TWR-LCD

Demo Projects Walk-Through

Rev. 1.0

Freescale Semiconductor Inc.

Contents

Lab 1: Getting starting with the TWR-LCD

The following lab will guide the user through the pre-flashed Freescale Embedded GUI application, entering and using the built-in bootloader, and use of an additional pre-compiled GUI application based on the Freescale Embedded GUI Drivers and CodeWarrior Processor Expert components.

1.1 Demo Setup

Following is assumed

- CodeWarrior for MCU V6.3
- TWR-LCD Rev A board
- TWR-LCD has pre-flashed bootloader plus Freescale Embedded GUI application on it (factory default): 'JM128 Bootloader.S19' plus 'JM128_BL_EGUI_SPI.S19'
- Factory default switches (DIP SW 1: 1:OFF, 2: ON, 3:OFF, 4: OFF, 5:ON, 6: ON, 7:ON, 8:OFF), SW5: all OFF
- S19 files for Freescale Embedded GUI (EGUI), ECUI (Embedded Component UI) and I2C demo installed/available

1.2 Freescale Embedded GUI Demo

- Connect the TWR-LCD with your host PC to power up the board
- If the touch screen has not been calibrated, it will show a blue calibration screen the first time. Touch the 3 crosses as accurate as possible. Then the calibration values will be stored in FLASH memory of the JM128.

Figure 1: Touchscreen Calibration

- The Freescale Embedded GUI demo screen will show up. Use the touch screen to select demos. Alternatively you can use the navigation switch: left/right to move forward and backward, center to select/focus and up to deselect focus.

Figure 2: Freescale Embedded GUI Demo

- For more information about the Freescale Embedded GUI, see the additional documentation.

1.3 TWR-LCD Bootloader

The TWR-LCD features a bootloader to facilitate the loading of applications without the need for an external debugger.

- To enter the bootloader hold the 'BTLD' button while momentarily pressing the 'JMRST' button and finally releasing 'BTLD'.
- In bootloader mode, you will hear a beep from the sounder and the screen will write a welcome message.
- The bootloader will enumerate the TWR-LCD JM128 as a MSD (Mass Storage Device). Windows will recognize your board and appear as a removable storage drive labeled "BOOTLOADER".
- The device will show an empty file named 'READY.TXT' on it.
- Now drag&drop/copy a bootloader compatible S19 file to the device. The installation comes with the 'JM128_BL_ECUI_SPI.S19' file in the precompiled project folder. Drag this file to the bootloader device on your windows machine
- After bootloading the new file is finished, you will hear two beeps from the sounder, the LCD will show the progress and status, and the bootloader USB MSD device will show the empty file 'SUCCESS.TXT' on it.
- Press the 'JMRST' button to reset the board. The bootloader will recognize that a valid application has been loaded to the device and launch it

1.4 Embedded Component UI (ECUI) Demo

- This launches the ECUI demo. As the application flash has been erased, it will ask for a calibration first as well like in the previous demo.

- This demo is using the same low level drivers as the previous demo, but using a different UI based on Processor Expert components.
- You can use the touch screen to select demos, or alternatively use the navigation switch (up/down and left/right to navigate, enter to select/execute items)
- For more information about the Embedded Component UI, see the online documentation provided with the Processor Expert Embedded Components.

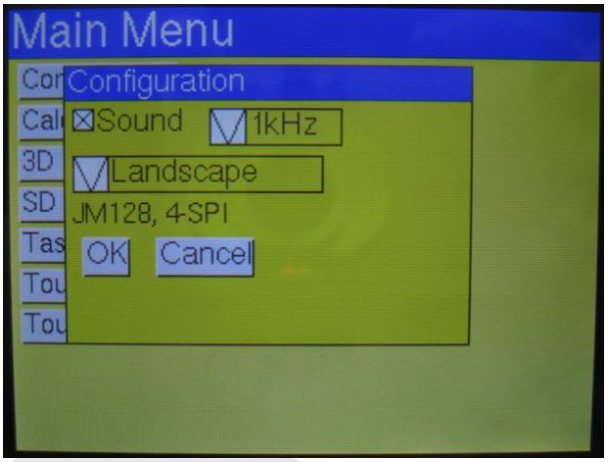

Figure 3: Processor Expert Embedded Component GUI Demo

1.5 Precompile Applications for the TWR-LCD USB Bootloader

The included CodeWarrior demo project folder included with the TWR-LCD contains the following precompiled applications that can be loaded to the MCF51JM128 on the TWR-LCD using the USB Bootloader:

- JM128_BL_EGUI_SPI
  This is the default application which comes pre-flashed onto MCF51JM128 of the TWR-LCD. The TWR-LCD display is driven using the SPI interface of the MCF51JM128. The application features the Freescale Embedded GUI demo based on the Freescale Embedded GUI Drivers. This application includes the TWR-LCD bootloader to enable flashing of the device via the USB cable.

- JM128_BL_ECUI_SPI
  The application features an alternative GUI based on the same Freescale Embedded GUI Drivers, but implemented using Processor Expert components. The TWR-LCD display is driven using the SPI interface of the MCF51JM128. This application also includes the TWR-LCD bootloader to enable flashing of the device via the USB cable.

- JM128_Bootloader
  This application includes only the TWR-LCD bootloader. This purpose of the application is to enable flashing of the MCF51JM128 device via the USB cable and to initialize the MCF51JM128 pins to ensure that, if required, they are properly driven to enable

communication of an additional Freescale Tower Controller Module to the TWR-LCD display and peripherals, such as the SD Card slot.

