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

How to Check the Ordering Part Number

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

For More Information

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

About Cypress

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

AN205555

F²MC-16FX Family, Emulator System MB2198

This application note describes how to use the Emulator MB2198-01 together with the Softune Workbench V30L33R08 or higher using an evaluation MCU with its Starter Kit.

Contents

1 Introduction

This document describes the first steps how to use the Emulator MB2198-01 together with the Softune Workbench V30L33R08 or higher using an evaluation MCU with its Starter Kit.

For the hardware set up please refer to the Emulator System MB2198-01 Installation Guide (002-05547), and the MB2198-01 and MB2198-500 Hardware Manuals.

To use this document, please install the corresponding MCU sample folders of your Cypress Micros Documentation & Software DVD to your PC. Also, please read the notes on the DVD inlet.

Note: This document describes exemplarily the proceeding with a MB96V300 MCU and a FLASH-CAN-100P-340 Board.

2 Getting Started

Preparing a New Project

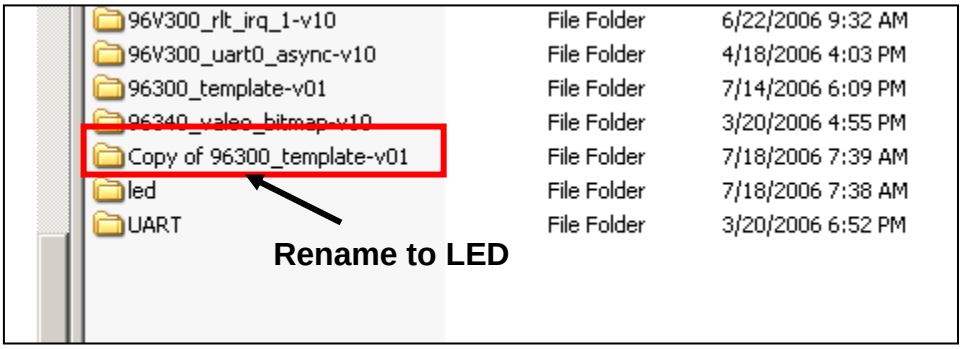
2.1 Copying the “template” directory

If you want to create a new project it is strongly recommended to use the “template project” as a base of operations.

The “template project” includes all basic settings for a quick and safe start. It includes the header and includes files of the latest released version.

To start, copy the “template” directory with all its sub folders.

For example, we want to create a project, which toggles the LEDs on the FLASH-CAN-100P-340 Starter Kit. For that, rename the copy of the “template” folder to “LED”.



2.2 Adjusting the project files

First click into the new “LED” folder and rename the following files:

Template.prj →Led.prj

Template.wsp →Led.wsp

Now open the file “Led.prj” with your favourite text editor.

You will find an entry like this:

```
[DirInfo]
PRJ=C:\Cypress\Sample\96340\template\
```

In the line “PRJ=” please edit the file path to your correct path. Of course, the “template” has to be renamed to “led”.

Save and close the file. Attention: If you use MS Word, please be sure to save the file as a normal text file, *not* as a word document!

Now open the file “Led.wsp” also with a text editor.

Do the following changes:

```
[PrjFile]
Count=1
FILE-0=Template.prj              →      FILE-0=Led.prj
ActivePrj=Template.prj           →      ActivePrj=Led.prj

[SubPrj-Template.prj]            →      [SubPrj-Led.prj]
```

Like in the Prj-File, you have to set your correct project folder path in the “WSP=” line:

```
[DirInfo]
WSP=C:\Cypress\Sample\96340\template\
```

Now save and close the file also as a normal text file.

The new project environment is now adjusted.

2.3 Adjusting Linker Settings

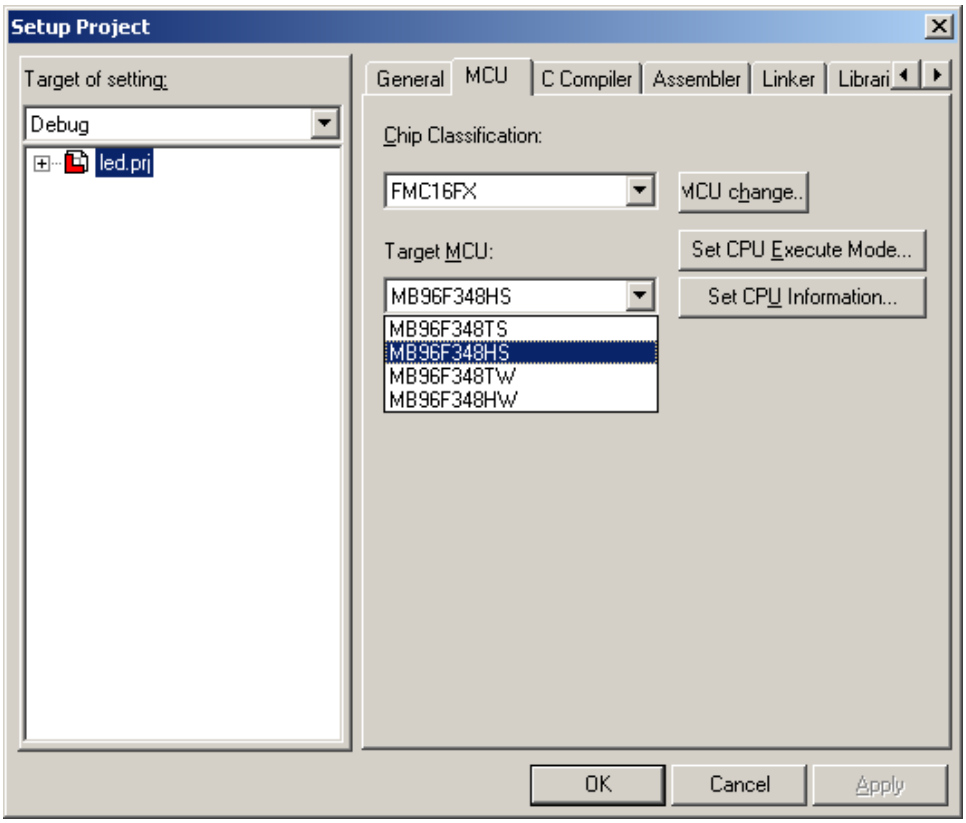
Note that especially the Linker settings in the category “Disposition/Connection” have to be adjusted device specifically.

In the chapter 7.3 is a list of all settings for the MB96F348 device.

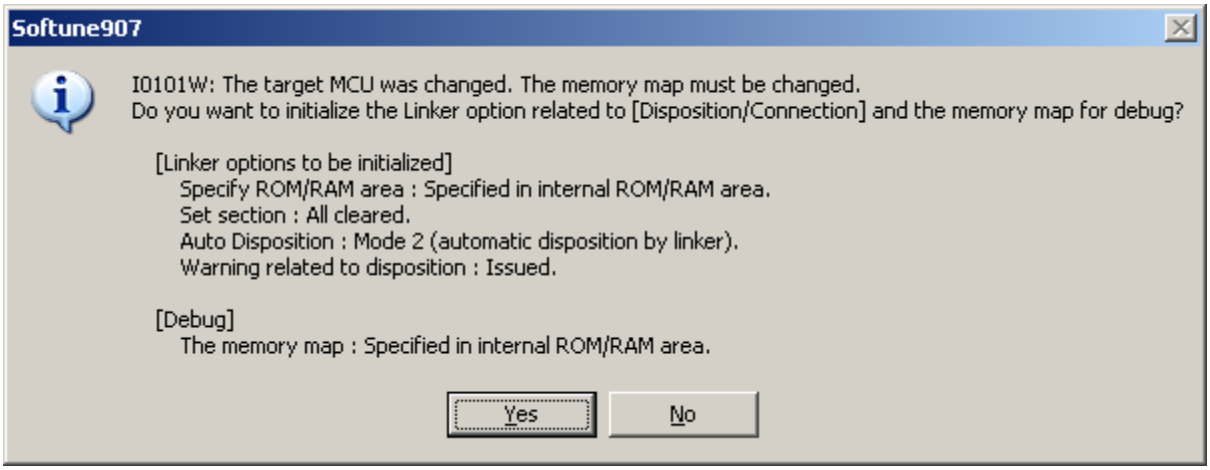
Please adjust RAM- and ROM-settings if you use a different device.

2.4 MCU change

In the project settings the MCU type can be adjusted:



Doing so, a dialog box will occur:



2.4.1 Choosing "Yes"

By choosing "yes" the correct memory size and locations will be set to the linker settings. All settings will be overwritten. Please note, that in this case the constant section is removed and have to be entered again "by hand".

