a Flash chip built in a target application or starter kit. Therefore a monitor kernel has to be included in the software project flashed to the target system which provides debugging functions via communication with a host computer.

OBSOLETE

Contents

1 Introduction 1
2 Hardware Connection 3
2.1 BGM Adapter 3
2.2 Connector on Target Board 5
2.3 Starter kit “CONCERTO-Kit” 7
3 Software Project 8
3.1 Overview on needed resources 8
3.2 Starting with Template Project 10
3.3 Integrating in existing project 17
4 Monitor Debugger 20
4.1 Flash Application to Target 20
4.2 Starting Debug Session 21
4.3 Features of Monitor Debugger 27
5 Prohibitions and Restrictions 28
5.1 Prohibitions 28
5.2 Restrictions 28
Document History 29

1 Introduction

This application note describes the Softune Workbench Monitor Debugger for the 8FX family. The Softune Workbench Monitor Debugger gives the possibility to control the program execution and view internal registers and memory areas directly on a Flash chip built in a target application or starterkit. Therefore a monitor kernel has to be included in the software project flashed to the target system which provides debugging functions via communication with a host computer.

As there is no MCU board with evaluation chip MB95FV100D-101/103 needed, it is a quite cheap solution for debugging.

In this document the hardware requirements as well as needed settings in the software project are shown. Furthermore, the usage of template project for debugging with the monitor debugger is explained.

Last chapter gives an overview about the debugging features of the Softune Workbench Monitor Debugger.

For an easy start with the Softune Workbench Monitor Debugger, please make sure to have the following hardware and software tools available:

- BGM adapter MB2146-09A-E
- Softune Workbench for 8L/8FX (V30L30 or later version)
- Flash Programmer: ‘Fujitsu Flash MCU Programmer for F²MC-8FX’ (current version V01L04) or ‘Cypress Flash BGM Programmer’ (current version V01L08)

- Software Template project depending on your target device:
  - ☐ 95100_template_bassoon_mb95f108-v12
  - ☐ 95110_template_clarinet_mb95f118-v12
  - ☐ 95120_template_tuba_mb95f128-v12
  - ☐ 95130_template_flute_mb95f136-v12
  - ☐ 95140_template_oboe_mb95f146-v11
  - ☐ 95150_template_trumpet_mb95f156-v11
  - ☐ 95160_template_trombone_mb95f168-v10
  - ☐ 951xx_template_concertokit-v14

(or later version)

Latest versions of programming tools and software examples/templates can be found on the Cypress Semiconductor website: http://www.cypress.com/cypress-mcu-product-softwareexamples

For details on Monitor Debugger Functions also refer to the Softune Workbench documentation:

- Softune Workbench User's Manual
- Softune Workbench Operation Manual

OBSOLETE

2 Hardware Connection

This chapter lists the needed hardware tools and the connection to the microcontroller.

2.1 BGM Adapter

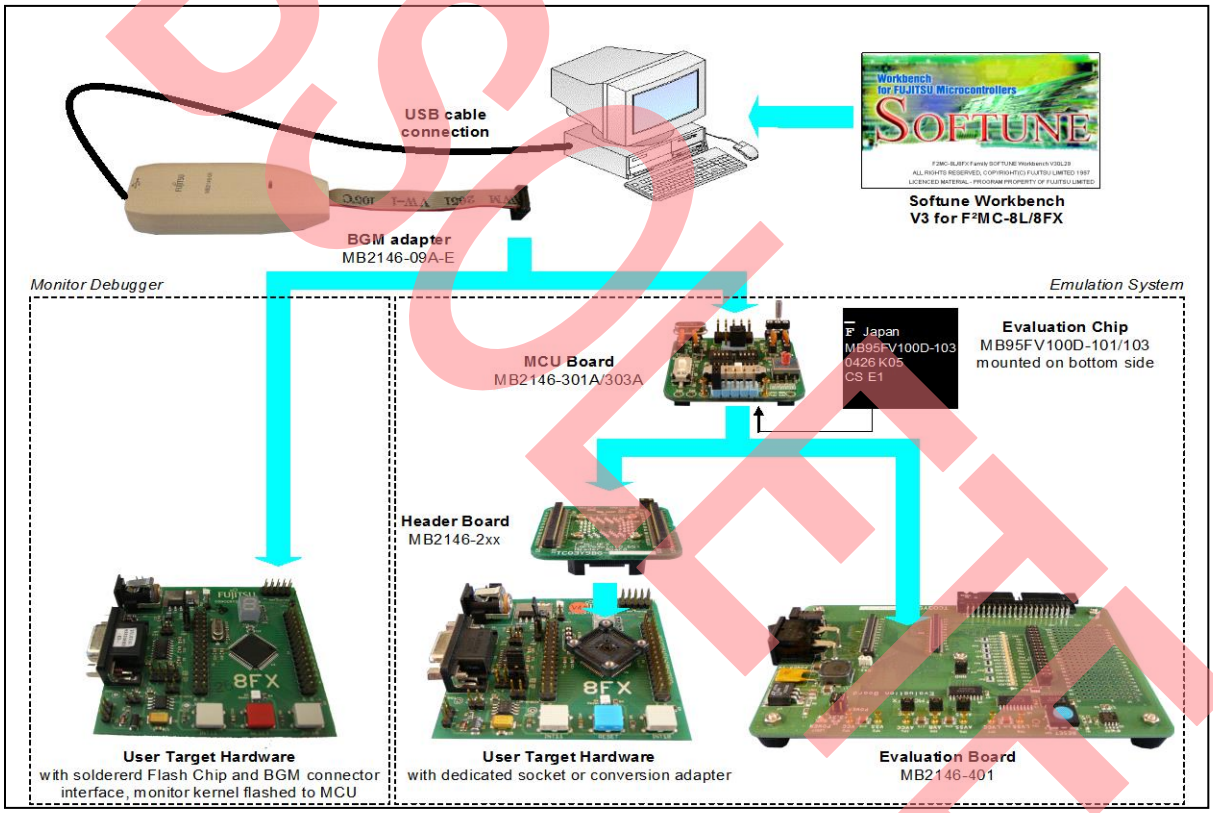
The BGM adapter MB2146-09A-E can be used as in-circuit emulator, serial synchronous flashing tool and USB-serial converter for the 8FX Softune Workbench Monitor Debugger.

Error! Reference source not found. shows an overview of these possibilities regarding debugging systems with the Softune Workbench IDE for 8FX.

For building up a complete emulation system, further hardware tools like a MCU board, an evaluation board or a header board are needed. For further details on the emulation system, please see following application note:

- mcu-an-395002-e-8fx_mb2146_09_emulator_hw_setup

Figure 1. Possible system configurations for debugging with the Softune Workbench IDE



For using the Softune Workbench Monitor Debugger, only the MB2146-09A-E is needed. The MB2146-09A-E is connected to the PC via USB cable. Connection to target hardware is done via 10pin BGM connector.

For details, refer the following manual:

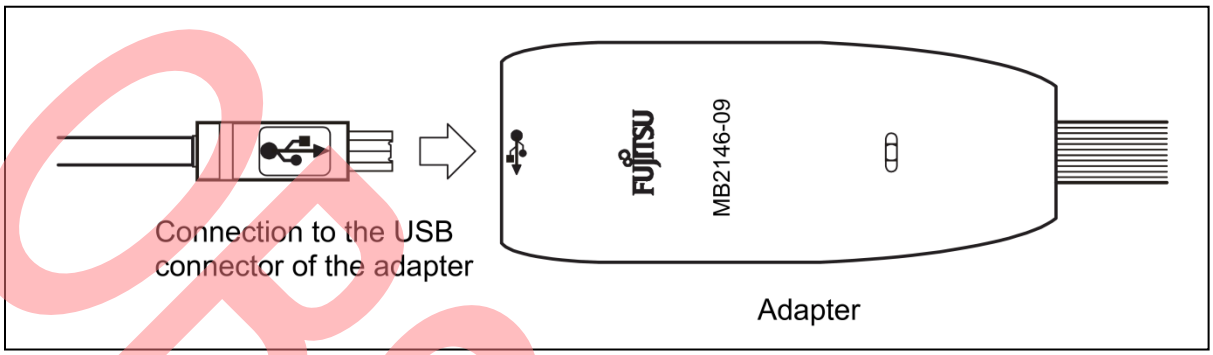
- BGM Adapter MB2146-09A-E Operation Manual

OBSOLETE

2.1.1 Connection to the Host Machine

Connect the adapter to the host machine using the USB cable.

Figure 2. Connecting the USB Cable



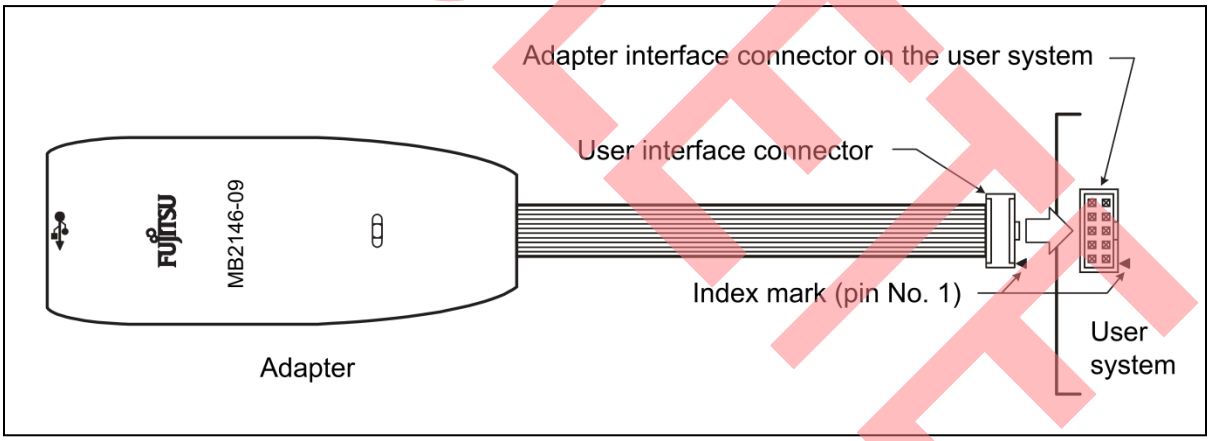
If the BGM adapter is connected the first time to the host machine, it is possible the operation system searches for a valid device driver. This driver can be found in your Softune installation directory in subfolder [Drivers], for example "C:\Softune\Drivers\SiUSBdB.inf".

