Application Note

FUJITSU

Softune Workbench

Getting started

History

09th Mai 99	TKa	V1.0	started
28th June 00	TKa	V1.1	New format, some minor changes

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product 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.

2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH 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 Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

   NO LIABILITY FOR CONSEQUENTIAL DAMAGES

   To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

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

.

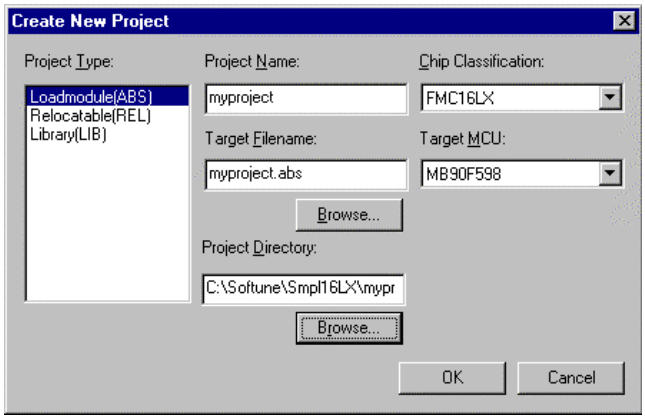
Softune Workbench getting started

The Softune Workbench is a software development environment to develop programs for the Fujitsu F2MC-8L, F2MC-16L/LX and FR families of microcontrollers. It is a combination of a development manager, emulator debugger, simulator and an integrated development environment for efficient development.

This is a short documentation to give an introduction how to start a project with the Softune Workbench. In the main it is referred to the F2MC-16L/LX Family but the settings for the F2MC-8L family are quite similar.

1. Getting started
1.1 Creating a new Project

To start a new project with the Softune Workbench just use the <File>, <New>, <Project File> menu. After that the following dialog box appears:

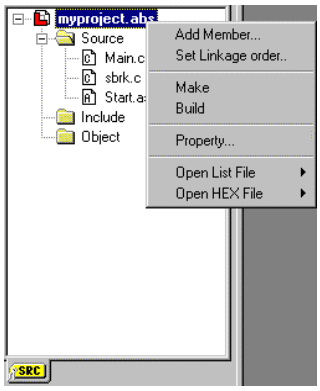


Here the project name must be entered first. This will create a new directory for this project. The location of this directory can be selected by the project directory browse button. The default target file name is myproject.abs, which is the loadmodule for the simulator or emulator. The default directory for this file is myproject\ABS.
In the next step the chip Family must be defined: 16LX, 16L (8L for the 8-Bit environment). In the last step the used target µC must be specified.

NOTE: Special template projects exists for each microcontroller series. These template projects include some basic settings for the MCU itself (which is done in a start.asm file), corresponding headerfiles and some basic project settings for the C-Compiler/Assembler/Linker and debugger. The easiest method to start a new project is to start with such a template project.
Of course the user has always to check the tool settings and settings in the start.asm file, which must be modified to the needs of the user application!

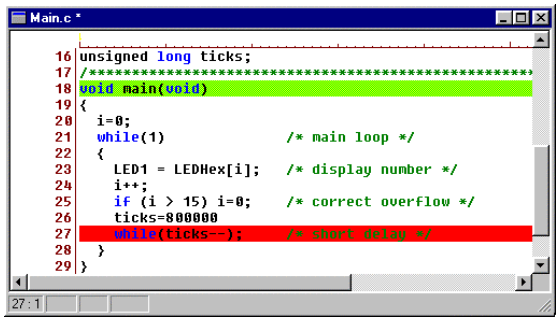
1.2 Edit the members List of the Project

When the setup of the new project is defined you will find all used modules of the project within the project window. Here the source files, include files and objects can be seen which are part of the current project. To add a new member to the project just click with the right mouse button on the target file name of the project (myproject.abs). A drop down menu appears to add the new member. Just browse to the corresponding location and add the module to the project.
Use the property tab to check the location of the .abs file and the location of the modules which are used in the project. With the "set linkage order" tab it is possible to set the order of the modules for the linker. It is recommended to have the start.asm file first in the linkage order. A standard start.asm file is offered by Fujitsu to do the basic initialisations and to define the segments which are used by the compiler. This start.asm file must be checked and modified corresponding to the needs of the application. This start.asm is just an example to simplify the first steps.