2.4.2 Choosing "No"

By choosing "no", the linker settings will not be affected. Please check in this case the correct RAM and ROM size for the chosen MCU type.

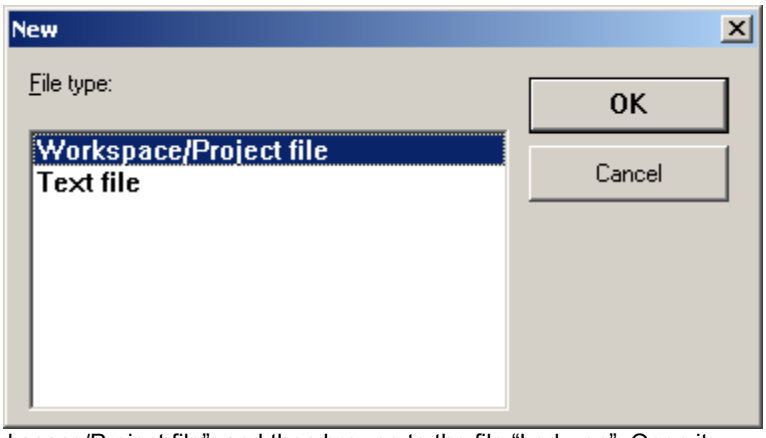
3 Creating the Project

Entering the Source Code

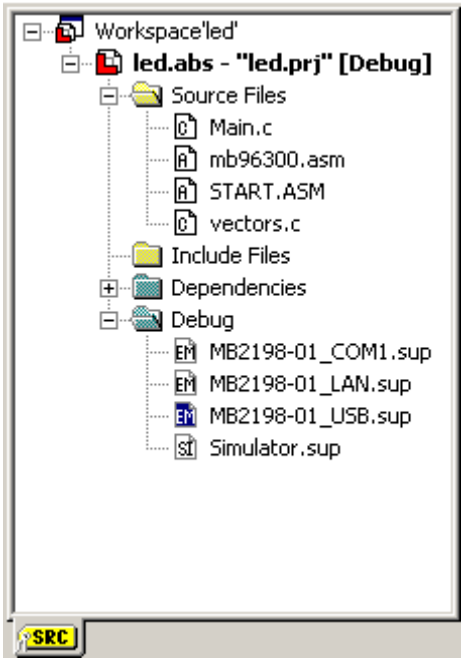
3.1 Opening the Softune Workbench

Now you can open the Softune Workbench. For installation of this tool please refer to the Installation Guide Emulator System MB2198-01 Installation Guide (002-05547).

Choose the Menu *File* > *Open*. The following pop up window will occur:



Click “OK” for “Workspace/Project file”, and then browse to the file “Led.wsp”. Open it.



Then the following project structure will be displayed on the left side of the Softune screen.

The top level is the Workspace itself: “Led”.

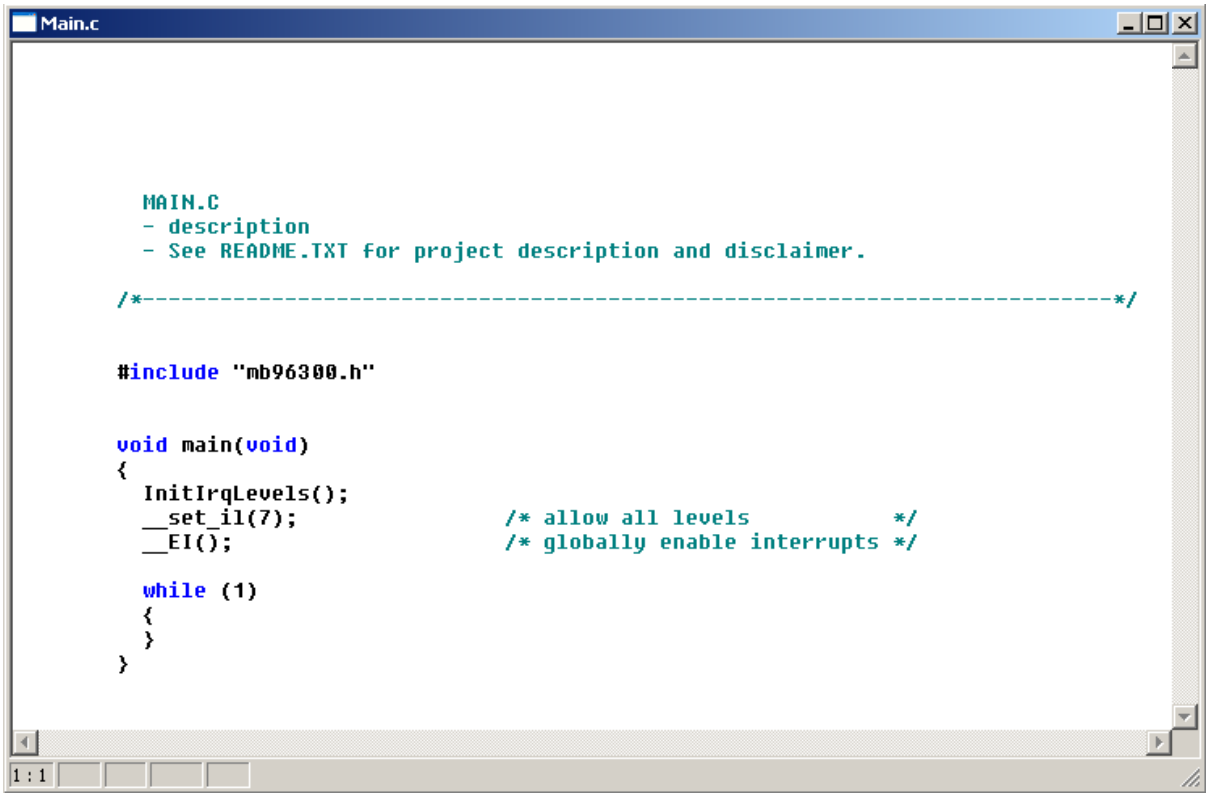
The next level contains the *ABS*-File of the Project, which contains the compiled program.

The source files, their dependencies, and the debugger settings follow.

For the moment the Source Files are in the point of interest.

Open the Source File of the Main Program *Main.c* by double clicking it in the window shown on the left.

Now a text editor window will open, which shows the contents of *Main.c*. It will look like this:



The sample code only contains the chip specific header inclusion and in the main routine a frame for the interrupt initialisation.

3.2 Entering the code

Assume that we want to have a “scrolling LED” on Port00 with a speed of 50 ms per LED.

Furthermore, we want to use an MCU Timer and an interrupt routine for the timing.

To do this, the files *Main.c* and *vectors.c* have to be edited.

3.2.1 Main.c

The following sample program example shows how to edit the main program source code for the proposed purpose:

```
  MAIN.C
  - description
  - See README.TXT for project description and disclaimer.

/*-----------------------------------------------------------------------*/

#include "mb96300.h"

unsigned char LED;                /* Global byte for LED state */

/*-------------- Sub Routines ------------------------------------------*/

void init_timer(void)             /* initialisation function for 16-bit timer */
{
    TMRLR0 = 0x61A7;              /* set reload value 25000 x 2us => 50ms */
    TMCSR0 = 0x181b;              /* prescaler 2,0us at 16 MHz */
}

void init_ports(void)
```

```
{
    DDR00 = 0xFF;                       /* Set Port00 (LEDs) to Output */
    PDR00 = 0x00;                       /* Turn off all LEDs */
}

/*-------------- Main Program -----------------------------------------------*/

void main(void)
{
    init_timer();
    init_ports();

    LED = 0x01;                         /* Prepare LED byte for 1st LED */

    InitIrqLevels();
    __set_il(7);                        /* allow all levels */
    __EI();                             /* globally enable interrupts */

    while(1);                           /* Let Interrupt handler do all things ... */
}

/*-------------- Interrupt Routine ------------------------------------------*/

__interrupt void Timer_IRQ_Handler(void)
{
    TMCSR0_UF = 0;                      /* reset underflow interrupt request flag */
    PDR00 = LED;

    LED = LED * 2;                      /* "move" LED */

    if (LED == 0)                       /* End reached? */
       LED = 0x01;                      /* Start up again at position 0 */
}
```

The program first initialises the 16-Bit Reload Timer 0, then the LED-Port.