2.1.2 Connection to the User System

Connect the adapter to the user system. Plug the user interface connector of the adapter into the adapter interface connector on the user system.

When plugging the user interface connector, align its index mark (pin no. 1) with the adapter interface connector's counterpart.

Figure 3. Connection to the User System



Use only BGM adapter MB2146-09A-E with the Softune Workbench Monitor Debugger. Previous version MB2146-09 does not support the Monitor Debugger Feature!

OBSOLETE

2.2 Connector on Target Board

2.2.1 Adapter Interface Specifications

Table 1 shows the pin out of the adapter interface connector to be mounted on the user system. Figure 4 shows the connector pins. **Error! Reference source not found.** shows a list of recommended interface connectors. Use one of these devices or similar connector on your target hardware.

Table 1. Adapter Interface Connector Pin out

Connector pin No.	MCU Pin Name	Input/output	Remarks
1	VCC	BGMA ← MCU	User power supply input
2	VSS	—	MCU GND
3	RSTX	BGMA → MCU	Tool reset output
4	N.C		Not connected
5	UO0	BGMA ← MCU	Serial data input (BGMA)
6	UCK0	BGMA → MCU	Synchronous Clock Output (BGMA)
7	UI0	BGMA → MCU	Serial data output (BGMA)
8	N.C	—	Not connected
9	GND	—	MCU GND (can be unconnected)
10	VCC	—	User power supply input (can be unconnected)

*: “BGMA” in the “Input/output” column in the table indicates the BGM adapter.

Figure 4. Adapter Interface Connector Pins

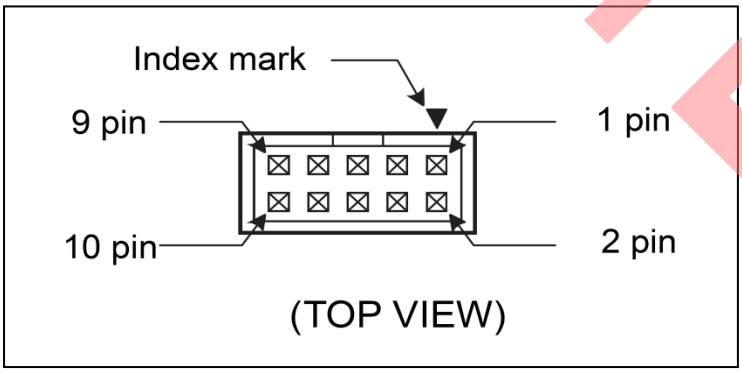


Table 2. Recommended adapter interface connectors

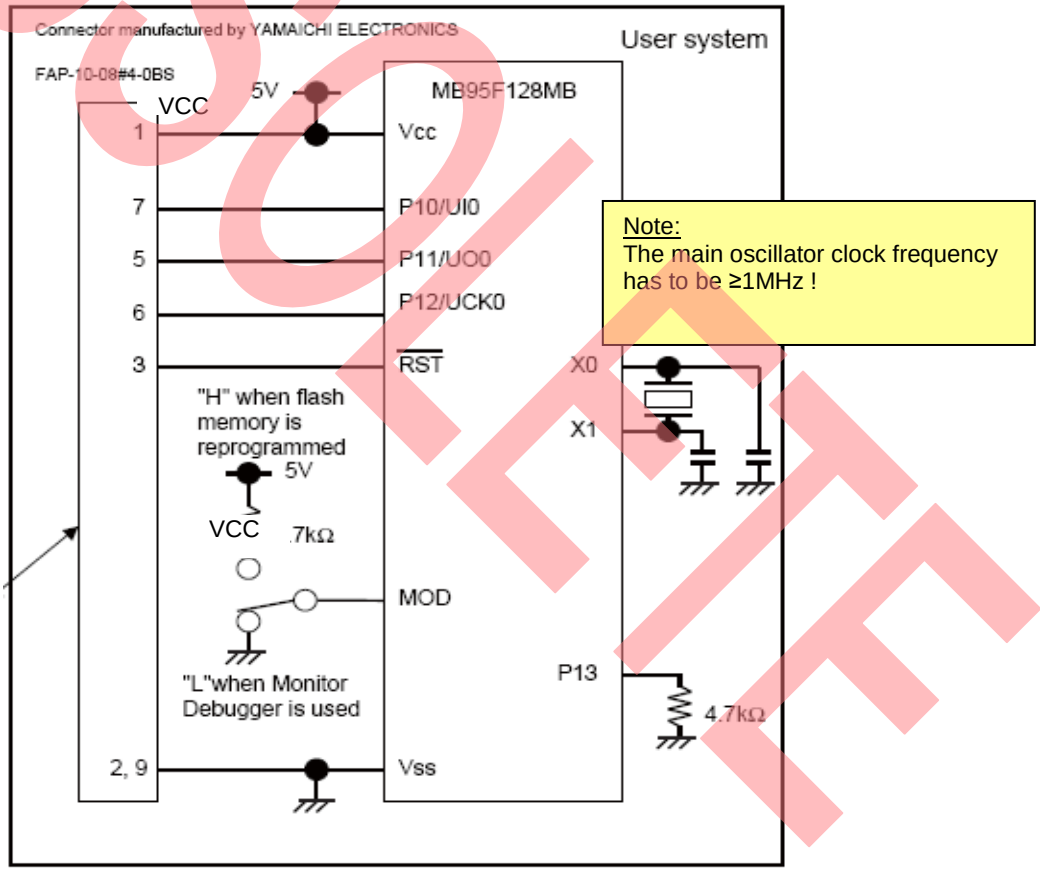
OBSOLETE

Part number	Specifications		Manufacturer
FAP-1001-2202-0BF	Right-angle solder dip	Housing provided, Middle latch provided	YAMAICHI ELECTRONICS Co., Ltd.
FAP-1001-2204-0BF	Straight solder dip	Housing provided, Middle latch provided	
FAP-10-08#2-0BF	Right-angle solder dip	Housing provided, Latch not provided	
FAP-10-08#4-0BF	Straight solder dip	Housing provided, Latch not provided	

2.2.2 Connection to Microcontroller

Use following connection of 10pin BGM connector to MB95F1xx MCU (MB95F128MB shown, but same for other 8FX flash devices).

Figure 5. Connection of programmer connector to MCU



By applying this connection, MB2146-09A-E can also be used as serial synchronous flash programming tool. See following application note for details:

- mcu-an-300050-e-8fx_sync_flash_programming

OBSOLETE

2.3 Starter kit “CONCERTO-Kit”

The CONCERTO-Kit is a small evaluation board for the 8FX family with a soldered MB95F108HS Flash chip. Connector X6 on the starter kit is used as BGM interface for serial synchronous programming or for use of the Softune Workbench Monitor Debugger.

Figure 6 shows the CONCERTO-Kit with connected BGM adapter MB2146-09A-E. This hardware is used as target system for creating this application note.

Figure 6. CONCERTO-Kit with connected BGM adapter MB2146-09A-E



If using CONCERTO-Kit from Cypress please make sure that on-board RS232 driver on starterkit does not drive against BGM adapter signals. Therefore make sure jumpers JP4 TXD and JP5 RXD are not closed in position 3-4!

OBSOLETE

3 Software Project

This chapter explains the needed settings to be implemented in software project to use the Softune Workbench Monitor Debugger.

3.1 Overview on needed resources

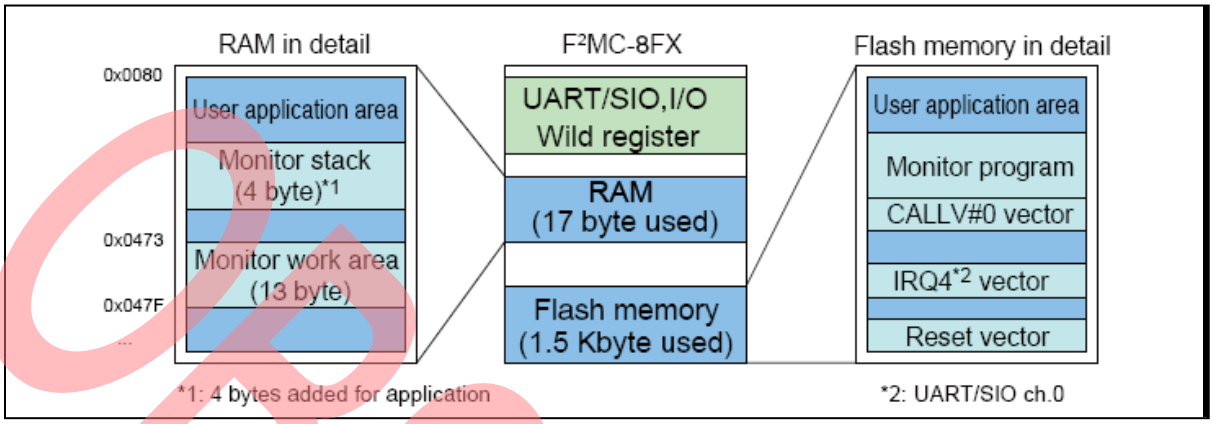
The Softune Workbench Monitor Debugger kernel uses some resources of the microcontroller that therefore cannot be used in user application. Please take care that you do not modify registers of these resources inside the application.