- JM128_BL_TWR_I2C
  This application includes the TWR-LCD bootloader and is a basic application to send I2C messages to the Freescale Tower Controller Module. The purpose of the application is to send I2C messages regarding the state of the Navigation Switch, and to initialize the MCF51JM128 pins to ensure that, if required, they are properly driven to enable communication of an additional Freescale Tower Controller Module to the TWR-LCD display and peripherals, such as the SD Card slot.

1.6 Example Applications for the onboard JM128

The included CodeWarrior demo project folder included with the TWR-LCD also contains the following applications that can be loaded to the MCF51JM128 on the TWR-LCD using the BDM cable (no Bootloader required):

- JM128_noBL_EGUI_SPI
  This application is similar to the JM128_BL_EGUI_SPI application, with the exception of the bootloader. Once install the TWR-LCD flash will no longer contain the USB bootloader and must be reprogrammed use a BDM cable.

- JM128_noBL_ECUI_SPI
  This application is similar to the JM128_BL_ECUI_SPI application, with the exception of the bootloader. Once install the TWR-LCD flash will no longer contain the USB bootloader and must be reprogrammed use a BDM cable.

- Demo_MCF51JM_SPI
  This application features the Freescale Embedded GUI demo. This demo is similar to the JM128_BL_EGUI_SPI demo, but was not build using Processor Expert. It provides an example of using the Freescale Embedded GUI drivers directly. Once install the TWR-LCD flash will no longer contain the USB bootloader and must be reprogrammed use a BDM cable.

- HelloWorld_MCF51JM_SPI
  This application is a very simple “Hello World” demo build using the Freescale Embedded GUI driver. This demo can be used as a beginning reference is using the Freescale Embedded GUI drivers. Once install the TWR-LCD flash will no longer contain the USB bootloader and must be reprogrammed use a BDM cable.

1.7 Example Applications for the TWR-MCF51CN

The included CodeWarrior demo project folder included with the TWR-LCD also contains the following applications that can be loaded to the TWR-MCF51CN Tower Controller Module using OSBDM:

- Demo_MCF51CN_Flex
  This application features the Freescale Embedded GUI demo. This demo is similar to the CN128_EGUI_Flexbus_Accel demo, but was not build using Processor Expert. It provides an example of using the Freescale Embedded GUI drivers directly.

- Demo_MCF51CN_SPI
  This application features the Freescale Embedded GUI demo. This demo is similar to the Demo_MCF51CN_Flex demo, but uses the SPI interface to drive the LDC display. It provides an example of using the Freescale Embedded GUI drivers directly.

- HelloWorld_MCF51CN_Flex
  This application is a very simple “Hello World” demo build using the Freescale Embedded GUI driver. This demo can be used as a beginning reference is using the Freescale Embedded GUI drivers. The demo uses the TWR-MCF51CN to drive the LCD display using the SPI.

- HelloWorld_MCF51CN_SPI
  This application is a very simple “Hello World” demo build using the Freescale Embedded GUI driver. This demo can be used as a beginning reference is using the Freescale Embedded GUI drivers. The demo uses the TWR-MCF51CN to drive the LCD display using the SPI.

- CN128_EGUI_Flexbus_Accel
  This application features the Freescale Embedded GUI demo based on the Freescale Embedded GUI Drivers, similar to the JM128_noBL_EGUI_SPI but targeted to run on the TWR-MCF51CN Tower Controller Module. This application interfaces to the TWR-LCD display using Flexbus, an External Bus Interface (EBI). Additionally this application utilizes the TWR-MCF51CN accelerometer.

- CN128_ECUI_Flexbus_Accel
  The application features an alternative GUI based on the same Freescale Embedded GUI Drivers, but implemented using Processor Expert components, similar to the JM128_noBL_ECUI_SPI but targeted to run on the TWR-MCF51CN Tower Controller Module. This application interfaces to the TWR-LCD display using Flexbus, an External Bus Interface (EBI). Additionally this application also utilizes the TWR-MCF51CN accelerometer.

- CN128_ECUI_Flexbus_SD
  The application features the alternative GUI based on the same Freescale Embedded GUI Drivers, but implemented using Processor Expert components, similar to the CN128_ECUI_Flexbus_Accel application. This application interfaces to the TWR-LCD display using Flexbus, an External Bus Interface (EBI). Additionally this application features the ability to access the TWR-LCD SD Card slot from the TWR-MCF51CN.

Lab 2: TWR-LCD Bootloader

This document describes the CodeWarrior projects for the TWR-LCD board. It is assumed that CodeWarrior for MCU 6.3 is used.

2.1 Installing Processor Expert TWR-LCD Embedded Components

The TWR-LCD CodeWarrior project is using Processor Expert components. In a first step you need to install the components using the .PEupd file provided.