Then the program stops in an endless loop, but every 50 ms the Reload Timer sends an interrupt request. The MCU then executes the Interrupt Service Routine, which clears the Interrupt Request and sets a new “position” of the LED.

3.2.2 Vectors.c

Before executing the program, the interrupt definition has to be done. For that, open the file “vectors.c” and edit the bold and italic entries of the following sample code example:

```
/*---------------------------------------------------------------------------
  VECTORS.C
  - Interrupt level (priority) setting
  - Interrupt vector definition

---------------------------------------------------------------------------*/

#include "mb96300.h"

/*---------------------------------------------------------------------------
   InitIrqLevels()

   This function  pre-sets all interrupt control registers. It can be used
   to set all interrupt priorities in static applications. If this file
   contains assignments to dedicated resources, verify  that the
   appropriate controller is used.

   NOTE: value 7 disables the interrupt and value 0 sets highest priority.
```

```
   NOTE: Two resource interrupts always share one ICR register.
*/

#define MIN_ICR   12
#define MAX_ICR   88

#define DEFAULT_ILM_MASK 7

void InitIrqLevels(void)
{
  int irq = MIN_ICR;

  for(irq=MIN_ICR; irq<=MAX_ICR; irq++)
  {
    ICR = (irq << 8) | DEFAULT_ILM_MASK;
  }
    ICR = (51 << 8) | 2;          /*  IRQ51 = 16-Bit Reload Timer0
}
/*---------------------------------------------------------------------------
   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);
__interrupt void Timer_IRQ_Handler (void);

/*---------------------------------------------------------------------------
   Vector definiton

   Use following statements to define vectors. All resource related
   vectors are predefined. Remaining software interrupts can be added here
   as well.
   NOTE: If software interrupts 0 to 7 are defined here, this might
   conflict with the reset vector in the start-up file.
*/

#pragma intvect DefaultIRQHandler  9    /* software interrupt 9         */
#pragma intvect DefaultIRQHandler 10    /* exception handler            */
#pragma intvect DefaultIRQHandler 11    /* NMI                          */
#pragma intvect DefaultIRQHandler 12    /* Delayed Interrupt            */
#pragma intvect DefaultIRQHandler 13    /* RC Timer                     */
#pragma intvect DefaultIRQHandler 14    /* MC Timer                     */
#pragma intvect DefaultIRQHandler 15    /* SC Timer                     */
#pragma intvect DefaultIRQHandler 16    /* reserved                     */
#pragma intvect DefaultIRQHandler 17    /* EXT0                         */
#pragma intvect DefaultIRQHandler 18    /* EXT1                         */
#pragma intvect DefaultIRQHandler 19    /* EXT2                         */
#pragma intvect DefaultIRQHandler 20    /* EXT3                         */
#pragma intvect DefaultIRQHandler 21    /* EXT4                         */
#pragma intvect DefaultIRQHandler 22    /* EXT5                         */
#pragma intvect DefaultIRQHandler 23    /* EXT6                         */
#pragma intvect DefaultIRQHandler 24    /* EXT7                         */
#pragma intvect DefaultIRQHandler 25    /* EXT8                         */
#pragma intvect DefaultIRQHandler 26    /* EXT9                         */
#pragma intvect DefaultIRQHandler 27    /* EXT10                        */
#pragma intvect DefaultIRQHandler 28    /* EXT11                        */
#pragma intvect DefaultIRQHandler 29    /* EXT12                        */
#pragma intvect DefaultIRQHandler 30    /* EXT13                        */
#pragma intvect DefaultIRQHandler 31    /* EXT14                        */
```

```
#pragma intvect DefaultIRQHandler 32    /* EXT15                          */
#pragma intvect DefaultIRQHandler 33    /* CAN0                           */
#pragma intvect DefaultIRQHandler 34    /* CAN1                           */
#pragma intvect DefaultIRQHandler 35    /* PPG0                           */
#pragma intvect DefaultIRQHandler 36    /* PPG1                           */
#pragma intvect DefaultIRQHandler 37    /* PPG2                           */
#pragma intvect DefaultIRQHandler 38    /* PPG3                           */
#pragma intvect DefaultIRQHandler 39    /* PPG4                           */
#pragma intvect DefaultIRQHandler 40    /* PPG5                           */
#pragma intvect DefaultIRQHandler 41    /* PPG6                           */
#pragma intvect DefaultIRQHandler 42    /* PPG7                           */
#pragma intvect DefaultIRQHandler 43    /* PPG8                           */
#pragma intvect DefaultIRQHandler 44    /* PPG9                           */
#pragma intvect DefaultIRQHandler 45    /* PPG10                          */
#pragma intvect DefaultIRQHandler 46    /* PPG11                          */
#pragma intvect DefaultIRQHandler 47    /* PPG12                          */
#pragma intvect DefaultIRQHandler 48    /* PPG13                          */
#pragma intvect DefaultIRQHandler 49    /* PPG14                          */
#pragma intvect DefaultIRQHandler 50    /* PPG15                          */
#pragma intvect Timer_IRQ_Handler 51    /* RLT0                           */
#pragma intvect DefaultIRQHandler 52    /* RLT1                           */
#pragma intvect DefaultIRQHandler 53    /* RLT2                           */
#pragma intvect DefaultIRQHandler 54    /* RLT3                           */
#pragma intvect DefaultIRQHandler 55    /* PPGRLT                         */
#pragma intvect DefaultIRQHandler 56    /* ICU0                           */
#pragma intvect DefaultIRQHandler 57    /* ICU1                           */
#pragma intvect DefaultIRQHandler 58    /* ICU2                           */
#pragma intvect DefaultIRQHandler 59    /* ICU3                           */
#pragma intvect DefaultIRQHandler 60    /* ICU4                           */
#pragma intvect DefaultIRQHandler 61    /* ICU5                           */
#pragma intvect DefaultIRQHandler 62    /* ICU6                           */
#pragma intvect DefaultIRQHandler 63    /* ICU7                           */
#pragma intvect DefaultIRQHandler 64    /* OCU0                           */
#pragma intvect DefaultIRQHandler 65    /* OCU1                           */
#pragma intvect DefaultIRQHandler 66    /* OCU2                           */
#pragma intvect DefaultIRQHandler 67    /* OCU3                           */
#pragma intvect DefaultIRQHandler 68    /* OCU4                           */
#pragma intvect DefaultIRQHandler 69    /* OCU5                           */
#pragma intvect DefaultIRQHandler 70    /* OCU6                           */
#pragma intvect DefaultIRQHandler 71    /* OCU7                           */
#pragma intvect DefaultIRQHandler 72    /* FRT0                           */
#pragma intvect DefaultIRQHandler 73    /* FRT1                           */
#pragma intvect DefaultIRQHandler 74    /* IIC0                           */
#pragma intvect DefaultIRQHandler 75    /* IIC1                           */
#pragma intvect DefaultIRQHandler 76    /* ADC                            */
#pragma intvect DefaultIRQHandler 77    /* ALARM0                         */
#pragma intvect DefaultIRQHandler 78    /* ALARM1                         */
#pragma intvect DefaultIRQHandler 79    /* UART0 RX                       */
#pragma intvect DefaultIRQHandler 80    /* UART0 TX                       */
#pragma intvect DefaultIRQHandler 81    /* UART1 RX                       */
#pragma intvect DefaultIRQHandler 82    /* UART1 TX                       */
#pragma intvect DefaultIRQHandler 83    /* UART2 RX                       */
#pragma intvect DefaultIRQHandler 84    /* UART2 TX                       */
#pragma intvect DefaultIRQHandler 85    /* UART3 RX                       */
#pragma intvect DefaultIRQHandler 86    /* UART3 TX                       */
#pragma intvect DefaultIRQHandler 87    /* Main Flash memory interrupt    */
#pragma intvect DefaultIRQHandler 88    /* Sat. Flash memory interrupt    */


/*-----------------------------------------------------------------------
   DefaultIRQHandler()
```

```
   This function is a placeholder for all vector definitions. Either use
   your own placeholder or add necessary code here.
*/
__interrupt
void DefaultIRQHandler (void)
{
    __DI();                                 /* disable interrupts */
    while(1)
        __wait_nop();                       /* halt system */
}
```

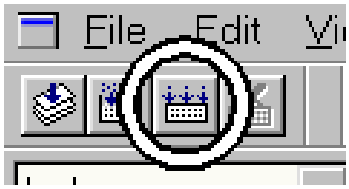
The level for the interrupt in the example above is set to 2. Because the global level threshold is set to 7 by the main program, you can choose any other number between 0 and 6.