Table 3. Resources used by the Softune Workbench Monitor Debugger Kernel

Resource	Details	Remarks
UART/SIO	UART/SIO channel 0 is used for communication with host program	SMC10 SMC20 SSR0 RDR0 TDR0 IRQ4 interrupt vector
CALLV#0	CALLV#0 instruction is used for breakpoint handler	CALLV#0 vector
UART/SIO Baud rate Generator	UART/SIO channel 0 is used for communication with host program	PSSR0 BRSR0
I/O Ports	I/O Ports shared with UART/SIO channel 0 pins are used for UART/SIO communication	P10/UI0 P11/UO0 P12/UCK0
Wild Register	Wild registers are used for breakpoint detection	WRDR0-2 WRAR0-2 WREN WROR
RAM	17 bytes of RAM are used for monitor debugger kernel 0x0473-0x047F (13 bytes) Stack (4 bytes)	13 bytes used for monitor kernel work area 4 bytes on stack area are put on application stack, add these 4 bytes to application stack size
Flash (ROM)	About 1.5 kbytes of Flash are used for monitor debugger kernel 0xFA8C*-0xFFB0 0xFFB1-0xFFB5 0xFFB6-0xFFBF 0xFFC0-0xFFC1 0xFFF2-0xFFF3 0xFFFC-0xFFFF	* start address of monitor code may vary depending on version of monitor kernel Monitor kernel code FGM ID Table FGM Configuration Table CALLV#0 vector UART/SIO ch 0 ISR vector Reset Vector

OBSOLETE

Figure 7. Outline of memory usage of monitor kernel

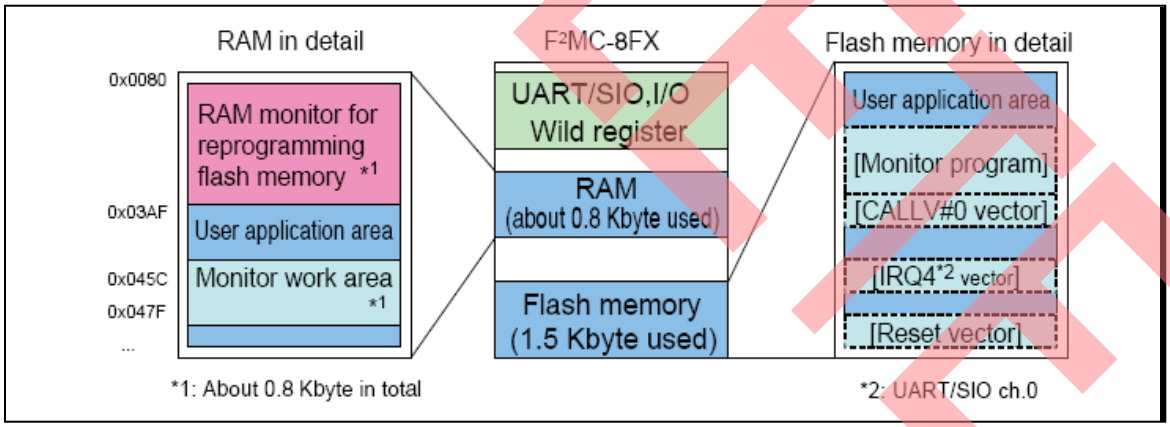


When (re-)loading the user application by use of the monitor kernel, the amount of RAM needed is increased. As in this case there is no execution of user program code, this causes no further limitations to the user application.

Table 4. RAM area used for reprogramming flash memory

RAM	About 0.8 Kbytes • 0x0080-0x03AF • 0x045C-0x0472	• RAM code for reprogramming flash memory • Monitor kernel work area

Figure 8. Outline of memory usage when loading

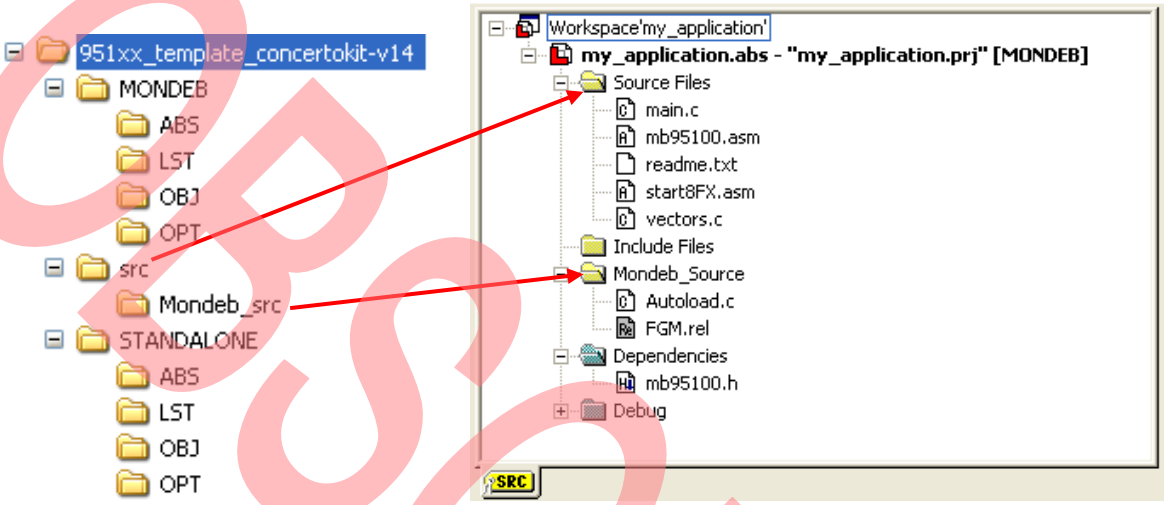


OBSOLETE

3.2 Starting with Template Project

For a quick and easy start up of an own software project, Cypress offers template projects for each MCU series. The latest 8FX template projects include a configuration for working with the Softune Workbench Monitor Debugger.

Figure 9. Overview on CONCERTO-Kit template project



Please refer to application note mcu-an-395001-e-8fx_swb_getting_started for details on how to start on template project.

3.2.1 Project Settings in MONDEB Configuration

The template project offers two configuration settings: STANDALONE is used for generating code to be flashed to flash microcontroller for standalone usage or for use with the emulation system. MONDEB is used for generating code to be debugged with the Softune Workbench Monitor Debugger. Please select MONDEB configuration.

Figure 10. Selection menu for project configuration



MONDEB Configuration includes two additional source files in folder Mondeb_Source. These source files include the monitor kernel. In STANDALONE configuration, these source files are not included (symbol is crossed).

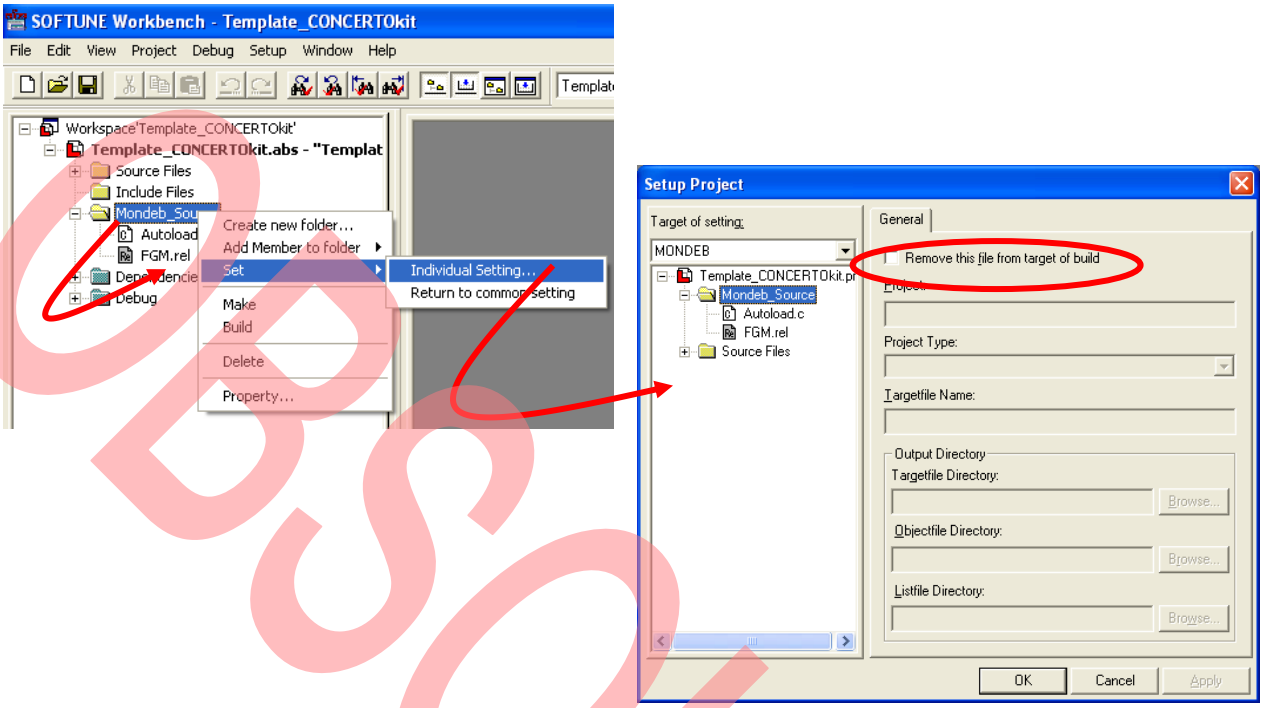
Whether or not these files should be included can be selected in following menu:

Right click on folder ‘Mondeb_Source’ in project window > Set > Individual Setting… > check/uncheck option ‘Remove this file from target of build’

It is also possible in this way to do the settings for single source files. Make sure that this option is not checked for files ‘Autoload.c’ and ‘FGM.rel’ for using monitor debugger.