- Launch CodeWarrior IDE
- Select the menu 'Processor Expert' > 'Update' > 'Update Processor Expert from Package'

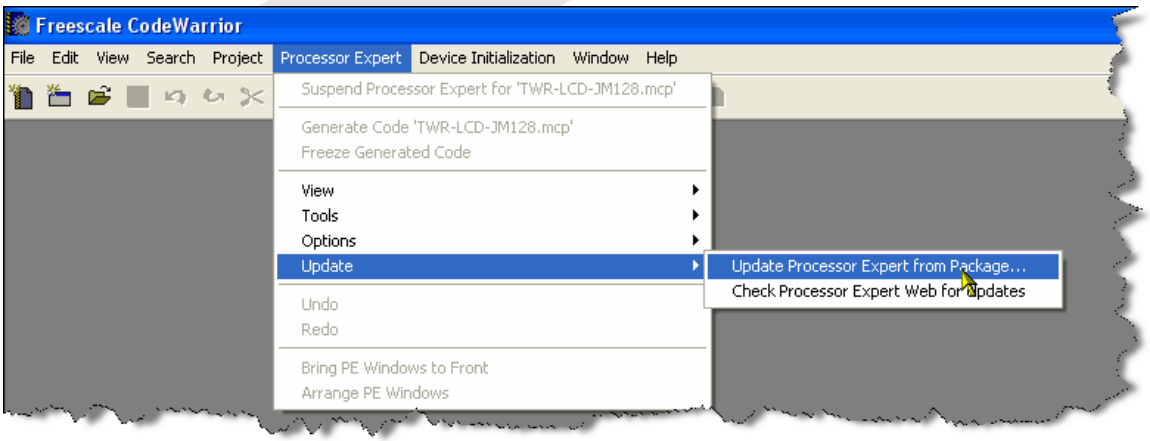

Figure 4: Updating Processor Expert from Package

- Browse to the 'TWR-LCD_Components.PEupd' file located in the root of your CodeWarrior demo project folder and import all the components
- Close CodeWarrior IDE and restart CodeWarrior: this will ensure the new components are recognized
- Load now the CodeWarrior project 'TWR-LCD-JM128.mcp' from the Processor Expert folder.

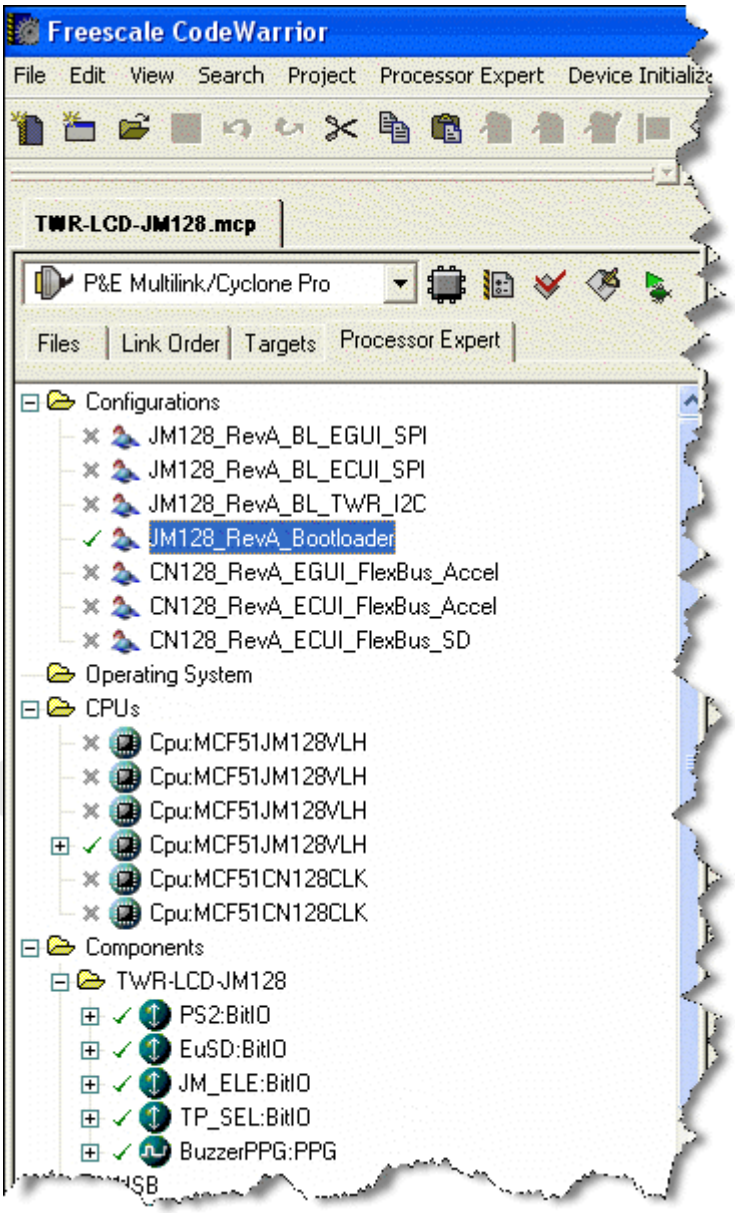

Figure 5: CodeWarrior project open

2.2 Building the MCF51JM128 Bootloader

The bootloader allows you to download new applications to the MCF51JM128, without the need for a debug cable. However, you need first to program the bootloader to the TWR-LCD (if it does not already have the bootloader on it). To program the bootloader you need a BDM cable (e.g. P&E USB Multilink) to flash the bootloader. Additionally you need to connect the TWR-LCD board USB connector with your host system, as the bootloader is getting the S19 files from the host through a USB connection.