Note that a prototype of your Interrupt Routine has to be defined as well as the vector of this routine.

3.3 Compiling the Project

Because of the usage of the “Template Project”, you now can compile this example at this moment. No other settings have to be done.

There are two ways to build to the whole project data. Either choose *Project* > *Build* or click the button in the circle in the illustration below:



If no errors occur, the following message should be displayed in the lower window:

```
Now building...
------------------Configuration: Led.prj - Debug--------------------
Start.asm
Main.c
vectors.c
Mb96300.asm
Now linking...
<Your Path>\LED\ABS\Template.abs
Now starting load module converter...
<Your path>\LED\ABS\Template.mhx

-------------------------------
No Error.
-------------------------------
```

Assume you have forgotten to type the last ';' in the "Main.c" file. Then the message will look like the following:

```
Now building...
------------------Configuration: Led.prj - Debug--------------------
Start.asm
Main.c
*** <Your path>\led\src\main.c(48) E4062C: syntax error near `}'
vectors.c
Mb96300.asm
------------------------------
Error detected.
------------------------------
```

To move quickly to the place where the error was found, simply double click the line with the red error code. The error containing line will then be highlighted red in the code window.

4 Starting Debugging

Executing the Program

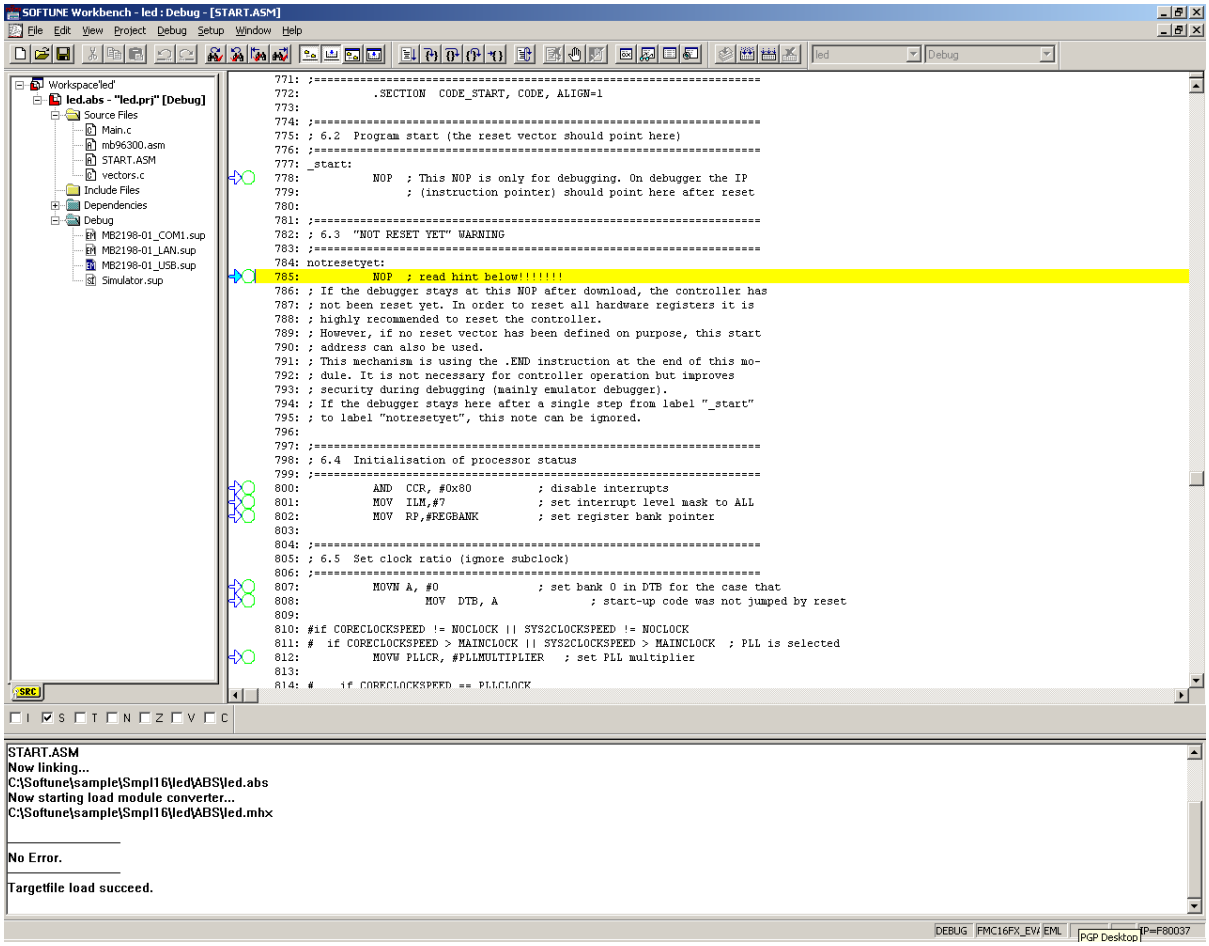
4.1 Enter Debugging Mode

After a successful project built you can enter the debugger mode via three different interfaces depending on your hardware set up:

- PC-COM-Port (RS232)
- LAN Interface
- USB Port

By double clicking on the correct connection type of the Debug dependency in the Project window (*.sup-files), your PC will connect to the MB2198 Emulator and the Softune Workbench will enter the Debugging Mode. Please refer to the Installation Guide (Emulator System MB2198-01 Installation Guide (002-05547)), for detailed information.

After a successful connection the Softune Workbench will look like this:



Note, that the interface has to be determined only once. Later you can choose the menu *Debug* > *Start debug* to enter this mode.

4.2 Executing the Program

Now choose *Debug* > *Run* > *Go*, press F5 or click the  button. Look at the Starter Kit. A “scrolling” LED has to be flashing on the Board.

To halt the execution, choose *Debug* > *Abort* or click the  button (Note that this button is only active in execution mode). The emulator system now is being halted. The execution can be continued by entering the “Go”-Mode again.

4.3 Ending the Debugging

To end the debugging just choose the menu *Debug* > *End debug*.

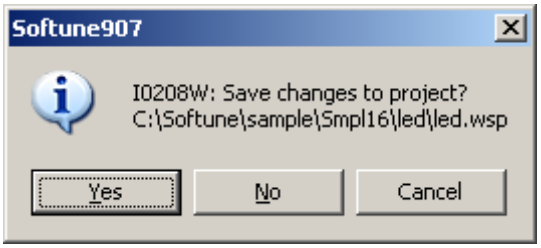
5 Closing the Project

Saving the Project Data

5.1 Last steps

To save the project data choose *File* > *Close Workspace*.

Then the following pop up window will occur.



Click “Yes”. Afterwards you can close the Softune Workbench.

Note, that in the next start of the Workbench, your new LED-Project will occur in the menu list *File* > *Recent Workspace File*. By simply clicking to “Led.wsp”, the whole project will open again.

6 Debugging

A Short Assortment Of Some Dubugger Functions

6.1 Break Points

You can set break points in your compiled code to force the execution to stop at these points.

Therefore enter the debugger mode. In general the code of the *Start.asm* file is displayed. You can easy display the code of your *Main.c* file by double clicking it in the Workspace window. Beside each command an arrow and a circle is displayed: 

Clicking in the circle sets a break point. Depending on the selected break point type, a differently coloured cross is shown in the circle. For the software break point (default), the cross is blue, while there is a red cross in the circle for the hardware break point: 

Clicking to this circle again releases the break point.

If you now start the execution, the CPU will halt on this break point. The actual line gets a yellow background colour. Now you see the flag state of the status register just under the project window.

The execution of course can be continued again by entering the run mode.

6.2 Mixed Display

Sometimes it is useful to see the compiled code together with the C source.

To display this enter the debugger mode, open the file to observe and click on the right mouse button just over the code window. A pop up window with the entry “Mix display” has to be selected. The code window now will look like this:

```
        19: void init_timer(void)                   /* initialisation function for 16-bit timer */
        20: {
        21:   TMRLR0 = 0x61A7;              /* set reload value 25000 x 2us => 50ms */
F80000: 5662A761          MOVW    I:62,#61A7
        22:   TMCSR0 = 0x181b;                      /* prescaler 2,0us at 16 MHz */
F80004: 56601B18          MOVW    I:60,#181B
        23: }
F80008: 66                RETP
        24:
```

Note that now the break points can be set exactly to the regarding CPU Op Code.