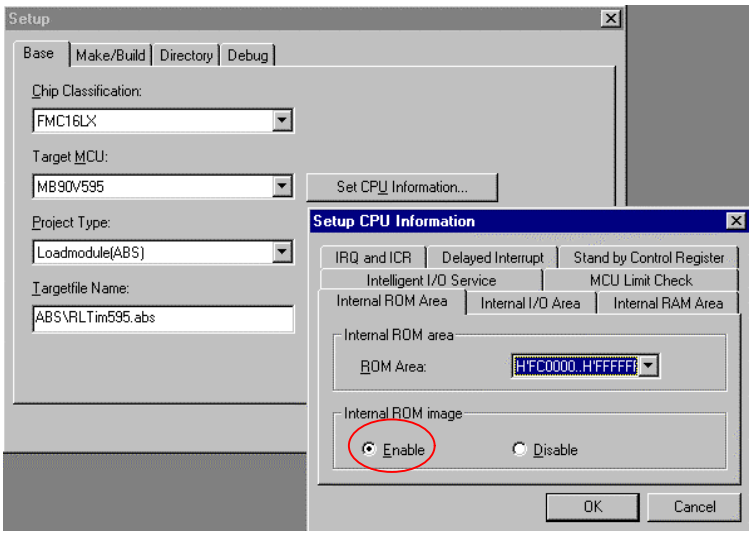
Note: The displayed order of the modules does not correspond to the linkage order. Therefore use always the linkage order tab to check the right order.
The project tab can also be used to have a look at the generated List Files (generated by the Assembler, C-Compiler, Linker) and the Hex output Files generated by the converter. After a Build of a project a look at the mapping file is highly recommended to check the segment locations, Stack, Vector table and so on.

The source files can be opened with the editor be a double click on the source name. To create a new source file use the <File>, <New>, <Text File> menu. The editor supports syntax highlighting, line numbers, different font selection and some more features which can be configured by the user. To customize the editor view just open the source file and use the right mouse button to drop down the context menu.



1.3 Check Project Setup

To check and modify the project setups use the <Project>, <Setup> menu. Here the microcontroller Family, the target CPU and the detailed CPU Information can be configured. Especially the <CPU Information>, <Internal ROM Area> tab should be checked. Here the internal ROM area is defined and the ROM mirror function is enabled to use the ROM mirror feature (mirrors address area 0xFF4000-0xFFFFFF to the area 0x004000 – 0x00FFFF). The mirror function is recommended for single chip mode with small or compact memory model (Defined in the C-Compiler Options).



1.4 Tool Option Setup

Before a “Build” some basic settings should be done at least for the Linker. Use the <Project>, <Setup Tool Option> menu to enter the options for the C-Compiler, Assembler Linker and the other development tools.

- **C Compiler**:
  - Category: Set Include path:
    - e.g. c:\Softune\lib\907\include\sample sets the default include path
    - Enable the “Output Debug Information” to be able to use the emulator debugger or simulator
    - Enable the “ Create assembly list file” in order to check the generated assembly code. If Compiler setting "Assembly List" file is on, also setting: - INF srcin is recommended. This adds C-source lines to the assembly list. The list files can be opened by selecting *.lst from List-files entry of the context menu in the member list.
  - Category: Target dependent

  Select the memory model which should be used (Small, Medium, Compact or Large Model). Corresponding to this setting 16Bit or 24Bit addresses will be generated for code and data adresses. Please look at the C-Compiler Manual for more detailed information on that topic. Small memory model should be idle for first getting started. Medium model should be used for bigger projects (code size > 64Kbyte).

  Now define whether Constants should be used in RAM or not. If Constant in RAM is selected all “variables” which are declared by const in the C-source file, will be accessed in RAM. For that purpose a segment CINIT is generated which is located by default in the RAM area. To have the correct values available in the CINIT segment, it is necessary to copy the initial values from ROM to the CINIT segment which is done in the start.asm file. To use the copy routines it is also necessary to set RAMCONST in the start.asm file! Otherwise no copy of the initial values to the RAM area will be done!

If ROMCONST is selected in the start.asm file, do not enable the compiler option Const in RAM! Otherwise the constants are located into the CINIT segment. But this CINIT is not initialized because the copy routines are not executed if ROMCONST is selected. If compiler is set to ROMCONST and the startup file (start.asm) is set to RAMCONST, the startup file will only generate an empty section. The code, which copies from CONST to CINIT will not have any effect then. It is highly recommended to set the compiler to ROMCONST for single-chip mode or internal ROM+ext bus.