To build the boot loader, make sure your current CPU is the JM128:

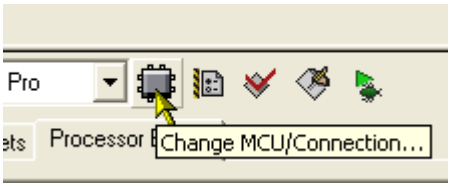

Figure 6: Change MCU/Connection

'Change MCU/Connections...' opens the following dialog:

Verify that your target CPU is the MCF51JM128:

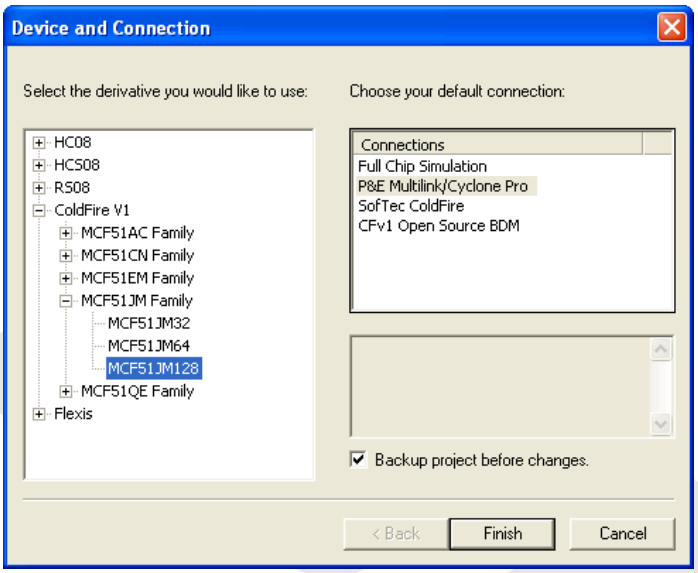

Figure 7: Change MCU/Connection

Verify that your current configuration is the bootloader one. If not, select it as active configuration:

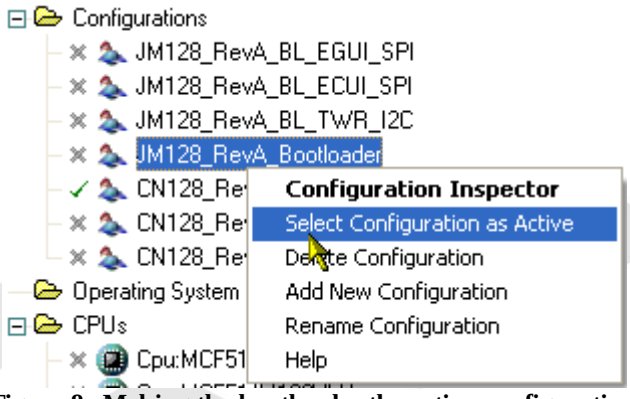

Figure 8: Making the bootloader the active configuration

Using the Configuration inspector, each configuration provides additional information. If you hover over a configuration, it will show up with a pop-up window:

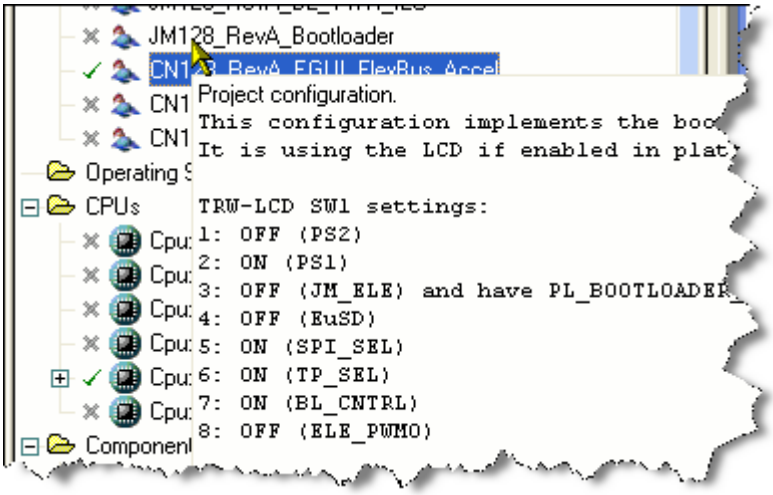

Figure 9: Configuration details pop-up window

If you build the bootloader, you might get a linker error about multiply defined flash registers:

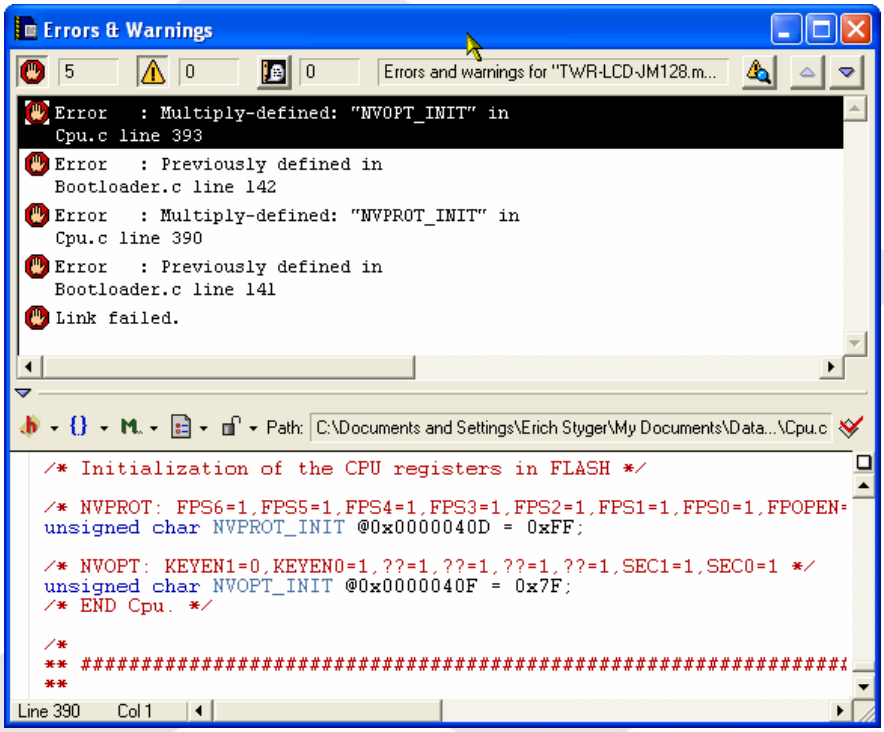

Figure 10: Linker error for NVPROT_INIT and NVOPT_INIT

The reason is that the generated code for the bootloader by Processor Expert and the bootloader code itself are initializing the NVPROT_INIT and the NVOPT_INIT registers. Future versions of Processor Expert will have an option to prevent initializing NVPROT_INIT and NVOPT_INIT.

Solution: comment/disable the above two initializations in Cpu.c and recompile/relink:

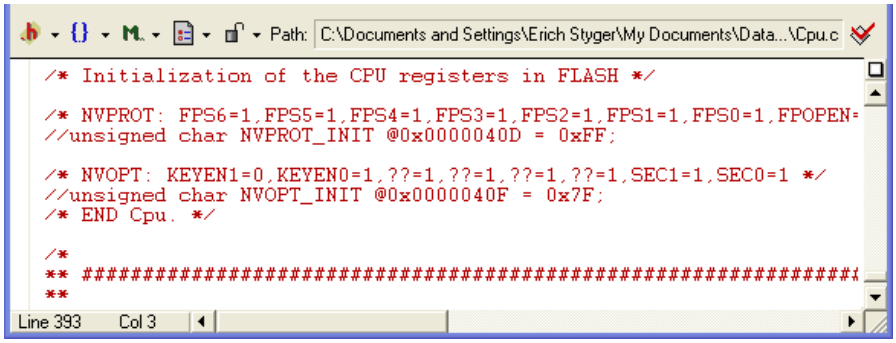

Figure 11: Workaround for linker error for NVPROT_INIT and NVOPT_INIT

After successful build, you can download/flash the bootloader to the MCF51JM128:

Figure 12: Program the bootloader to the target

Using 'Start/Continue (F5)' you can launch the bootloader:

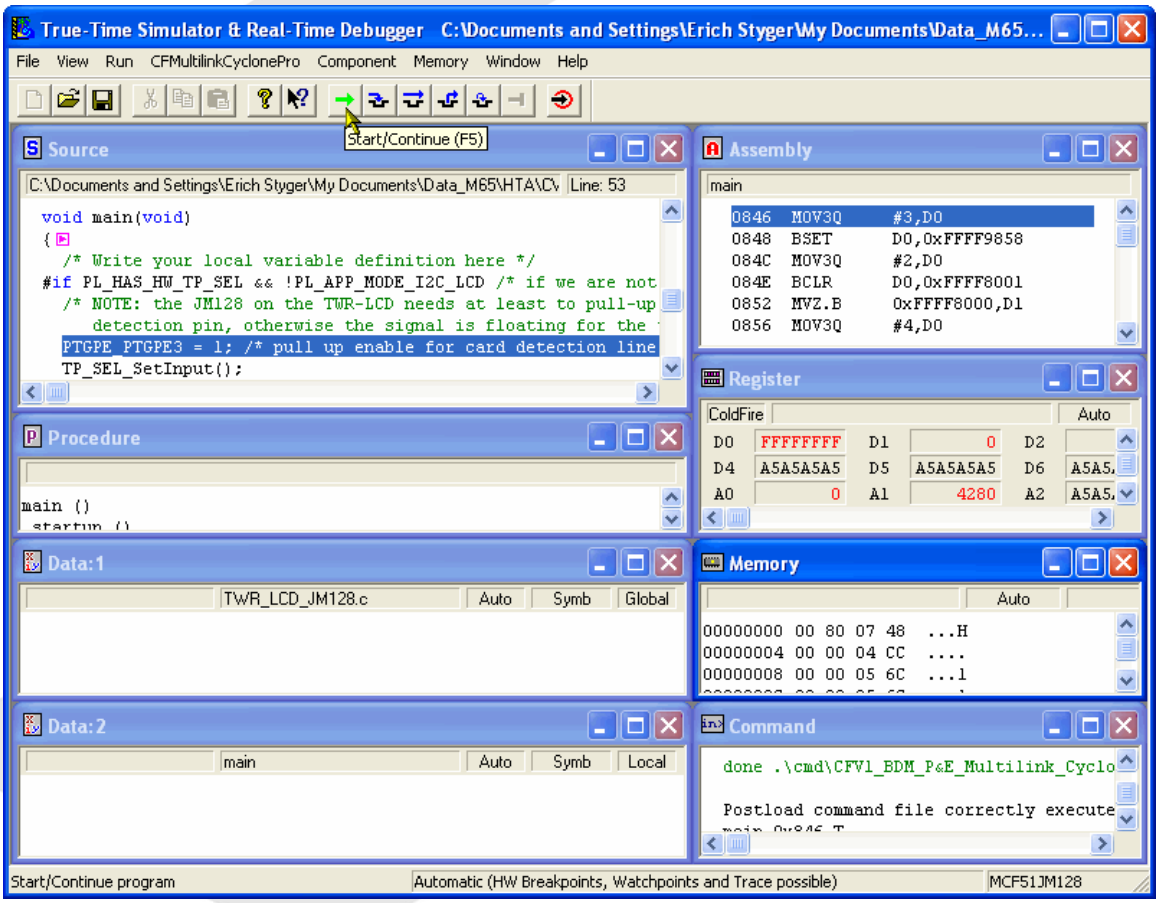

Figure 13: Ready to launch the bootloader

2.3 Using the MCF51JM128 Bootloader

Your TWR-LCD shall come with the bootloader already flashed. The bootloader allows you to load applications to the target without the need for an external debugger.
The bootloader is entering bootloader mode in following cases

- if there is no application loaded, the bootloader will recognize this and automatically enter the bootloader mode
- if an application is already loaded, then you need to reset the board (press the JMRST button) while holding down the BTLD button.