OBSOLETE

Figure 11. Add/Remove files from target of build



To implement preconfigured settings and code parts from ‘Start8FX.asm’ and ‘Vectors.c’ which are needed for the monitor debugger execution, please add macro definition “MONITORDEBUGGER_CONFIGURATION” to C-compiler and assembler:

Project > Setup Project… > Tab: C Compiler > Category: Define Macro

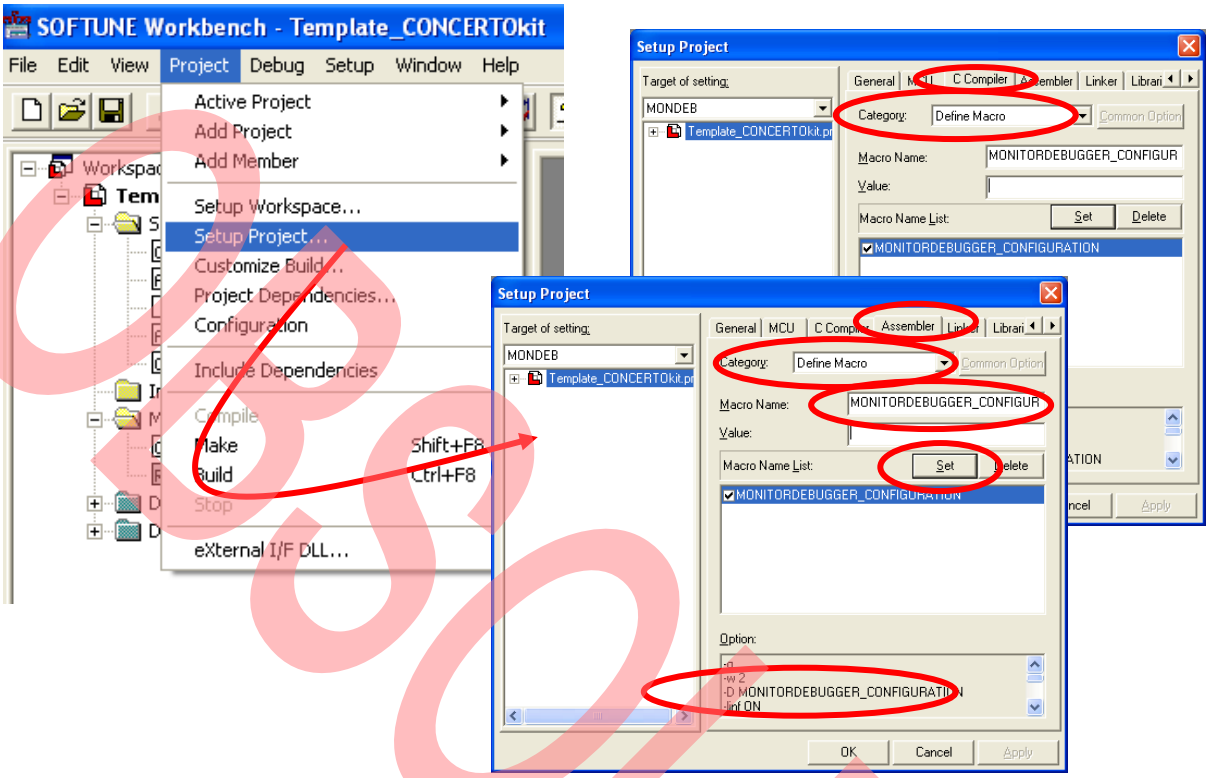
Project > Setup Project… > Tab: Assembler > Category: Define Macro

Enter macro name “MONITORDEBUGGER_CONFIGURATION” and click “Set” button (needs to be done separately for compiler and assembler).

The option ‘-D MONITORDEBUGGER_CONFIGURATION’ is added to both C-compiler and assembler options shown in option window of setup project dialogue.

OBSOLETE

Figure 12. Add macro definition



3.2.2 Start8FX.asm

The startup file contains some settings and code parts for use of the Softune Workbench Monitor Debugger which are explained in this chapter.

The section 4 “Settings” provides the possibility to enter a 32bit password which only allows users knowing the password to debug the application. Therefore change the default value 0xFFFFFFFF under 4.9 “Monitor Debugger” to a user-defined value. This password is checked when connecting the Softune Workbench to the monitor kernel on MCU when starting a debug session. (see chapter 0 for details).

Note: This setting is only relevant for MONDEB configuration.

```
;------------------------------------------------------------------------
; 4.9 Monitor Debugger
;------------------------------------------------------------------------
;
#set   USER_PASSWORD    0xFFFFFFFF  ; <<< Define User Password for
;                                          monitor debugger entry (32bit)
;
```

The next sections are included only if the macro definition like described in chapter 3.2.1 before is included. Normally you do not have to change any of these items.

OBSOLETE

```
;=====================================================================
; 5.2  External declaration of symbols
;=====================================================================
#ifdef MONITORDEBUGGER_CONFIGURATION
        .IMPORT     _AutoBootCheck
        .EXPORT     _FGMTBL_STARTADR
        .EXPORT     _FGMTBL_STACKTOP
        .EXPORT     _FGMTBL_ABOOTADR
        .EXPORT     _FGMTBL_PASSWORD
#endif
```

```
;---------------------------------------------------------------------
; 5.5 Monitor Debugger configuration data
;---------------------------------------------------------------------
#ifdef MONITORDEBUGGER_CONFIGURATION

        .SECTION    FGMCFGTBL,  CONST,  LOCATE=0xFFB6
; configuration table
_FGMTBL_STARTADR:
        .DATA.H     __start         ; Start address of user application

_FGMTBL_STACKTOP:
        .DATA.H     STACK_TOP       ; Stack top

_FGMTBL_ABOOTADR:
        .DATA.H     _AutoBootCheck  ; Auto-boot address

_FGMTBL_PASSWORD:
        .DATA.L     USER_PASSWORD   ; password (4byte)

; CALLV vector for FGM
        .SECTION    FGMCALLVECT,    CONST,  LOCATE=0xFFC0
            .IMPORT     _FGM_CALL

    .DATA.H     _FGM_CALL           ; CALLV #0
#endif
```

```
;-----------------------------------------------------------
; 5.6 Define Reset vector and Modebyte
;-----------------------------------------------------------
; The Mode Byte is defined at the beginning of the start.asm

      .SECTION  RESVECT, CONST, LOCATE=H'FFFD
      .DATA.B MODEBYTE
    #ifdef MONITORDEBUGGER_CONFIGURATION
      .IMPORT _FGM_INIT
      .DATA.W _FGM_INIT
    #else
      .DATA.W __start
    #endif
```

OBSOLETE

The .IMPORT instruction in section 5.2 fetches labels from other modules, whereas the .EXPORT instruction makes labels from Start8FX.asm known for other modules, e.g. the monitor kernel.

Section 5.5 sets the Monitor Debugger Configuration Table which contains several definitions for monitor debugger initialization. It also creates an entry for CALLV#0 in the vector table. This is used similar to a software interrupt instruction to perform a vector call to a sub function. By default, a label _FGM_CALL which is start address of monitor kernel break point handler function is assigned here.

Section 5.6 sets the start address after reset depending on configuration used to the real start address of the user application or to the initialization function of the monitor kernel.

Please see the following tables for a short explanation on the labels used:

Table 5 . Imported labels in Start8FX.asm

_AutoBootCheck	This label is the start address of function AutoBootCheck() defined in Autoload.c. It is assigned to the label _FGMTBL_ABOOTADR. Please refer to chapter 3.2.5 for details.
_FGM_CALL	This label is the start address of the breakpoint handler function of the monitor kernel. This function is included in the library FGM.rel. The label is assigned to the CALLV#0 vector in the vector table.
_FGM_INIT	This label is the start address of the initialization function of the monitor kernel. This function is included in the file FGM.rel. The label is assigned to the reset vector in the vector table.

Table 6. Exported labels from Start8FX.asm -> Monitor Debugger Configuration Table

_FGMTBL_STARTADR	The start address of the user application is specified here. By default, this start address is the label "__start" which is set as reset vector when not using monitor debugger.
_FGMTBL_STACKTOP	The address for the stack area to be used by the monitor kernel has to be specified here. By default, label "STACK_TOP" is used here.
_FGMTBL_ABOOTADR	After startup of the monitor kernel, a function to check whether or not the application should start automatically or wait for connection with the Softune Workbench can be executed. Specify the start address of this function here. For details on this function, please refer to chapter 3.2.5. By default, the template contains a function AutoBootCheck() that is stated here. Specifying 0x0000 or 0xFFFF makes the auto-boot checker invalid.
_FGMTBL_PASSWORD	Here the user password mentioned before is set in the configuration table. By default, the value USER_PASSWORD defined in section 4.9 is assigned.

OBSOLETE

3.2.3 Vectors.c

The file Vector.c contains the settings for the interrupt vectors. It includes a function to set the default interrupt levels as well as the interrupt vector table definition.

To use the Softune Workbench Monitor Debugger, which uses the UART/SIO channel 0, the corresponding interrupt vector (IRQ4, 0xFFF2) has to be used. Therefore in MONDEB configuration the interrupt level of IRQ4 is set to 0 (highest level) and the interrupt service routine FGM_INT from the monitor kernel is entered as interrupt vector for this interrupt source.