- **Assembler**:
  - Set Include path: e.g. c:\Softune\lib\907\include\sample
  - Enable the "Output Debug Information"

Note: With the above mentioned include path, it is referred to the default header files supplied by Fujitsu Limited. Fujitsu Electronic Devices Europe (Germany) also offers some headerfiles which are compatible to the previous Softune Version V01. Therefore have a look onto the latest CD-ROM or contact Fujitsu for any updates.

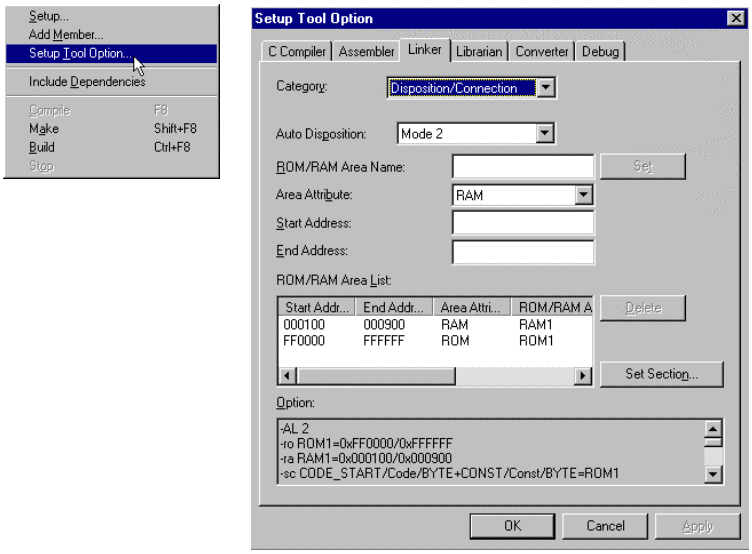
- **Converter**

  This tool generates a file with binary object code only. This is used for programming Flash or datafor mask ROM release. The emulator debugger does not use this file and reads the binary object from the .abs file.
  - The Converter itself must be enabled in the <Project>, <Setup>, <Make/Build> menu. Enable "Absolute Module Converter is started". Which converter should be used is defined in the <Setup>, <Tool Option> menu.
  - Set "Output Data Format" to "Intel" or "Motorola". For the 16-Bit µC "Motorola" format is recommended to be able to use the serial Flash Programming utility. Otherwise program download will not work. Default setting is Motorola S-Record. For the 8-Bit µC "Intel" format is recommended if the 8-Bit starterkit is used to download and test the software.
  - Enable the "Output Debug Information"

- Linker Settings

One of the basic settings are the linker settings to define the RAM and ROM areas with the adequate location settings for the used segments. First use the Category "Disposition Connection" and select Auto Disposition "Mode 2". In this mode the linker will place the segments corresponding to the attribute type into the RAM or ROM area automatically. Now the RAM and ROM areas must be defined.
For example:

- **Linker Settings**
  - Category: Disposition/Connection
  - Auto Disposition: **Mode2**



- ROM/RAM Area Name:
  - RAM1
  - Area Attribute: RAM
  - Start Address: 0x100
  - End Address: 0x900

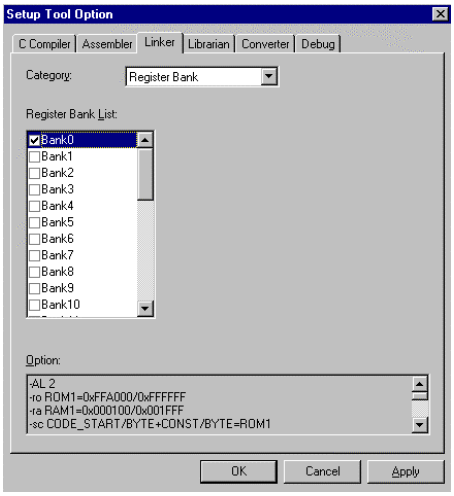
  - ROM1
  - Area Atrribute: ROM
  - Start Address: H'FF0000
  - End Address: H'FFFFFF

  - Set Section
    - RAM/ROM Area Name: ROM1
      - Section Name: START_CODE
      - Contents type: CODE
      - Alignment: Byte

    - RAM/ROM Area Name: Specify in Address
      - Section Name: CONST
      - Adress: FF4000
      - Contents type: CONST
      - Alignment: Byte

This linker settings are used to locate the START_CODE segment – which is defined in the start.asm file – to the address 0xFF0000. The other code is linked automatically beyond this location into the ROM area. Of course it is possible to define several sections independantly.
The special Linker address setting for the CONST segment is necessary to support the ROM mirror function. This ROM mirror function – if enabled in the start.asm file – mirrors the address area 0xFF4000 – 0xFFFFFF to the area 0x004000 – 0x00FFFF. This allows to access the upper ROM area, even if the small memory model is used. Therefore set CONST to any address above 0xFF4000.