Once the bootloader has been started, it will you will hear a 'beep', the LCD will show message:

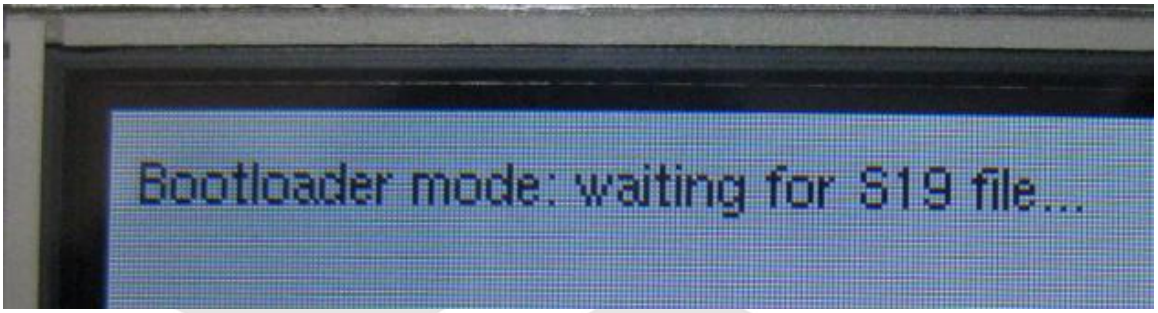

Figure 14: Bootloader LCD message

The windows host will recognize the bootloader as FAT16 mass storage device:

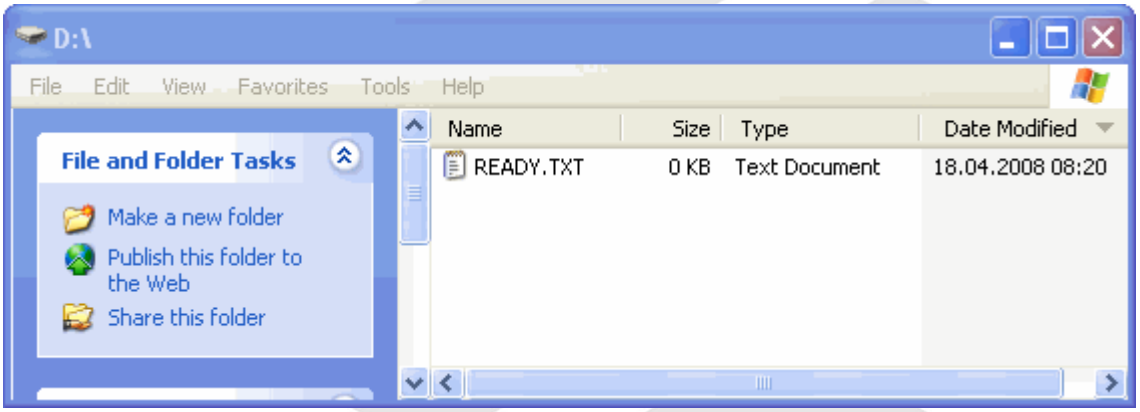

Figure 15: Bootloader recognized as mass storage device

Now you can drag&drop / copy S19 (Motorola S-Records) files to the bootloader:

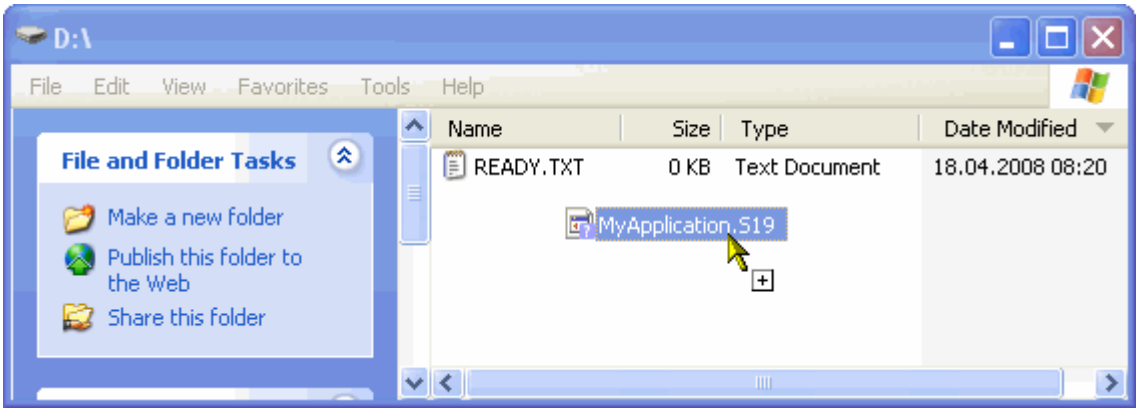

Figure 16: Drag&Drop S19 file to bootloader

The bootloader will load the file, parse it and flash the application to the target. Progress of this is shown on the LCD display.
If the downloading is successful, you see this indicated on the LCD, plus you will hear two 'beeps'.