Please take care that you cannot use UART/SIO channel 0 in the application when using monitor debugger. Take care that you do no change the interrupt level of this channel in your own program code.

```
void InitIrqLevels(void)
{
…
#ifdef MONITORDEBUGGER_CONFIGURATION    // if monitor debugger is used
    ILR1 = 0xFC;        //  IRQ4:  UART/SIO ch0
                        //                 --> enabled for monitor debugger
                        //  IRQ5:  8/16-bit timer ch0 (lower)
                        //  IRQ6:  8/16-bit timer ch0 (upper)
                        //  IRQ7:  LIN-UART (reception)
#else                               // if no monitor debugger is used
    ILR1 = 0xFF;        //  IRQ4:  UART/SIO ch0
                        //  IRQ5:  8/16-bit timer ch0 (lower)
                        //  IRQ6:  8/16-bit timer ch0 (upper)
                        //  IRQ7:  LIN-UART (reception)
#endif
…
}
```

```
/*------------------------------------------------------------------------
   Prototypes
------------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler (void);

#ifdef MONITORDEBUGGER_CONFIGURATION
    __interrupt void FGM_INT (void);
#endif
```

```
/*------------------------------------------------------------------------
   Vector definition
------------------------------------------------------------------------*/
…
#ifdef MONITORDEBUGGER_CONFIGURATION
    #pragma intvect FGM_INT             4    // IRQ4 for monitor debugger
                                             // usage:  UART/SIO ch0
#else
    #pragma intvect DefaultIRQHandler   4    // IRQ4 standalone ISR:
                                             // UART/SIO ch0
#endif
```

OBSOLETE

3.2.4 FGM.rel

The monitor kernel itself is included as relative format load module (.rel) in the template project. The source files of the monitor kernel are already compiled and linked. As the output was no absolute format load module (.abs), this file can be used as input file for the linker again.

The file FGM.rel includes the already mentioned functions for kernel initialization (FGM_INIT), the interrupt handler for a UART/SIO channel 0 interrupt (FGM_INT) and a breakpoint handler function (FGM_CALL) as well as all needed internal functions for running the monitor kernel.

If you are interested in the source code, please have a look at the assembly file fgm_main.asm which you can find in your Softune Workbench installation directory in subfolder [sample\896\MONDEB\FGM], e.g. C:\Softune\sample\896\MONDEB\FGM.

3.2.5 Autoload.c

The function AutoBootCheck() is called by the monitor kernel to select whether to start the application or to wait for communication with the Softune Workbench. This decision is done based on the return value from the called function. By returning 0 (0x0000), the kernel waits for communication. By returning any other value than 0, the application is started automatically.

The function to be called is defined in the Monitor Debugger Configuration Table (see Table 6 in chapter 3.2.2).

The function included in each microcontroller series template project is an empty function without any code which just returns 0x0000. So when linking the application in MONDEB configuration and flashing to MCU, the kernel will always wait for the host program to communicate.

```
#include "../mb95100.h"

int AutoBootCheck(void)
{
        return 0x0000;  // DEBUG
}
```

It is possible to enter own code inside this function to define conditions for autostart and debug mode. Just keep in mind that this function is executed immediately after microcontroller is released from a reset. It therefore must be coded to be executable with resources not initialized or with C-defined initial-valued variables not initialized yet.

The template project for the CONCERTO-Kit gives an example for such an own function implementation. It checks the logical level at IO Port PE0/INT10. On CONCERTO-Kit, a push button is connected to this pin. In default state when button is not pressed (open), the logical level is forced to low level due to a pull-down resistor. If the button is pressed (closed), the level is changed to high level due to a direct connection to Vcc.

If the button is not pressed after reset (default), the application waits for a communication with the Softune Workbench. If after reset the button is pressed, the user application is started.

```
#include "../mb95100.h"
int AutoBootCheck(void)
{
        DDRE_DE0 = 0;          // PE0=Input
        if (PDRE_PE0 == 0)     // if( PDRE_DE0 == 0)
                              // --> button 'INT10' not pressed
        {
                return 0x0000;  // DEBUG
        }
        else                    // if( PDRE_DE0 == 1)
                              // --> button 'INT10' pressed
        {
                return 0x0001;  // AutoBoot
        }
}
```

OBSOLETE

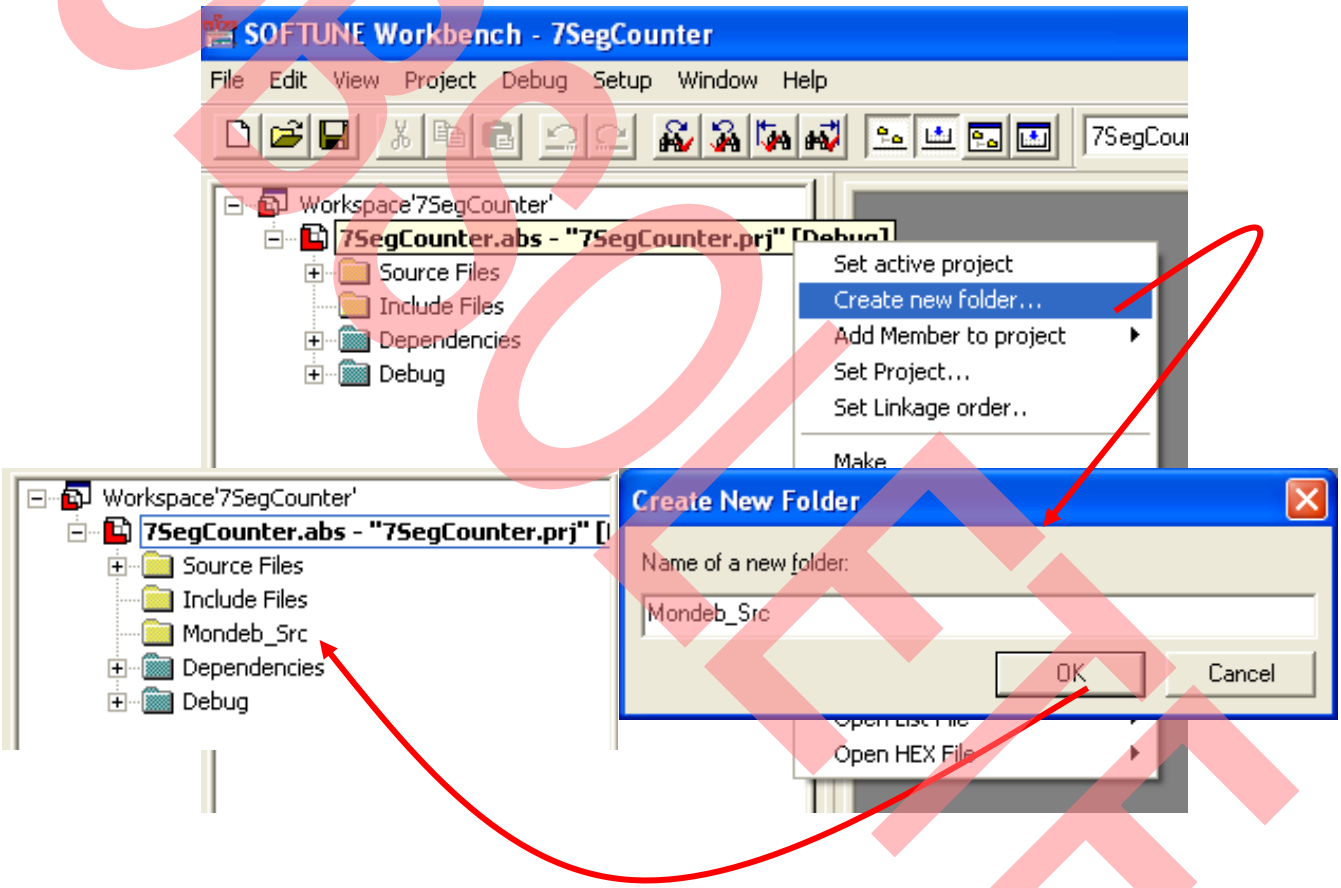
3.3 Integrating in existing project

To modify an existing application to work with the Softune Workbench Monitor Debugger, the points mentioned in chapter before have to be added to the project.

3.3.1 Integrating Monitor Kernel

Please copy the folder [src\Mondeb_src] including the files FGM.rel and autoload.c from one of the template projects to your application's source folder. Open your project workspace. Right click in project window on line '*your_application*.abs – *your_application*.prj' (*your_application* is the name of your project, e.g. 7SegCounter.asb – 7SegCounter.prj in example below). Select 'Create new folder…' and enter new folders name in popup window. After pressing enter button, new folder is added as subfolder of project.