- Category Register Banks
  - At least register bank 0 has to be selected in this setting if the C-Compiler is used. This reserves the memory area for register bank0 which is used by default from the C-Compiler.



Special Linker Feature

The Softune Workbench offers a special Linker feature which can be used for versatile applications. This feature is here explained to be used for the INIT and DIRINIT segments as an example.

Note: It should be mentioned that the default start.asm does not use this special feature for the above mentioned segments, because the far addressed data need to be collected for all modules in a special section (DTRANS, DCLEAR). This feature can be very helpful for other applications, e.g. to store the compiled code in ROM which is executed in RAM later on.

Special Linker setting:

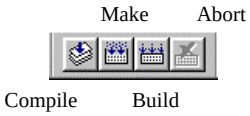
- RAM/ROM Area Name: ROM1
  - Section Name: @INIT
  - Section Name: @DIRINIT
    If in the program a variable is used with an initial value, it is often necessary to rewrite the variable during code execution. Therefore the variable must be located in the RAM area. The initial value of course has to be located in the ROM area. These initial values are linked to the INIT and DIRINT segments.
    So after startup the initial values has to be transferred from ROM to RAM. This can be done by the @INIT and @DIRINIT definition. The INIT and the DIRINIT segment are located in RAM, while the @INIT and @DIRINIT are located in the ROM area. With this definitions the symbols _RAM_INIT (transfer destination address) and _ROM_INIT (transfer source address) are generated by the linker. So these symbols can be used for a macro to copy the segment areas from ROM to RAM.

Save Settings

NOTE: All software settings are saved automatically when the Softune Workbench is closed or by the <File>, <Save>, <Project File>.
Note: The settings are not saved by the <File>, <Save All> command.

1.5 Build a Project

After this basic settings a Compile, Make or Build command can be executed. Therefore the special buttons can be used:



If there are error you will get warning or error messages in the output window corresponding to the selected warning level (Tool option settings).

To jump to the source code line which invoked this error message just double click the error/warning message or use F1 to get the adequate help information for this message.

1.6 Headerfiles

Fujitsu provides for each microcontroller series an example for a corresponding headerfile. After the default installation you will find Headerfiles in c:\Softune\lib\907\include\sample. These headerfiles have a complete different structure and I/O Register declaration as the former headerfiles supplied for Softune version 01.
To make the Software transfer from Version 01 to Version 03 as smooth as possible Fujitsu Electronic Devices Europe supplies new headerfiles which are compatible to the previous version. You will find these headerfiles on the CD-ROM. For any updates please contact Fujitsu or your local Distribution partner. These headerfiles consist of two files, a .asm file and a .h file, e.g. mb90595.asm, mb90595.h. In the .asm file the I/O register definitions are done, in the .h file the declarations are done. To use these headerfiles just include the .h file by e.g. #include “mb90595.h” and add the .asm file to your project.

1.7 Fine Tuning and Error Checking

After Linking it is always worthwhile to have a look at the linkage map and the output listfiles which were generated. In the Linker Mapping file (.mp1) can be checked whether any segment are overlapping. The location of the segments itself can be checked (e.g. Stack area) and the I/O register addresses can be checked. Right at the beginning also the linkage order can be seen. The other generated list files (if enabled in the Tool options) contains further detailed information which are very useful in case of debugging and error search.
The following Table shows the segment listing, which is an extract from the Linker mapping file.

```
FFMC-16 Family Softune Linker Mapping List

S_Addr. -E_Addr.   Size      Section Type Al     Sec.
00000000-000000BF  000000C0  IO    N RW-- 00 ABS  IOBASE
00000100-00000103  00000004  DATA  P RW-- 02 REL  DATA
00000100-........  00000000  DIR   P RW-- 02 REL  DIRDATA
00000100-........  00000000  DIR   P RW-- 02 REL  DIRINIT
00000104-0000010B  00000008  DATA  P RW-- 02 REL  INIT
0000010C-........  00000000  DATA  P RW-- 02 REL  CINIT
0000010C-........  00000000  DATA  P RW-- 02 REL  LIBDATA
0000010C-........  00000000  DATA  P RW-- 02 REL  LIBINIT
0000010C-0000040B  00000300  STACK P RW-- 02 REL  SSTACK
0000040C-0000040D  00000002  STACK P RW-- 02 REL  USTACK
00001900-00001FFF  00000700  DATA  N RW-- 00 ABS  IOXTND
00FFA000-00FFA103  00000104  CODE  P R-XI 01 REL  CODE_START
00FFA104-00FFA104  00000001  CONST P R--I 02 REL  CONST
00FFA105-00FFA185  00000081  CODE  P R-XI 01 REL  CODE
00FFA186-00FFA18D  00000008  CONST P R--I 02 REL  DCONST
00FFA18E-........  00000000  DIRC  P R--I 02 REL  DIRCONST
00FFA18E-........  00000000  CONST P R--I 02 REL  LIBDCONST
00FFA18E-........  00000000  CONST P R--I 02 REL  DCLEAR
00FFA18E-........  00000000  CONST P R--I 02 REL  DTRANS
00FFFF6C-00FFFFC3  00000058  DATA  N RW-- 00 ABS  INTVECT
00FFFFDC-00FFFFDF  00000004  CONST N R--I 00 ABS  RESVECT
```