6.3 Monitoring Variables

To watch the content of a variable during stopped execution, choose *View* > *Watch*. A small new window will open. Go with the mouse pointer into it and press the right mouse button. Then choose *Set*… . Enter now the name of the variable. It will then be displayed with its name and its value in the Watch-Window.

To change the radix notation of the value, select the variable and click on the right mouse button. The menu content Radix offers now display in Binary, Octal, Decimal, and Hexadecimal formats.

6.4 Monitoring the CPU Registers

To monitor the CPU registers, choose *View* > *Register*. A small window with all CPU registers and their contents will open then.

6.5 Monitoring Memory

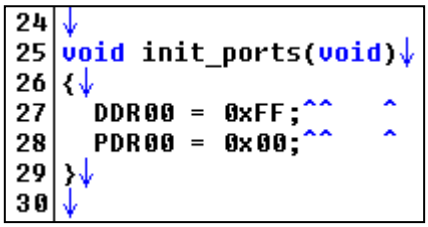
To observe memory contents, choose *View* > *Memory*. Then a small dialog window will open. You can enter the start address of the memory to be displayed.

Then a new Hex-List-Window will open. With the scroll bar you can change the view to any other address.

7 Miscellaneous

Remarks And Hints

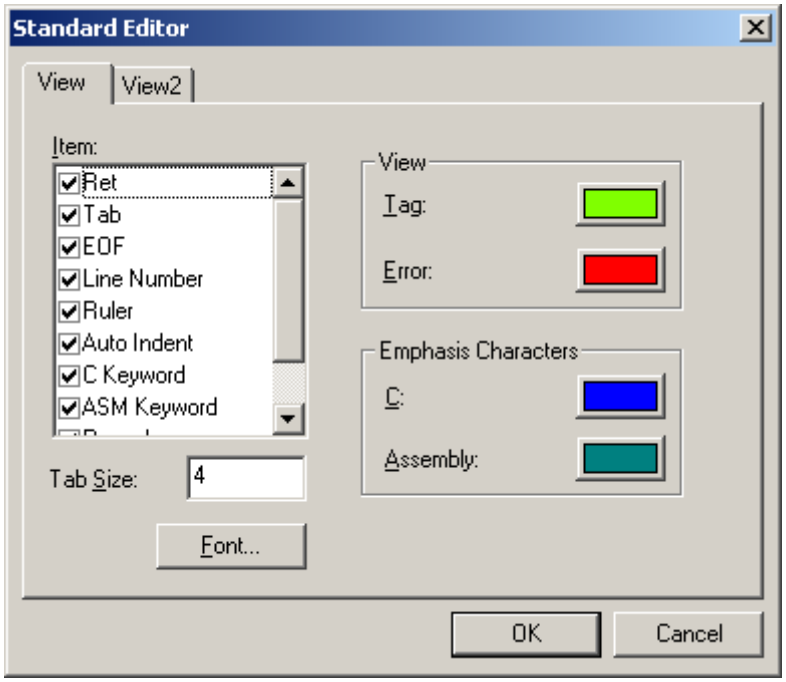
7.1 View Mode of the Editor



When starting the Softune Workbench Software for the first time, the text editor has some default viewing settings, which can be switched off. Source codes will look like in the left picture.

You can disable the viewing of the tabulators, Return signs, and the End-of-File delimiter by clicking on the right mouse button just over the text window. Then a large context menu will open.

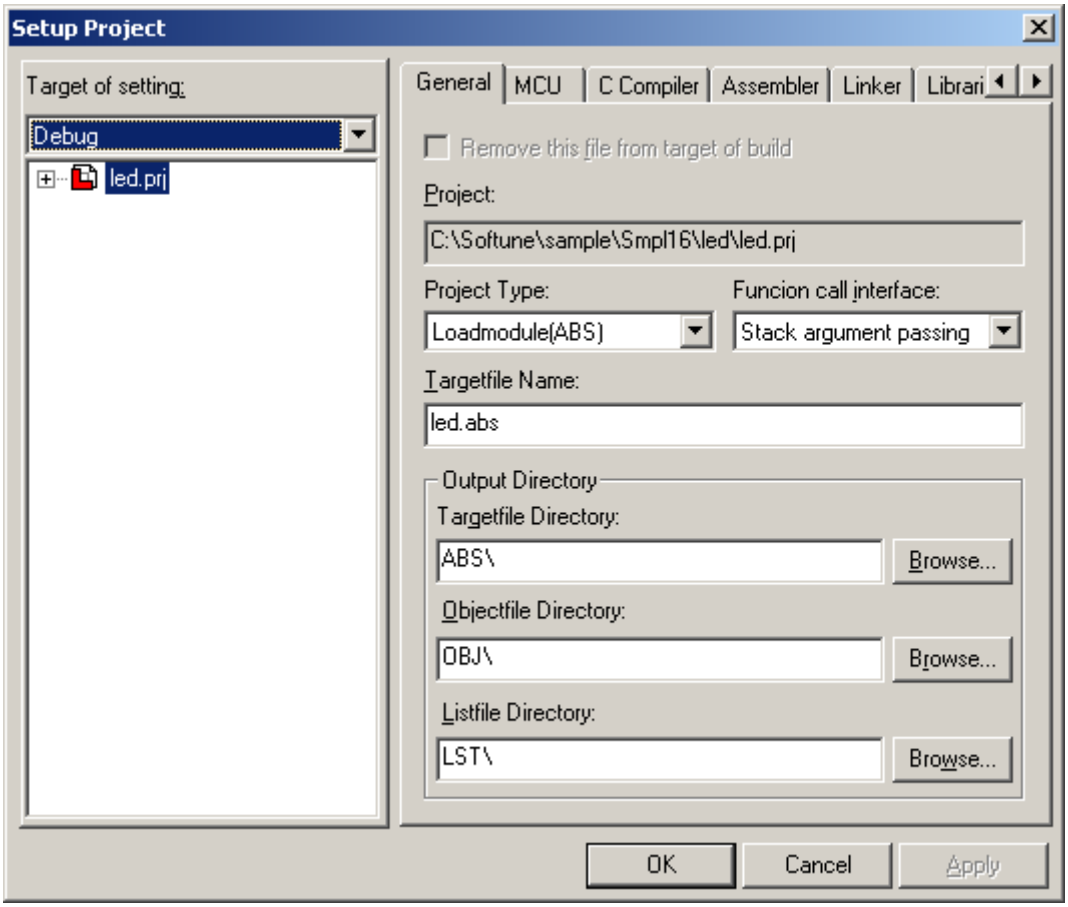
Choose *Customize*. Then the following new window will occur:



The first three entries in the item list are for selecting and deselecting the view of these non-printing characters.

7.2 Renaming the Abs-File

To change the name of the *Abs*-File, choose *Project* > *Setup Project.* The following window will be displayed:



In the text bar *Targetfile Name* you can edit the entry. Note that the new name always must have the extension ".abs".

7.3 Reference Settings for the LED-Project

Use the following default setting for MB96F348 in case of troubles.

General:

Project:	<Your Path>\LED\Led.prj
Project Type:	Loadmodule(ABS)
Target Name:	led.abs (Template.abs)
Targetfile Directory:	ABS\
Objectfile Directory:	OBJ\
Listfile Directory:	LST\

MCU:

Chip Classification:	FMC16FX
Target MCU:	MB96F348HSA

C Compiler:

Category: **General**

Outputs debug information
Warning Level: Level 3
Creates an assembly list file
Control of default option file
Other Option: -INF srcin –T p,-B

Category: **Optimize**

Speed Optimize

Category: **Language**

In-line expansion of the function which qualified by '__interrupt'
Treat as '&volatile' the variable which qualified by '__io'
Language specification level: ANSI + CYPRESS extension

Category: **Target Depend**

Memory Model: Medium

Assembler:

Category: **General**

Outputs debug information
Warning Level: Level 2
Control of default option file

Category: **Output List**

Creates a list file
Outputs information list
Outputs source list
Outputs section list
Outputs include list
Line: 60
Column: 100
Tab: 8
Macro Development Department List: OBJ

Linker:

Category: **General**

Outputs debug information
Warning Level: Level 2
Control of default option file
Other Option: -Xset_rora

Category: **Disposition/Connection***

Auto Disposition: Mode 2
RAM-Start Address: 002000
RAM-End Address: 007FFF

ROM1-Start Address: DE0000
ROM1-End Address: DE7FFF
ROM2-Start Address: DF0000
ROM2-End Address: DF7FFF
ROM3-Start Address: F80000
ROM3-End Address: FFFFFF

Setup Section → Specify in Address:
CONST/Const/WORD=H'FF8000

Category: **Register Bank**

Bank0

Note: This setting is for MB96F348. Please refer to the device specification if you are using a different MCU.