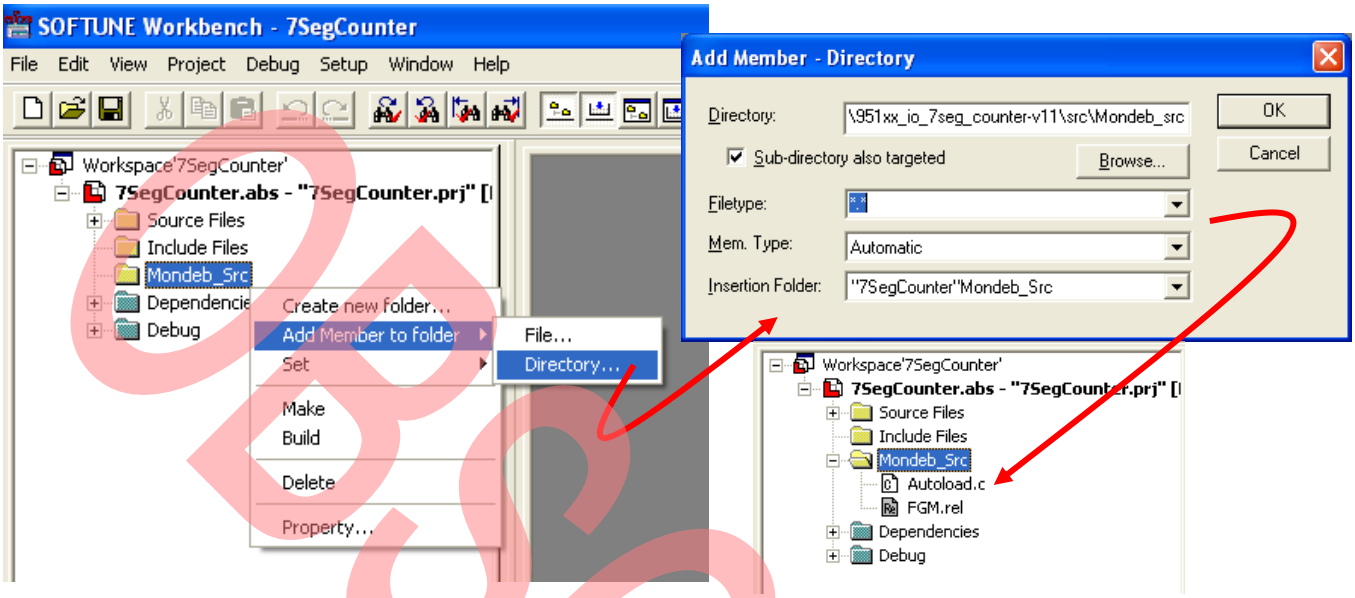
Figure 13 .Add new folder to project



Right click on the new created folder and select 'Add Member to project' > 'Directory…' Browse to the folder [Mondeb_Src] in your projects source folder in the upcoming dialogue box. Set file type [*.*] to include .rel and .c file from the folder. Please make sure that Insertion folder is set to ["*Your_application*"Mondeb_Src].

PROFT

Figure 14. Add monitor kernel to project



3.3.2 Configuration & Vector Table

Easiest way to adopt your source code for use with the Softune Workbench Monitor Debugger would be to use current version of Start8FX.asm and Vectors.c taken from template project. You only have to add your project-specific changes (e.g. clock settings, vector table) to these files. Please refer to chapter 3.2.1/Figure 12 for adding the macro definition "–D MONITORDEBUGGER_CONFIGURATION" to your project.

If it is not possible just to modify these two files, you have to take care manually to adopt relevant locations in your source code and to add FGM configuration table.

- Set CALLV#0 vector to label FGM_CALL
- Set UART/SIO channel 0 interrupt vector (IRQ4) to label FGM_INT
- Set UART/SIO channel 0 (IRQ4) interrupt priority to high (ILR1[0..1] = b'00)
- Set Reset Vector to label FGM_INIT
- Add FGM configuration table (refer to chapter 3.2.2/Table 6)
- Take care to have 4bytes on stack available for monitor debugger kernel
- Modify the AutoBootCheck() function according ton your needs

OBSOLETE

3.3.3 Watchdog Usage

If you are using the internal watchdog function of the 8FX family, please replace the watchdog timer start routine (first write to WDTC register after reset) with the monitor kernels function void FGM_WDTON(unsigned char SETVAL). The parameter SETVAL is the value to be written to WDTC register.

e.g. replace following line

```
// start watchdog timer based on output cycle of time-base timer (2^20/FCH)
WDTC = 0x45;
```

with the function call to the kernel function

```
// start watchdog timer based on output cycle of time-base timer (2^20/FCH)
FGM_WDTON(0x45);
```

Using this kernel function can prevent the watchdog timer from generating a reset even when control is passed to the monitor program.

OBSOLETE

4 Monitor Debugger

This chapter shows the steps to start a debug session and the debugging possibilities of the Softune Workbench Monitor Debugger.

4.1 Flash Application to Target

Before being able to start a debug session with the Softune Workbench Monitor Debugger, the source code including the monitor kernel has to be flashed once to the microcontroller using an external flash tool.

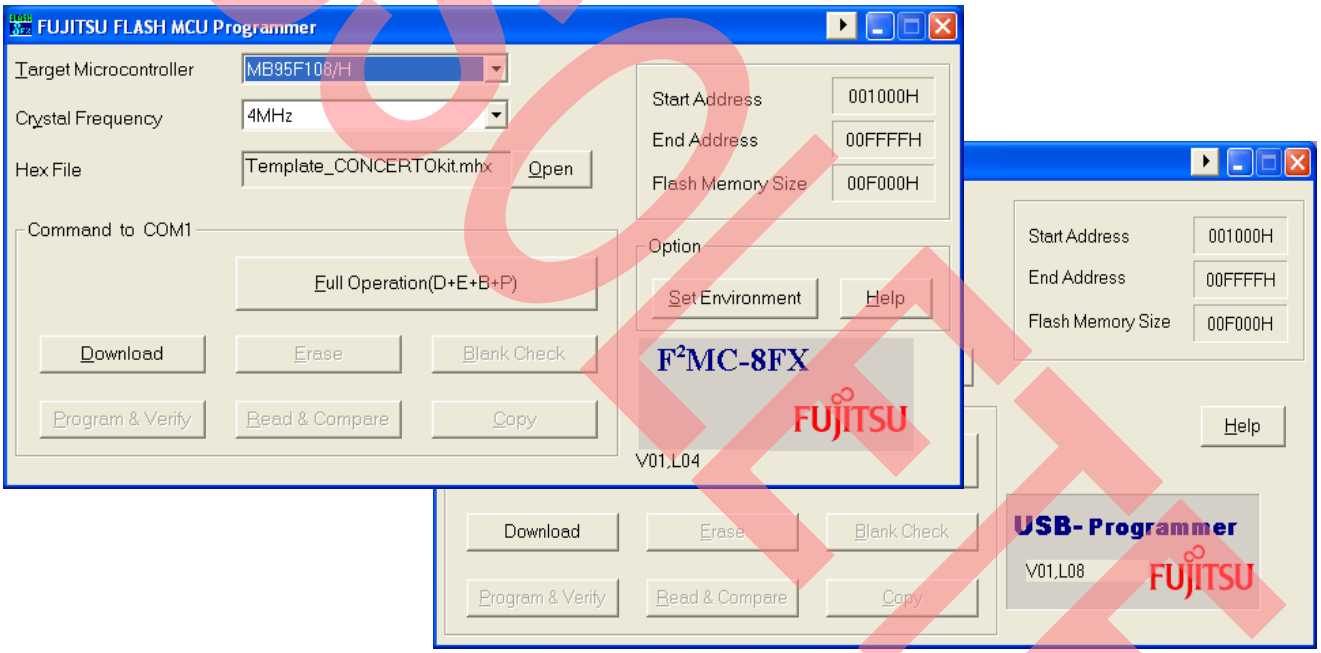
Cypress provides a free-of-charge serial asynchronous tool and a tool for serial synchronous flashing via BGM adapter MB2146-09A-E.

Please refer to separate application notes for details:

- mcu-an-395003-e-8fx_async_flash_programming
- mcu-an-300050-e-8fx_sync_flash_programming

As the serial synchronous interface requires the same connection of MB2146-09A-E to the microcontroller and to the PC, this would be the easiest way for programming.

Figure 15. Cypress Flash Programmer Software



If you are using a different kind of programming tool (e.g. Xeltec), this programmer can also be used for flashing the application to microcontroller the first time.

After successful programming of microcontroller and connection to monitor kernel, the kernel itself is able to flash updated or changed application to MCU without the need to use an external programming tool.

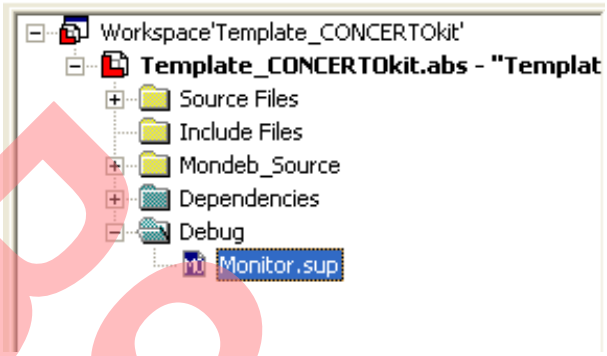
Please make sure to set microcontroller to run mode (pin MOD = GND) after programming!

OBSOLETE

4.2 Starting Debug Session

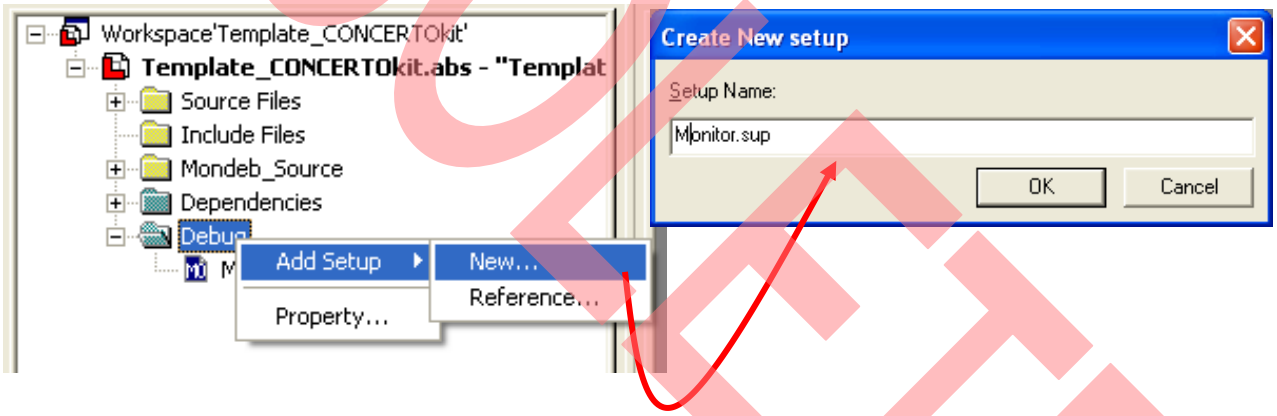
To start debug session via monitor debugger, please select MONDEB configuration in template project. In folder Debug in project window should exist the entry Monitor.sup.