1.8 Download Program to old Starterkit 16

To download the program to the old Starterkit 16, the SKWizard or the HexloadW utility can be used. Therefore the SKWizard can be added to the <Setup>, <Tool execution> menu. First the <Setup>, <Tools> menu must be started to add the SKWizard to the Utility list. Use the following options for the SKWizard: 1 –sk16 –C %D%X.mhx.
Alternately the HexloadW can be used with the option: 1 –sk16 -c %D%X.mhx

NOTE:
With the new Softune Workbench Symbolic debugging is not supported anymore. Furthermore the QSD source level debugger does not work if the code was created with the Softune Workbench. This is because HexloadW and the QSD debugger cannot use the new output file format of the Softune Workbench.

1.9 Converting a Project from Softune Version 01

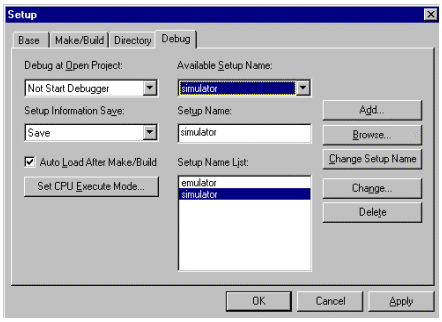
To convert a project from Softune version 01 to version 03 a special converter is offered. This conversion is necessary because some macro assembler instructions has been changed (e.g. definition of sections, reserving Bytes). Unfortunately this converter has some restrictions so that some modifications has to be done manually. For a detailed description of this converter please have a look at the manual. Also some documents are delivered describing the differences between the old and the new versions of C-Compiler, Assembler, Linker and so on.

1.10 Debugging

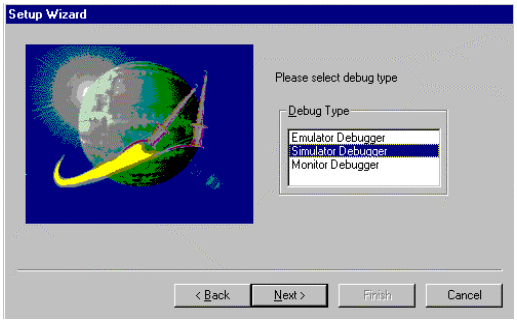
To debug the generated source code the simulator or emulator debugger can be used. Therefore the simulator and the emulator setup must be done first. Within the Softune Workbench it is possible to save different setups. The important news is that the debugger is now fully integrated into the Workbench environment.

1.10.1 Setup the Simulator/Emulator Debugger

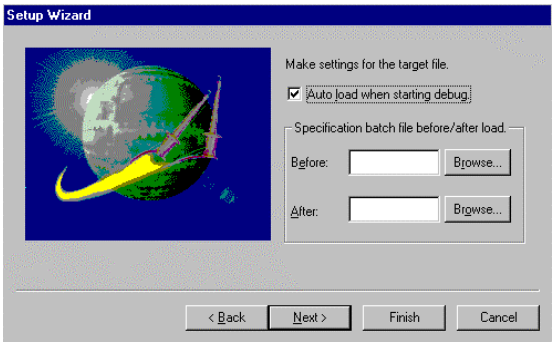
Use the <Project>, <Setup>, <Debug> tab to enter the setup configuration. After that Setup Dialog box is opened to enter the settings. Select “Not Start Debugger” or “Start Debugger” to start the debugger via the <Debug>, <Start Debug menu> or automatically after the project was built without any errors. The “Auto Load” option should be selected to ensure that the latest file is always loaded when the debugger is started.