Category: **General**

Outputs debug information
Control of default option file

Converter:

Absolute module converter is started
Control of default option file
Output Data Format: Motorola S Format

Debug:

Category: **Setup**

Setup Name List:
MB2198-01_COMn (n=1, 2, 3, 4)
MB2198-01_LAN
MB2198-01_USB
Simulator

8 Installing LAN

This Chapter Describes How To Install The Lan Interface

8.1 Overview

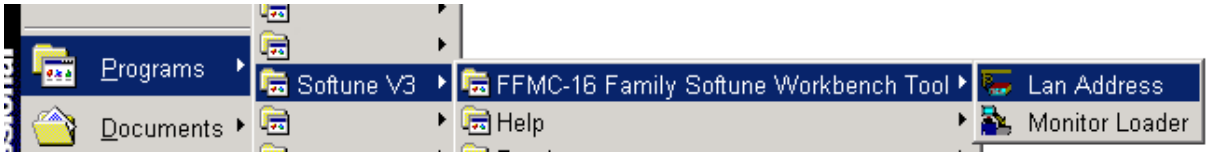
The emulator MB2198-01 is equipped with a local area network adapter that can be used to program and debug a device over a network connection. No additional hardware connection except the LAN connection itself is required for this purpose

Using the Cypress LAN-remote controlled debug facility,

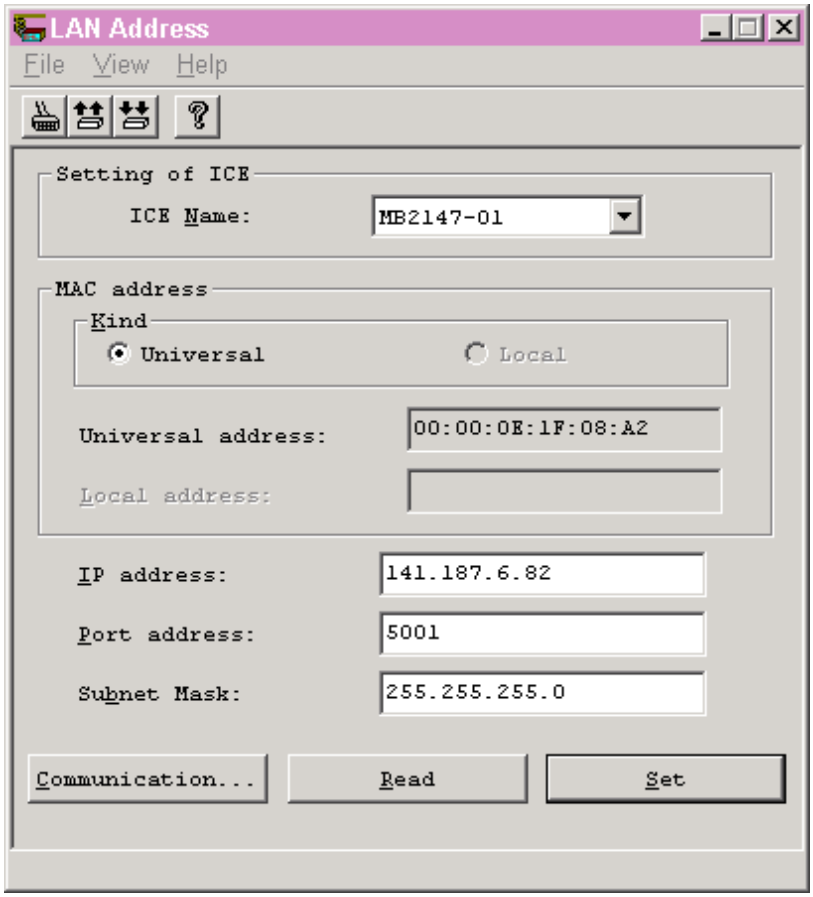
- You can control the emulator from different locations without having to move your hardware installation from one place to another.
- Program download will be more than 6 times faster than with RS232C

8.2 Configuring the LAN Adapter

1. Refer to the LAN Installation Manual coming with the LAN Adapter.
   This Application Note will give only some additional information.
2. Connect emulator and PC by the serial RS232-Port (or USB-Port)
3. Start the „LAN Address“ Program in the Softune Workbench Folder



4. Click on *Communication* to define the serial RS232 Port where the emulator is connected to or use USB connection.
5. Read current status from the emulator
6. Set the unique IP address given from your network manager. This is a very important point, every IP-address within a network has to be unique.
7. Check the Port address: must be 5001
8. Make settings valid by „Set“, and reset the emulator when prompted
9. Close Program „LAN Address“



8.3 Configuring Operating System "Windows™"

1. Within your Windows™-directory (e.g. *C:\windows* or *C:\WINNT\system32\drivers\etc*) you should find three files:

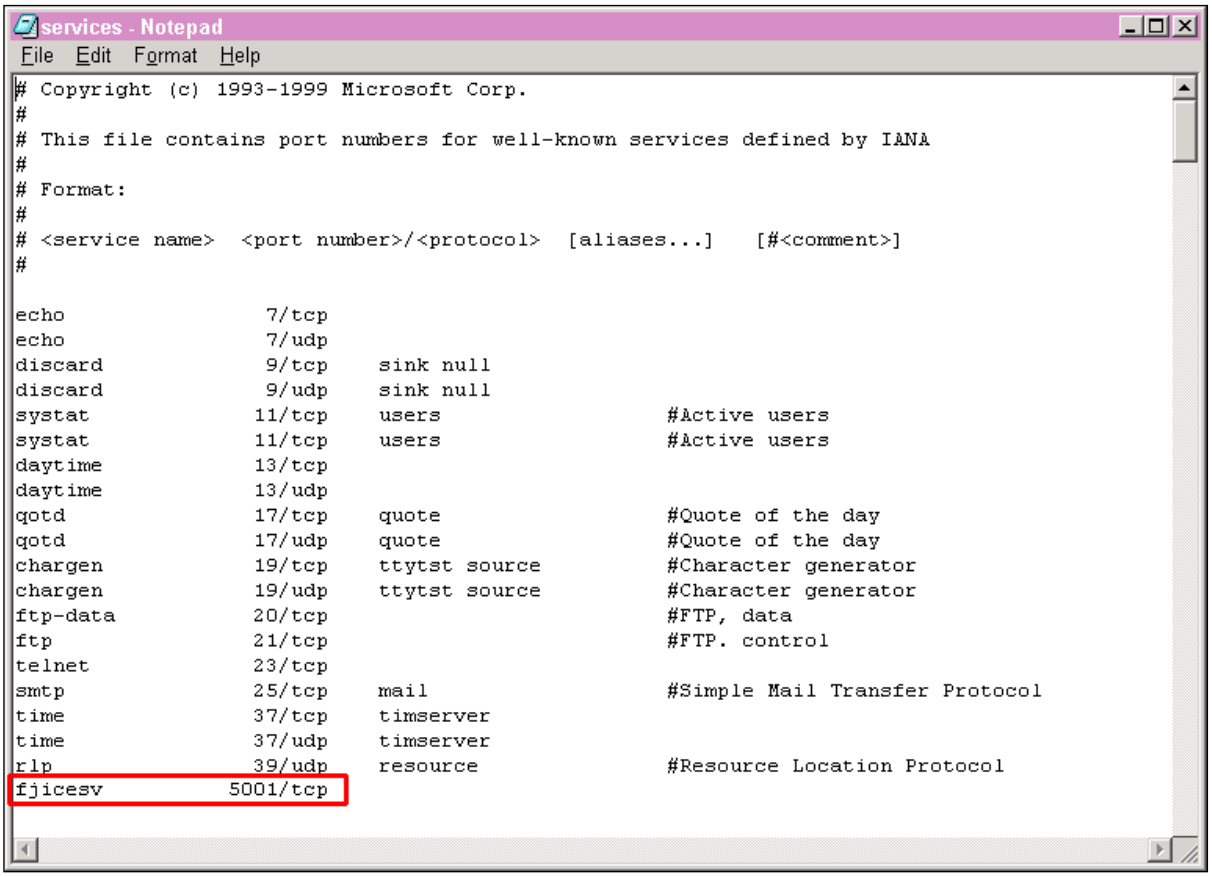
   ➔ *services*
   ➔ *hosts*
   ➔ *lmhosts*[*.sam*]

   (Note: The files may not have a file-extension!)