Figure 16. Start Debug Session



If the entry monitor.sup is not included, you have to set up new. Right click on Debug > Add Setup > New… Enter new name in the upcoming dialogue box.

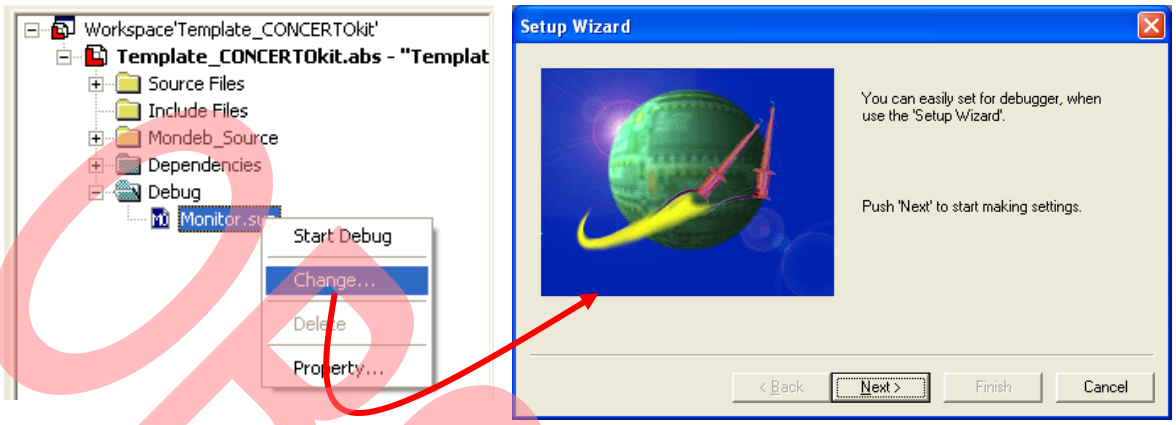
Figure 17. Add new debugger setting



Then the setup wizard for the debugger setting is opened. To modify an existing debugger setting, please right click on the setting and select ‘Change…’. Now also setup wizard is opened.

OBSOLETE

Figure 18. Modify debugger setting

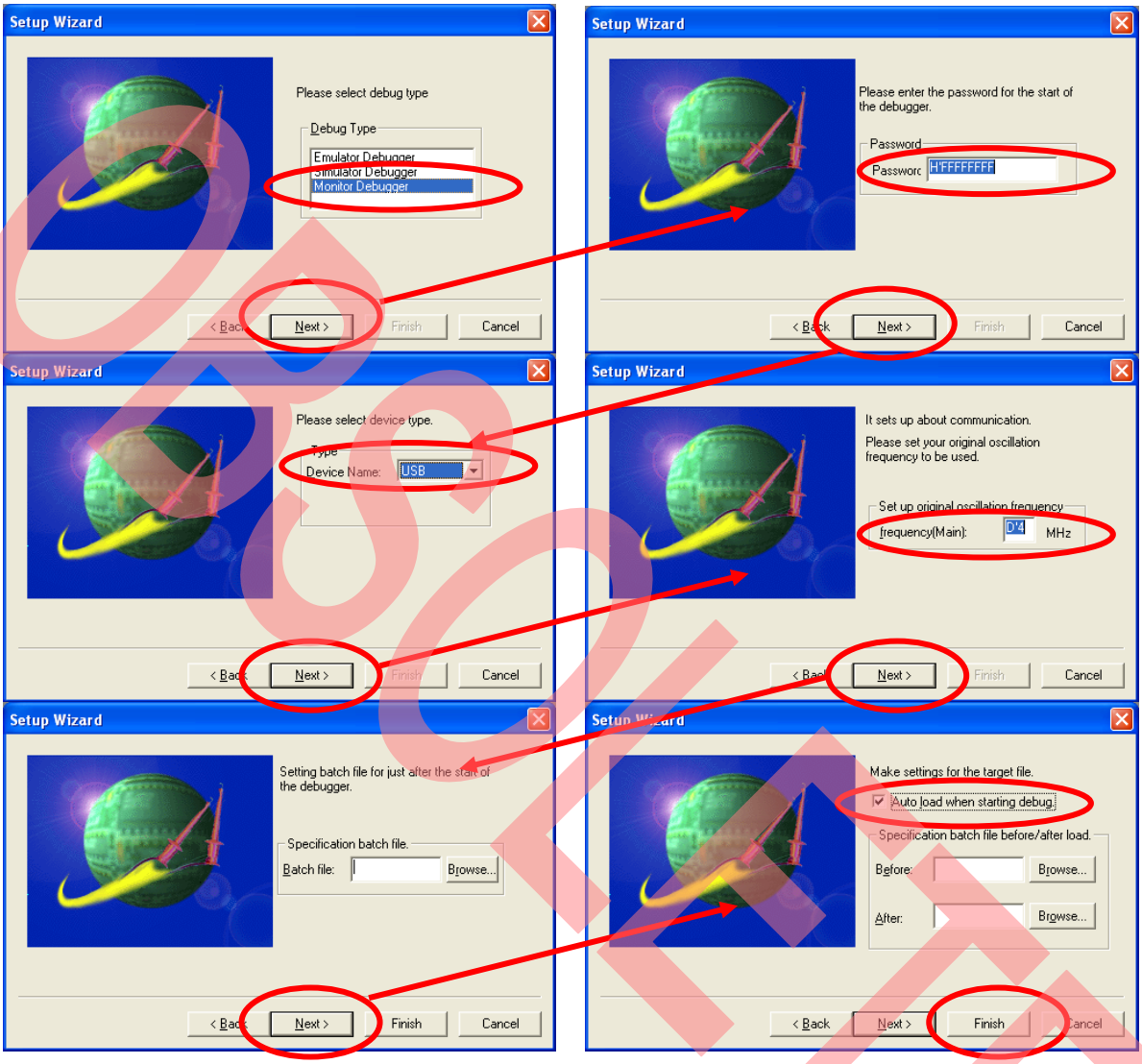


Please follow the steps below to adapt the correct settings:

- Click ‘Next’ to go to the next step
- Select debug type ‘Monitor Debugger’
- ‘Next’
- Enter the password defined in Start8FX.asm (see chapter 3.2.2).
- ‘Next’
- Set device name ‘USB’.
- ‘Next’
- Set frequency of the main oscillator
- ‘Next’
- You can specify a procedure file to be executed directly after starting debug session (default: none)
- ‘Next’
- Check ‘Auto load when starting debug’ (project is loaded when starting debug session), you can also set procedure file to be executed before/after project loading (default: none)
- ‘Finish’

OBSOLETE

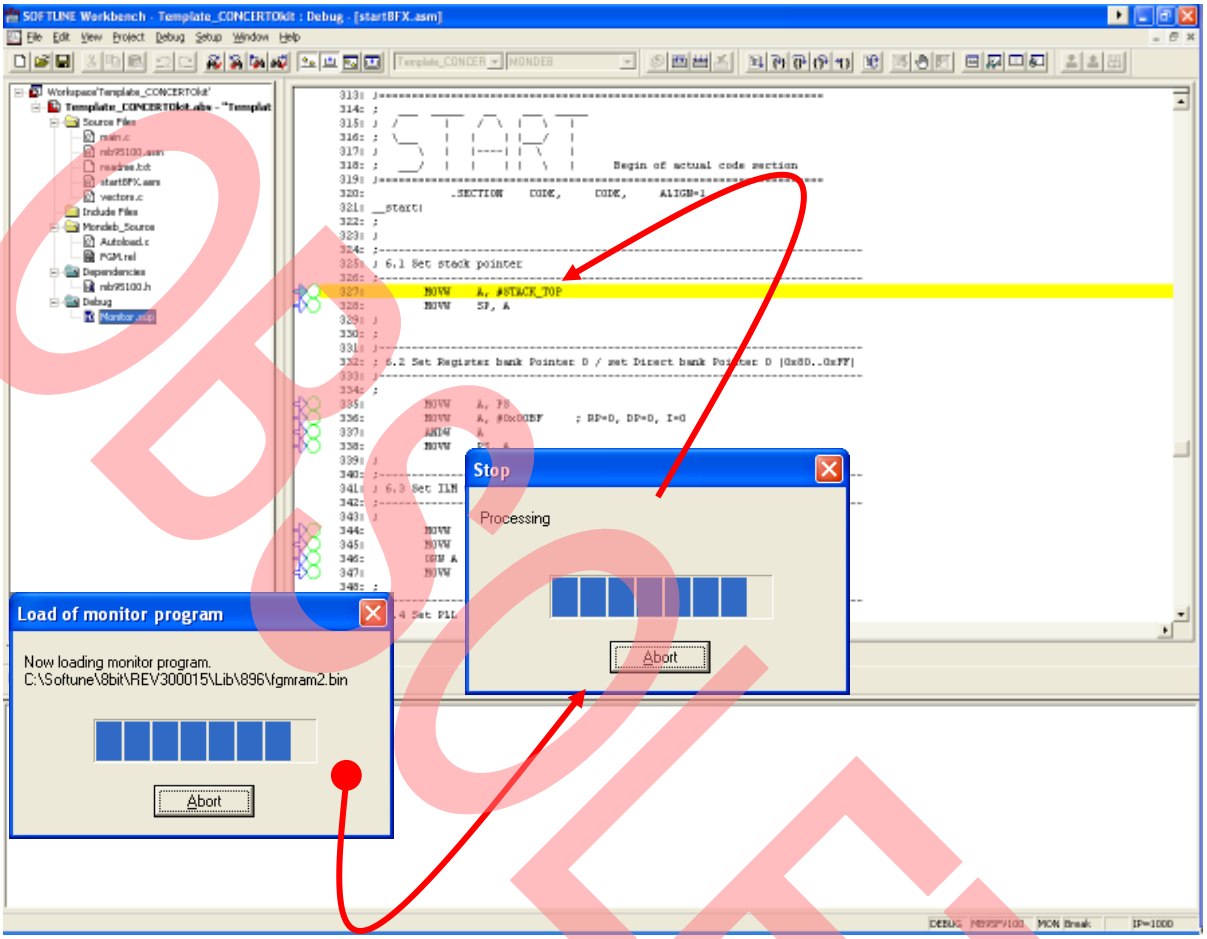
Figure 19. Setup Wizard for debugger settings



Double click on Monitor.sup in folder Debug of project window to start the debug session. The Softune Workbench downloads some monitor programs to the BGM adapter MB2146-09A-E and then loads the .abs project including the debug information on PC. After successful connection, the program counter is set to __start label in software.