In the next step enter the setup name, e.g. simulator. After that push the Add button. After that the setup Wizard is called to do the settings. Select for the debug type “Simulator Debugger” for a simulator setup and “Emulator debugger” for an emulator setting.



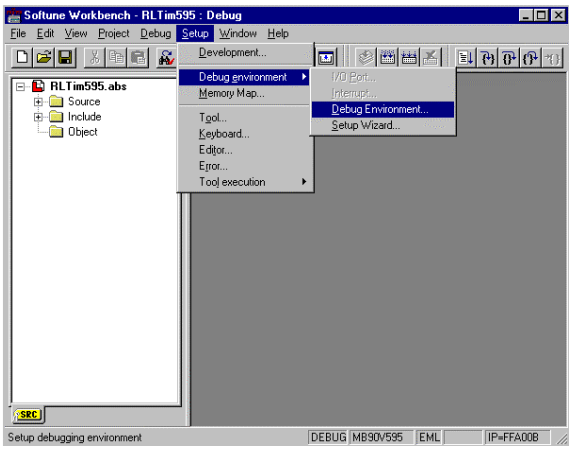
The Next button will lead to the next dialog box. Now select Auto load when starting debug if the debug file should be loaded automatically whenever the debugger is started. Furthermore a batch file can be entered which should be executed before or after the download. Such a batchfile could be e.g. a procedure file for the debugger.

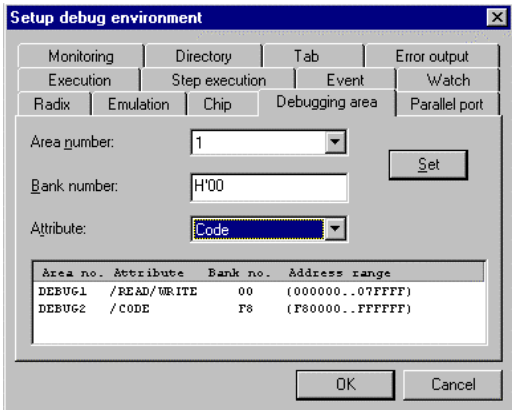


After pushing the Next button it is possible to select the items which should be restored after the debugger is started. By default all items are selected. Just use the Finish button to close the setup.

1.10.2 Start Emulator /Simulator Debugger

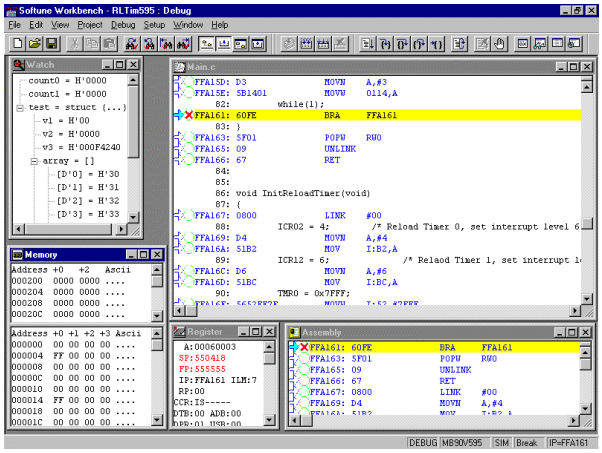
When the debugger is started first the debug area must be defined. There exits two 512Kbyte debug areas which can be defined by the user. Attributes like read, write, read/write and code can be defined for these areas.





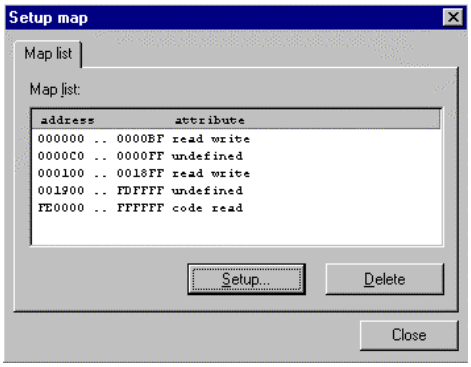
As a standard startup, for the debug area 1 the address area 0x000000 – 0x07FFFF (page 0) is used for data read/write access. For debug area 2 the address area 0xF80000 – 0xFFFFFF (page FF) is defined for code access.

After Starting the debugger the source window can be opened. Use the right mouse button (open context menu) to set Breakpoints, Watch Windows and so on. A special feature offers the memory window. First of all two different memory areas can be displayed in this window and each of these parts can be configured for different display modes (Binary, Byte, Word, LWord). The following screenshot gives a small impression of the look and feel.