2. Make a copy of all three files, like for an example: *services* to *services.old* , etc.

   Edit file *services* and add the following line:

```
fjicesv       5001/tcp             # Cypress emulator
```

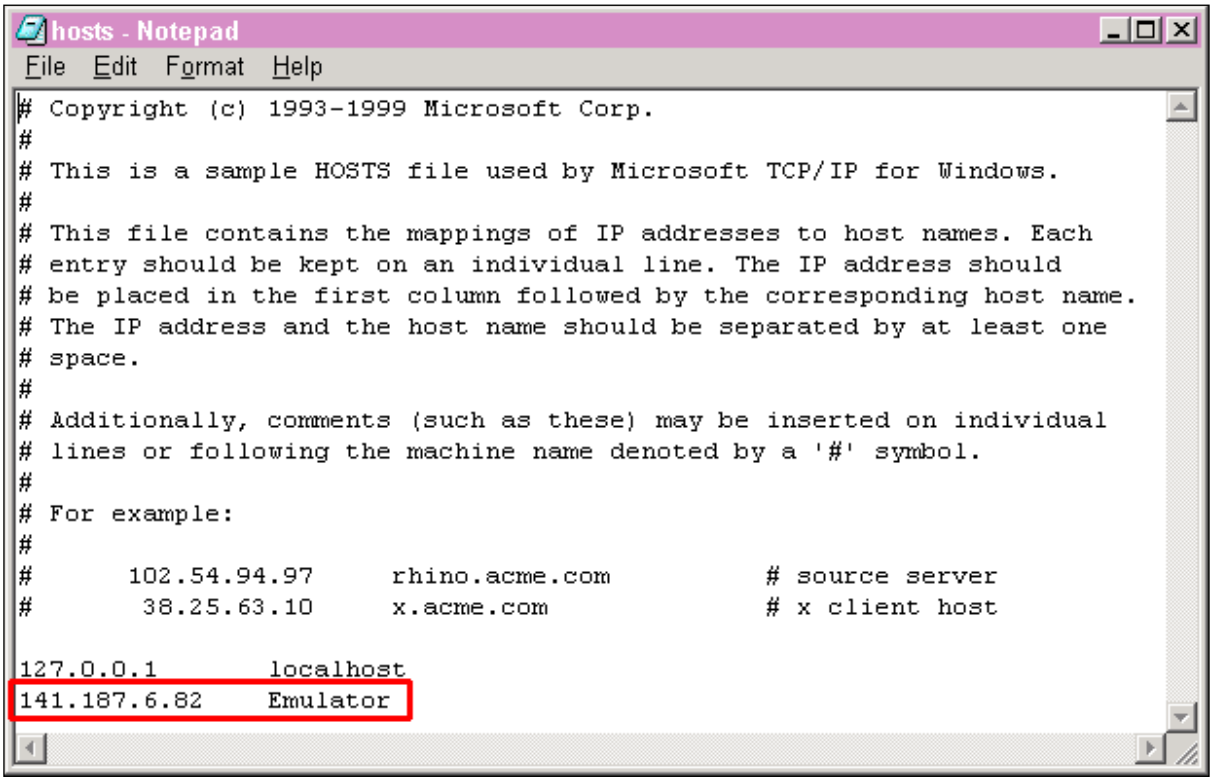


Note: If `5001/tcp` is already contained in the file *services*, use an unused number beginning with 5002 or greater, e.g. `fjicesv 5002/tcp`. In that case also the emulator address given by the program „LAN address" (see above) has to be changed!

3. When saving again the file *Services* be sure that your editor (e.g. notepad) will not add any extension, e.g. .txt, to your file. To be sure, use quotation marks for the filename:
File save as: **"***services***"**

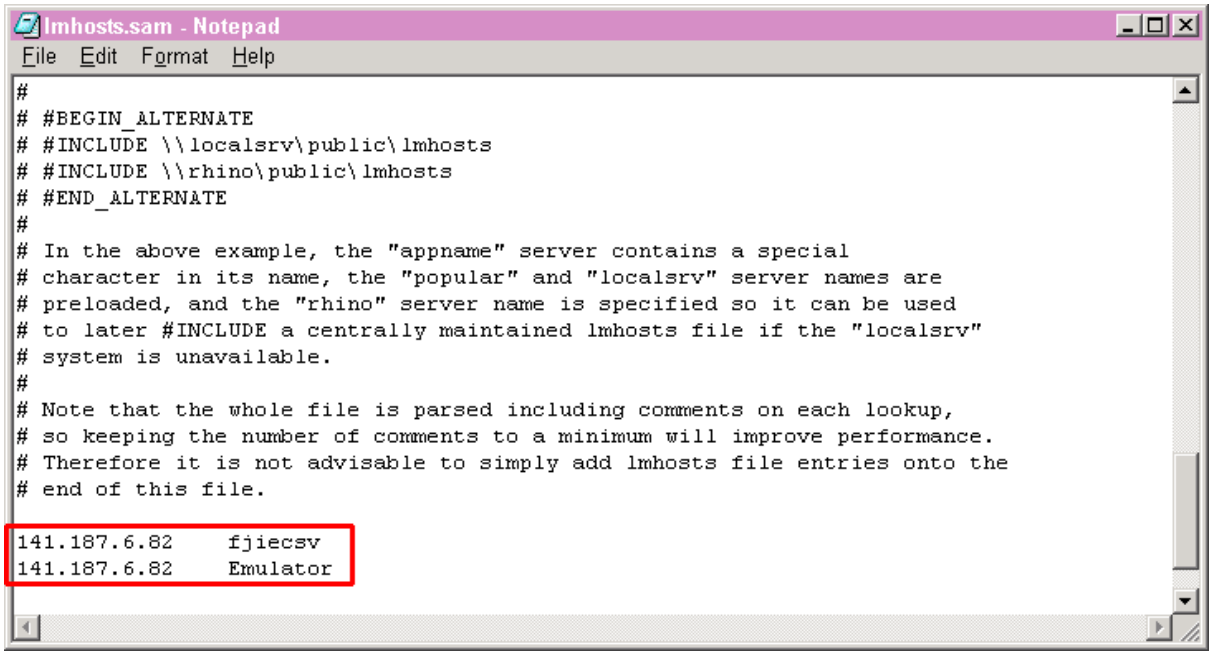
4. The file *hosts* is used to make a redefinition of the complex IP-number with a simple name within your global network. This may be important if you use a DNS-Server. Of course, you can define different names for the same IP-Address, as shown below. Edit file *hosts* and add the following line:

   „The unique IP address (as set by LAN-Address, see above)“ „Nickname of emulator“
   e.g. `141.187.6.82 Emulator`



   When saving again the file *Hosts* be sure that your editor (e.g. notepad) will not add any extension, e.g. *.txt*, to your file. To be sure, use quotation marks for the filename:
   File save as: **”***hosts***”**

5. The file *lmhosts* (or *lmhosts.sam*) is used to make a redefinition of the complex IP-number with a simple name within your local network.
   Of course, you can define different names for the same IP-Address, as shown below. Edit file *lmhosts* and add the following line:



„the unique IP address (as set by LAN-Address, see above)“ „nickname of emulator“
e.g.
`141.187.6.53 fjicescv`
`141.187.6.53 Emulator`

When saving again the file *Lmhosts* get sure that your editor (e.g. notepad) will not add any extension, e.g. .txt, to your file. To be sure, use quotation marks for the filename:
e.g. File save as: **”***lmhosts***”** (or *lmhosts.sam*)

8.4 Checking the network-connection

Disconnect serial RS232 (or USB) cable from Emulator and try to find the emulator:

- Open DOS-Window (or open RUN (“Ausführen” in german Windows™) in the Start-Menu)
- The command `ping Emulator` should acknowledge with some time-values, that means the network is set up right.

The emulator with LAN-adapter is successfully integrated in the network environment and can be used by the Softune Workbench.