PROFIT

Figure 20. Started debug session

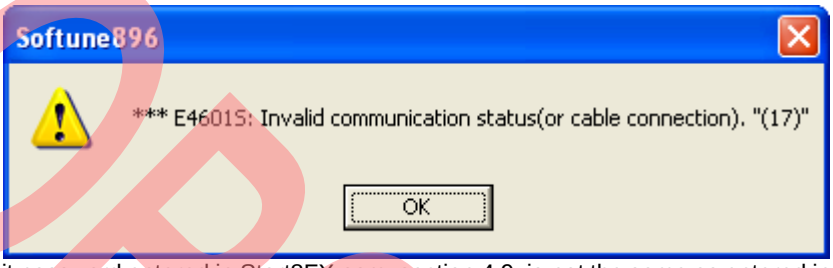


OBSOLETE

4.2.1 Error Messages

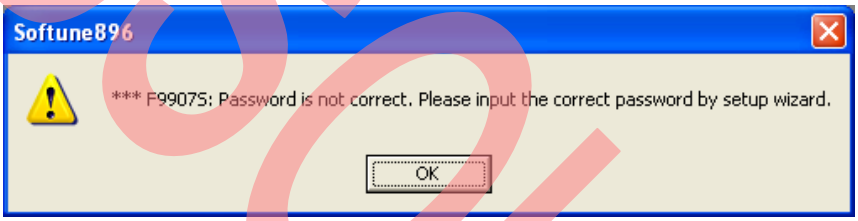
If you get following error message when starting debug message, please check correct USB connection. Make sure USB Programmer tool is not opened any more. If problem still occurs, reset the BGM adapter by shortly unplugging USB cable.

Figure 21. Error message invalid communication



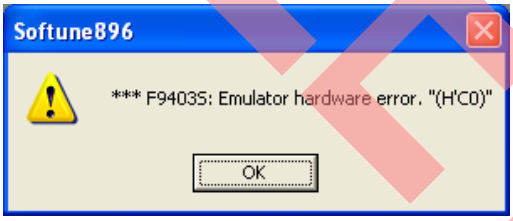
If the 32bit password entered in Start8FX.asm, section 4.9, is not the same as entered in the setup wizard for debugger setting Monitor.sup, the following error message occurs.

Figure 22. Error message wrong password



If following error message appears, please check the frequency setting for main oscillator in setup wizard for debugger setting Monitor.sup.

Figure 23. Error message hardware error



Following error message appears if user power is not supplied to target board or if BGM adapter is not connected to target hardware.

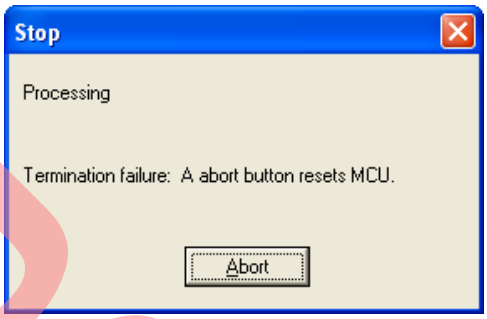
Figure 24. Error message missing user power



OBSOLETE

When issuing a break request via abort button in Softune Workbench, the following error message may shortly appear. This error message should disappear when break is executed.

Figure 25. Error message break request



OBSOLETE

4.3 Features of Monitor Debugger

The features of the monitor debugger are limited compared to emulator system. Please see list below for availability of functions:

Windows:

- Symbol
- Assembly
- Register
- Memory
- Local
- Watch
- Command

Source window:

- Source Code/Mixed View

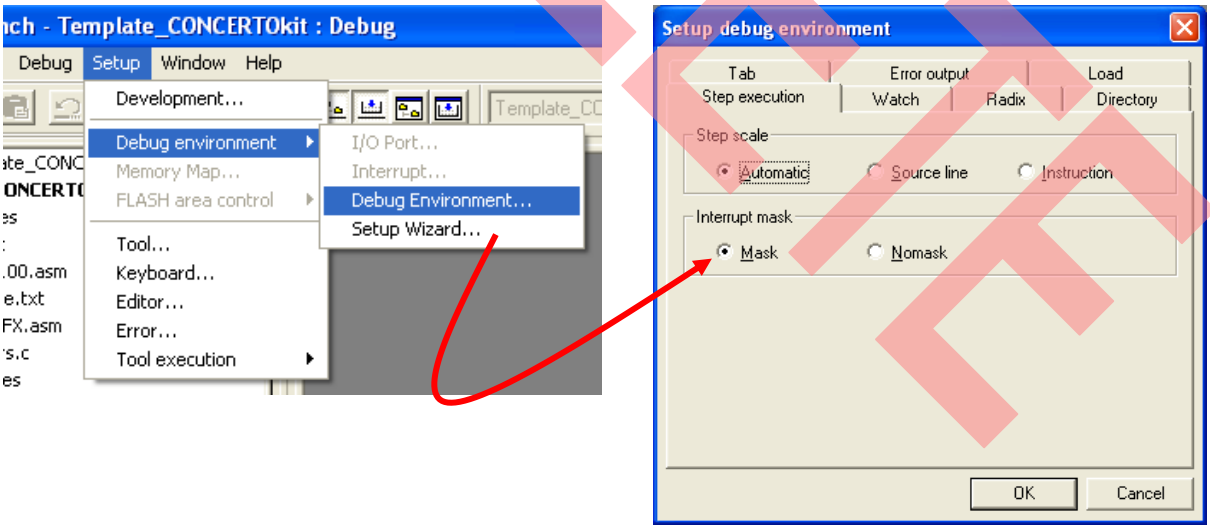
- Run
  - ☐ Go
  - ☐ Step In
  - ☐ Step Over
  - ☐ Step Out
- Reset of MCU
- Abort
- 2 Code Breakpoints:
- Call Stack View
- Function Call
- Vector Table View
-

Debug Functions:

Please be aware of following setting (Setup > Debug Environment > Debug Environment… > tab: Step Execution):

- Interrupt mask
  - ☐ **Mask**: During step execution, interrupt requests will be masked. This means, interrupt service routines are not entered.
  - ☐ **Nomask**: Interrupt requests are not masked. This means that during step execution interrupt routines will be entered.

Figure 26. Setting interrupt mask



OBSOLETE

5 Prohibitions and Restrictions

5.1 Prohibitions

- Do not manipulate any resource being used by monitor debugger!
  See chapter 3.1 for a list of used resources.
- Do not manipulate the clock/PLL registers PLLC or SYCC via the debugger!
  Monitor program reads these registers for adjusting communication speed. Setting via application program is no problem.
- Do not set a breakpoint within the monitor program!
  A breakpoint within a monitor function cannot be guaranteed and may disturb subsequent behaviour.
- Do not single-step through kernel function FGM_WDTON!
  This could cause the watchdog timer to get started before the monitor program is set up, involving the risk of generating a reset by the watchdog timer.

5.2 Restrictions

- The initial value of the stack pointer register SP is changed.
  Originally, the SP register is initialized to 0x0000 after reset. Since the monitor program uses the stack area, the application program is started with the register containing a value other than 0x0000.
- The time of start-up after reset is changed.
  Even when auto-booting is selected, the start-up time until execution of the user application in increased due to monitor program initialization routine and auto-boot checker function.
- A forced break by abort command is not available while UART/SIO interrupts are disabled.
  A forced break cannot be used to halt the application when the interrupt enable flag (I) is 0 or while an interrupt with higher priority than UART/SIO interrupt is being serviced.
- Dual clock products with sub clock input may require longer response times.
  The speed of communication with a microcontroller which can operate on sub clock may decrease to around 128 bps. If the microcontroller operates on sub clock, the communication speed remains at a speed of 128 to 4000 bps. Otherwise communication speed is increased.
- Code breakpoints are invalid during step-in execution.
  When Monitor Debugger performs step-in instruction for each machine instruction, it uses a wild register temporarily. In that period, no code breakpoint can be set, leaving code break points invalid. Be careful in particular if a break point has been set within an interrupt routine with the "interrupt mask" disabled during single-stepping. As the break point within the interrupt routine is invalid, no break takes place even when the break point service is executed.
- Use the "flash security feature" for read-out security of application and password.
  To use the password effectively, use the "flash security feature" of flash memory as well. Enabling the flash security feature protects flash memory from being read from or written to through external pins while allowing debugging with Monitor Debugger. (Only Chip Erase is possible.)

OBSOLETE

Document History

Document Title: AN205258 – F²MC - 8FX Family, MB951XX, SOFTUNE Workbench Monitor Debugger for 8FX

Document Number: 002-05258

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	WOFR	11/09/2007	V1.0 Markus Vogel First version
*A	5277559	WOFR	05/19/2016	Migrated Spansion Application Note “MCU-AN-300049-E-V10” to Cypress format.
*B	5612337	WOFR	01/31/2017	Spec obsoleted, no further updates planned.

OBSOLETE

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
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

OBSOLETE

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. 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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation,2007-2017. 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, 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.