1.10.3 Special Simulator Settings

Because the simulator is a core simulator only, it is not possible to simulate the resources as e.g. the UART. But it is possible to simulate interrupts and IO ports. I/O ports can be simulated using a file to input or output data whenever an access is made to a special I/O address. To simulate Interrupts or I/O port functionality use the <Setup>, <Debug Environment> menu, after the debugger is started.
After starting the debugger it is very important to setup the memory map for the debugger. Here the address areas must be defined which are accesses by the code. This means the I/O area, RAM and ROM area must be defined. The following screenshot gives an example setup configuration for the simulator.



Use the <Setup>, <Memory Map> menu to configure the address map. Different attributes Read, Write, Code can be defined for the areas.

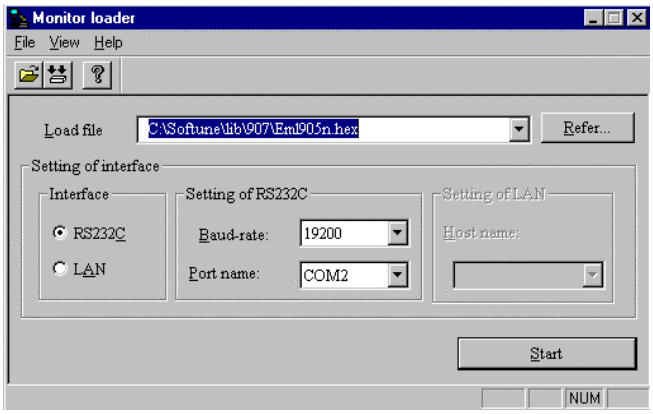
Note:

If the ROM Mirror Function is used with the simulator, it is necessary to copy manually the address area 0xFF4000 – 0xFFFFFF to 0x004000 – 0x00FFFF. Therefore open the memory window and use the right mouse button to select the copy command.

1.10.4 Special Emulator Settings

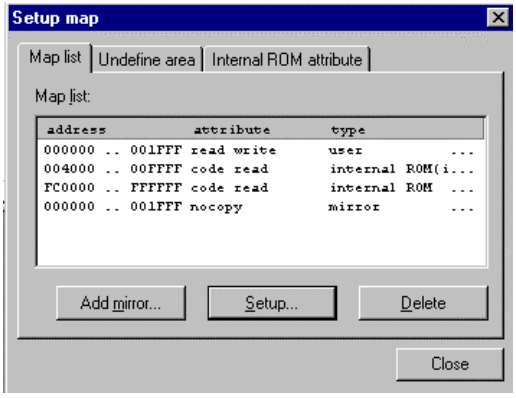
Monitor download

For the Hardware installation of the Emulator Debugger please look at the short description for the Emulator Hardware setup or read the manuals for the MB2141A main Unit and MB2145-507 emulation pod.
If the emulator is started for the first time or if the microcontroller Family (16L <-> 16LX) for the project has been changed it is necessary to download a monitor program first. For that reason start the Monitor Loader from the windows desktop (<Start>, <Programs>, <FFMC 16 Family Softune Workbench>, <Monitor Loader>).



Select the corresponding monitor program (eml905n.hex for 16LX family and MB2145-507 emulation pod, eml905.hex for 16LX family and MB2145-506 emulation pod, eml906n.hex for 16L family and MB2145-507 emulation pod, eml906.hex for 16L family and MB2145-506 emulation pod). The monitor files itself are located in the c:\Softune\lib\907\ directory. Also the communication interface (RS232 or LAN), the Baud-rate and the communication port must be selected. After this push the Start button to start the download.

Memory map setting
The address map for the emulator system is configured in the <Setup>, <Memory Map> menu.



Use the Setup button to enter the memory areas. For each areas different attributes like read, write, code, user memory, emulation memory can be specified. User memory is memory which is located in hardware on the target system. Emulation memory is memory which is located in the emulator system
A special feature is the memory mirror. A mirror area is always necessary for user memory to allow a memory read on the fly, which means to display the memory area while the program is running.
To add a mirror area just highlight the corresponding address area within the dialog box and click on the Add Mirror button. Therefore a mirror is necessary because otherwise the emulator has to perform additional read cycles to this memory area during the normal program execution. This would disturb the normal program run sequence and execution time. Therefore a mirror within the emulation memory is used, so on every CPU access to the memory this data is updated in the memory mirror.

1.11 Windows setup for Floating Point operations

If floating point operations are used in the application, the Windows setup “Regional Settings” must be checked! In the “Regional Setting” for Windows (which can be found in the “control panel”), the number definition for “Decimal Symbol” must be set to “.”. If “,” is used, float operations will not be performed correctly! This problem is solved with the Softune Workbench Package V30L24.