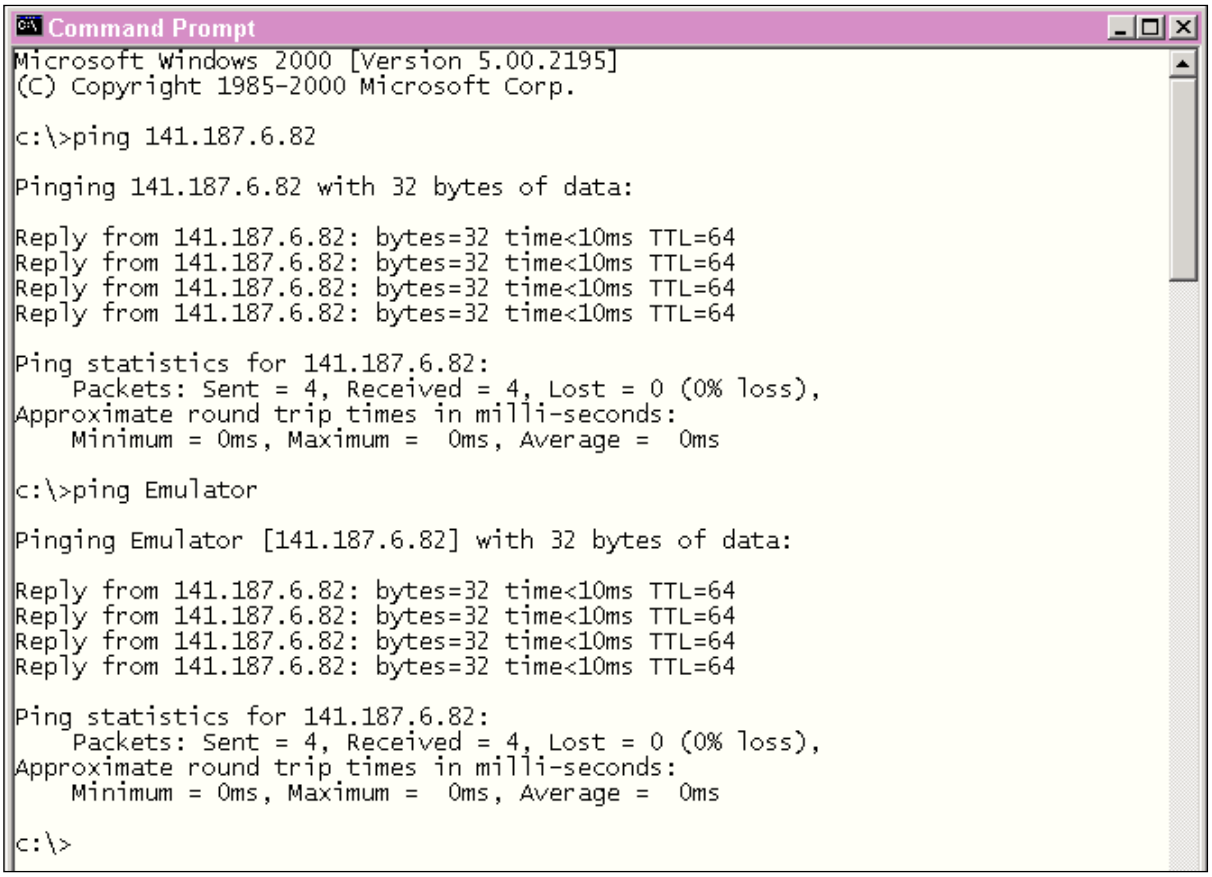
8.5 Troubleshooting

If the command `ping my_emulator` will reply with a timeout-message, try to find the emulator by its IP-address:
Type command `ping` *`IP-address`*, e.g. `ping 141.187.6.53`.

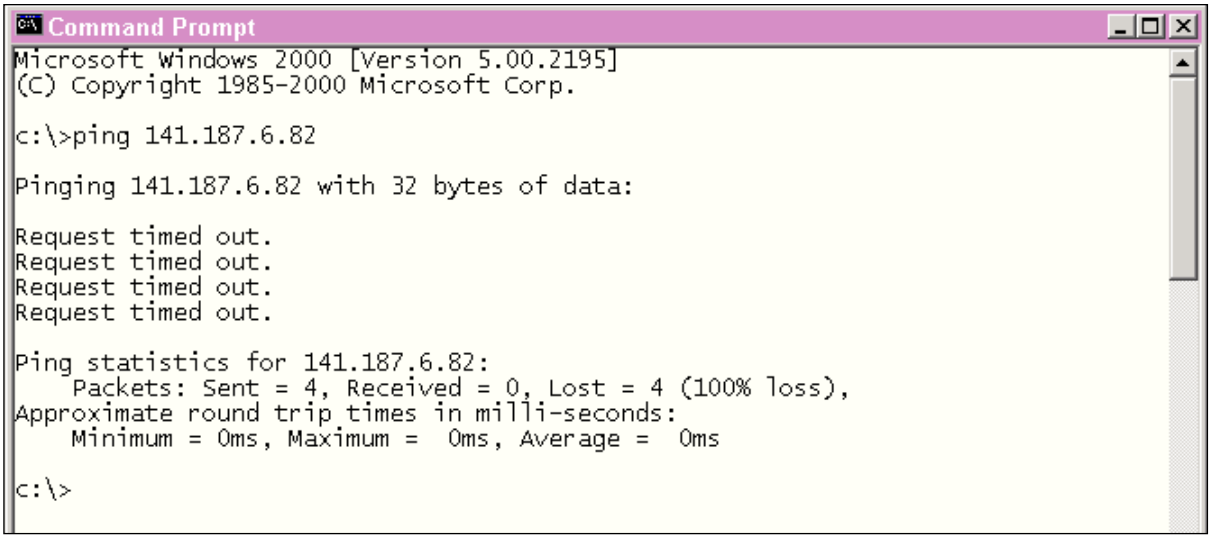
If this will work, then check settings (nickname and IP-address) within *Lmhosts* and *Hosts*.
If neither nickname (`my_emulator`) nor IP-address works, check settings done by the program *LAN-address* and check the file *Services*.

Also check your physical network interconnection cables. Please keep in mind that the emulator only works with 10Mbit/s. This means in case that for a 100Mbit/s network a 100Mbits/10Mbit - HUB is needed. When using a HUB for 10Base-T then a standard (1:1) network cable has to be used. If the emulator is connected directly to the PC a “crossed network cable”is necessary.

Good answer: Emulator/LAN-adapter replies with time-values:

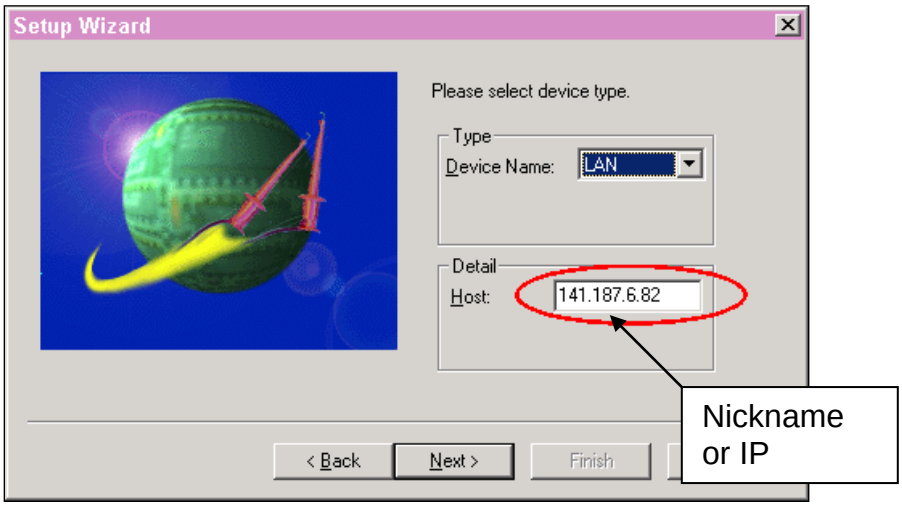


“Failed”-answer: Emulator/LAN-adapter replies with timeout message:



8.6 Softune Workbench

Within the Softune Workbench the LAN interface can be used instead of a serial RS232C communication. Detail-Host: can be the „nickname“ as defined in the files *hosts* and *lmhosts* or the IP-address of the emulator, set by the program „LAN-Address“, can be used.


Setup Wizard
Please select device type.
Type
Device Name:
LAN
Detail
Host:
141.187.6.82
< Back
Next >
Finish
Nickname
or IP

Document History

Document Title: AN205555 - F²MC-16FX Family, Emulator System MB2198

Document Number: 002-05555

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	12/14/2006	Initial Release
*A	5080524	NOFL	03/10/2016	Migrated Spansion Application Note from MCU-AN-300217-E-V10 to Cypress format
*B	5868913	AESATP12	08/31/2017	Updated logo and copyright.

Worldwide Sales and Design Support

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

Products

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

PSoC® Solutions

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

Cypress Developer Community

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

Technical Support

cypress.com/support

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