2.0 Special Notes for the F2MC-8L environment

After Installation of the Workbench a demo program (Led8lirq.prj) can be found for the Starterkit 8. This demo program is an example which explains how to implement interrupts and how to include the standard header files for the corresponding microcontroller.

Note:
If a Softune V01 project is opened with the Softune Workbench an automatic converter can be started. Unfortunately this converter does not convert the assembler- and header files. So the conversion of the Softune V01 assembler- and header files must be done manually. This is necessary because the syntax of the assembler pseudo instruction has slightly been changed, e.g. definition of segments and label instructions.

Creating a new Project

The following steps describe how to create a new project.

- File
  - New
  - Project File
    - Select Project Type: e.g. Loadmodul (ABS)
    - Enter Project Name: e.g name1
    - Select target MCU: e.g. 89T637A
      Note: This setting has an influence to the emulator/simulator settings. Address settings for the Linker must be done separately.

- To create a new source file use <File>, <New>, Text file
- In the example projects some init.asm files can be found. Within these init files some segment definitions and basic initializations are done. Corresponding to the application some changes has to be done, e.g. Stack size definition.
- In the example projects a file mode_ext.asm or mode_sng.asm can be found. This files give an example how to specify the Reset vector and the Mode Byte to use an external bus interface or to use the single chip operating mode.
- Additionally some settings has to be done, which are described in the next section.

Note: Special template projects exists for each microcontroller series. These template projects include some basic settings for the MCU itself (which is done in a start.asm file), corresponding headerfiles and some basic project settings for the C-Compiler/Assembler/Linker and debugger. The easiest method to start a new project is to start with such a template project.
Of course the user has always to check the tool settings and settings in the start.asm file, which must be modified to the needs of the user application!

Setups

If a new project has been created, **check the following setups** in the <Project>, <Setup Tool Option> menu. The following example shows some settings.

- **C Compiler**:
  - Set Include path: e.g. c:\Softune\lib\896\include\sample
  - Enable the “Output Debug Information”
  - Enable the “ Create assembly list file”

- **Assembler**:
  - Set Include path: e.g. c:\Softune\lib\896\include\sample
  - Enable the “Output Debug Information”

- **Converter**
  - Enable the “Output Debug Information”
  - Set “Output Data Format” for 8Bit Starterkit applications to “Intel”. Otherwise program download will not work. Default setting is Motorola S-Record
  - In the <Project>, <Setup>, <Make/Build> menu select: absolute module converter is started
- **Linker**
  - Category: Disposition/Connection
  - Auto Disposition: **Mode2**
  - ROM/RAM Area Name:
    - RAM1
    - Area Attribute: RAM
    - Start Address: 0x80
    - End Address: 0x480
    - ROM1
    - Area Atrribute: ROM
    - Start Address: H’2000
    - End Address: H’FFFF

    - Set Section
      - RAM/ROM Area Name: Specify in Address
        - Section Name: CODE
        - Start Address: H’2000
        - Contents type: Code
        - Section Name: CONST
        - Contents Type: Const

      - RAM/ROM Area Name: ROM1
        - Section Name: @INIT
        - Section Name: @DIRINIT

        If in the program a variable is used with an initial value, it is often necessary to rewrite the variable. Therefore the variable must be located in the RAM area. The initial value of course has to be located in the ROM area. These initial values are linked to the INIT and DIRINT segments.
        So after startup the initial values has to be transferred from ROM to RAM. This can be done by the @INIT and @DIRINIT definition. The INIT and the DIRINIT segment are located in RAM, while the @INIT and @DIRINIT are located in the ROM area. With this definitions the symbols _RAM_INIT (transfer destination address) and _ROM_INIT (transfer source address) are generated by the linker. So these symbols can be used for a macro to copy the segment areas from ROM to RAM which is done in the init.asm.

Download Program to Starterkit 8

To download the program to the Starterkit 8 the SKWizard can be used. Therefore the SKWizard can be added to the <Setup>, <Tool execution> menu. First the <Setup>, <Tools> menu must be started to add the SKWizard to the Utility list. Use the following options for the SKWizard: 1 –sk8 –R –C %D%X.ihx.
Alternately the HexloadW can be used with the option: 1 -i9600 -w -c %D%X.ihx

Note: Symbolic debugging is not supported at the moment. This is because Hexloadw3a cannot generate the symbol list from the Softune Workbench input file.