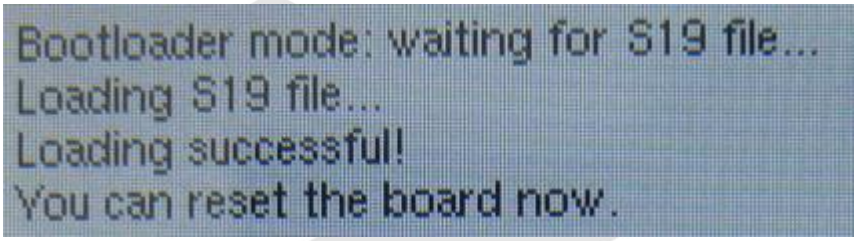

Figure 17: Bootloader has flashed S19 file

Additionally the MSD (Mass storage device will show 'SUCCESS.TXT'):

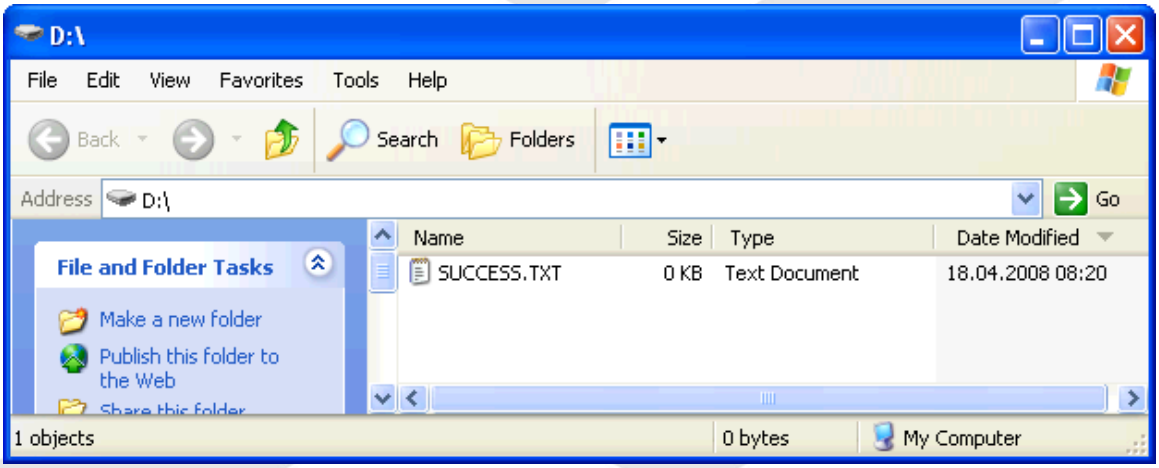

Figure 18: Bootloader successful MSD message file

Now you can reset the board (pressing JMRST), and this will launch your new application.

2.4 Debugging your application with the bootloader

In order to debug your application, you will need a BDM cable (e.g. P&E USB Multilink) connected to the JMBDM connector.

As with a bootloader there are two binaries (the bootloader plus your application) running on the target, the debugger needs to be aware of it.

In order to have complete visibility, do the following:

- Create a copy of your bootloader CodeWarrior project
- Build your bootloader. Download and flash it to the target with the BDM cable.
- Make copies of your bootloader binaries (e.g. name it JM128_Bootloader.abs, JM128_Bootloader.xMAP and JM128_Bootloader.S19) for later use
- Switch to your application project.
- Build your custom application. Best if you rename your application e.g. Application.abs, Application.xMAP and Application.S19 for later reuse. Reset your TWR-LCD board and load the application S19 file of it using the bootloader

Now we are going to connect to the target. Launch the debugger for your application:

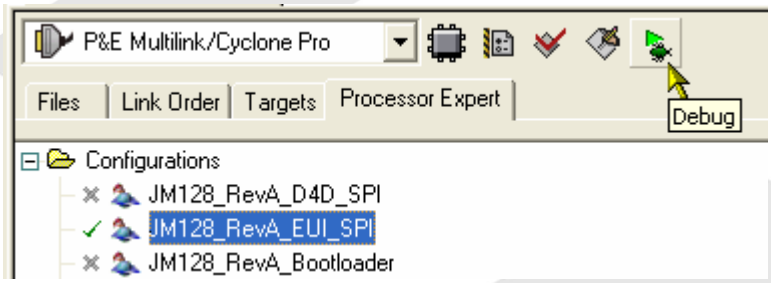

Figure 19: Launching debugger for application

Instead of downloading, we are only to hotsync to the target, using the HotSync button:

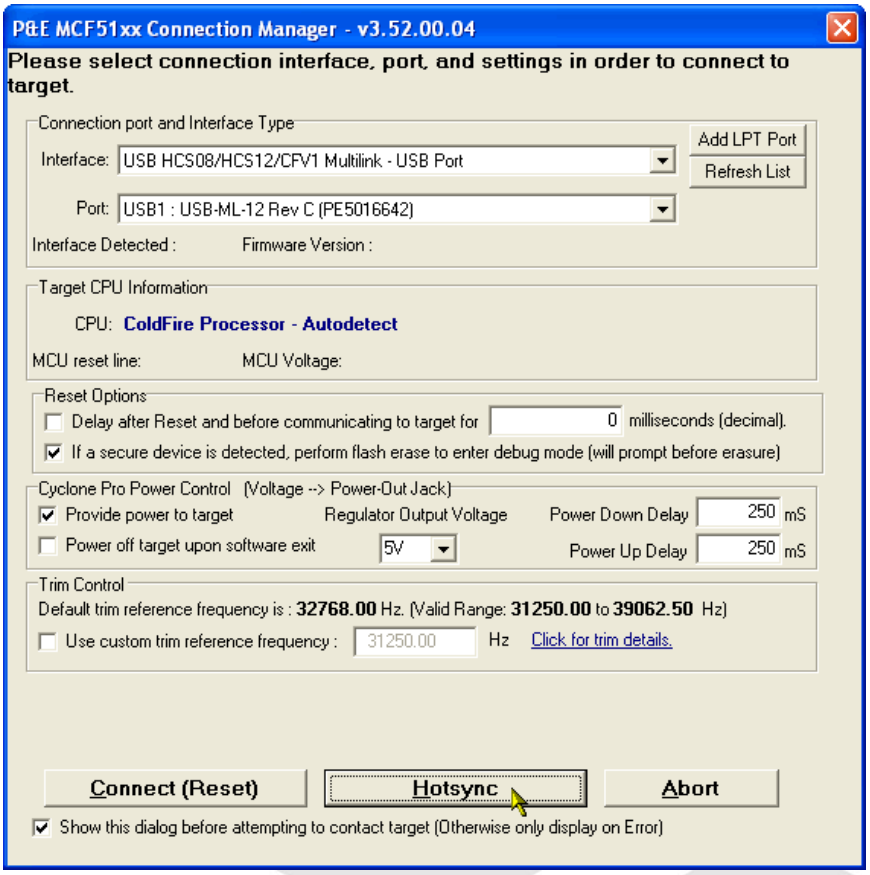

Figure 20: HotSync to the target

Reset the target:

Figure 21: Reset the target

Now we need to load the debug information (or symbolics) for the two binaries: the bootloader and your application: for this we made copies of the .abs file in the previous steps.

To load the debug information for each: Use the Load command in the debugger:

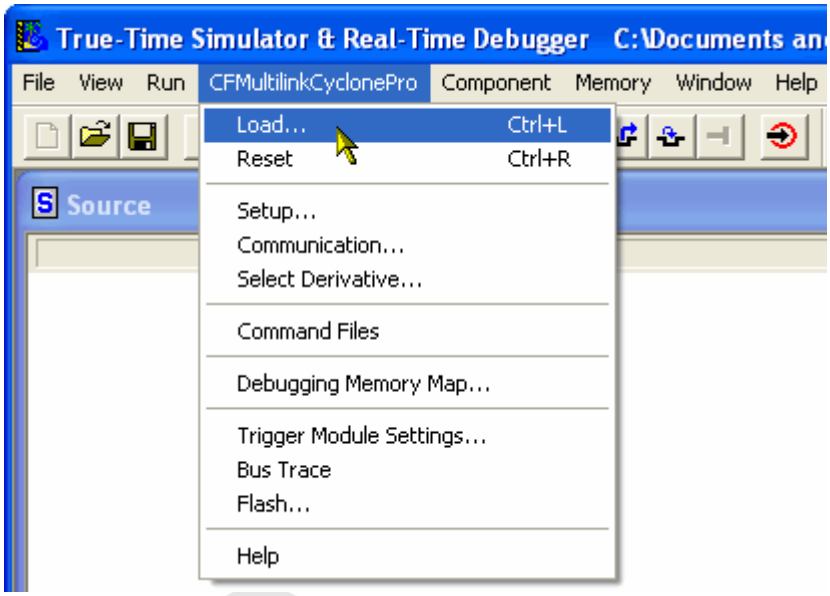

Figure 22: Loading in the debugger

Browse to the binaries (in the bin folder), select the bootloader .abs file and press the 'Load Symbols' button.

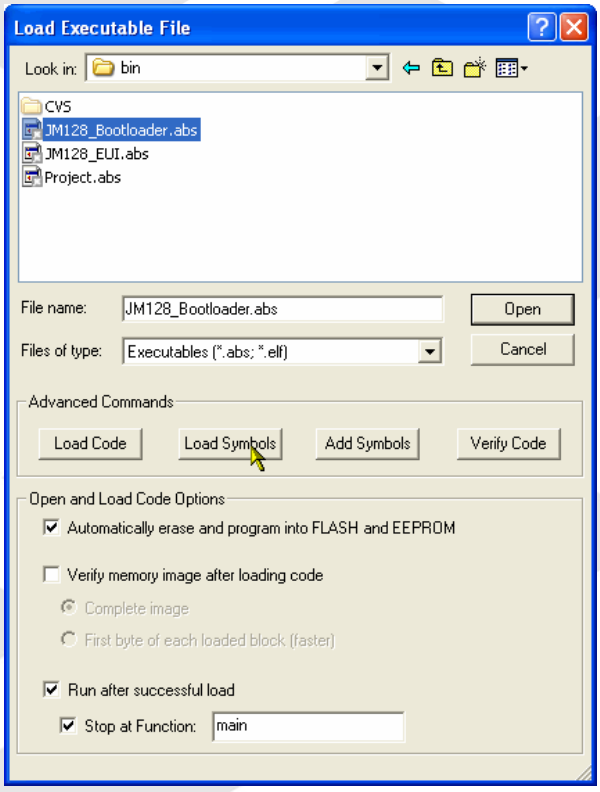

Figure 23: Loading bootloader symbols

With the same dialog, add the application symbols using the ‘Add Symbols’ of your application:

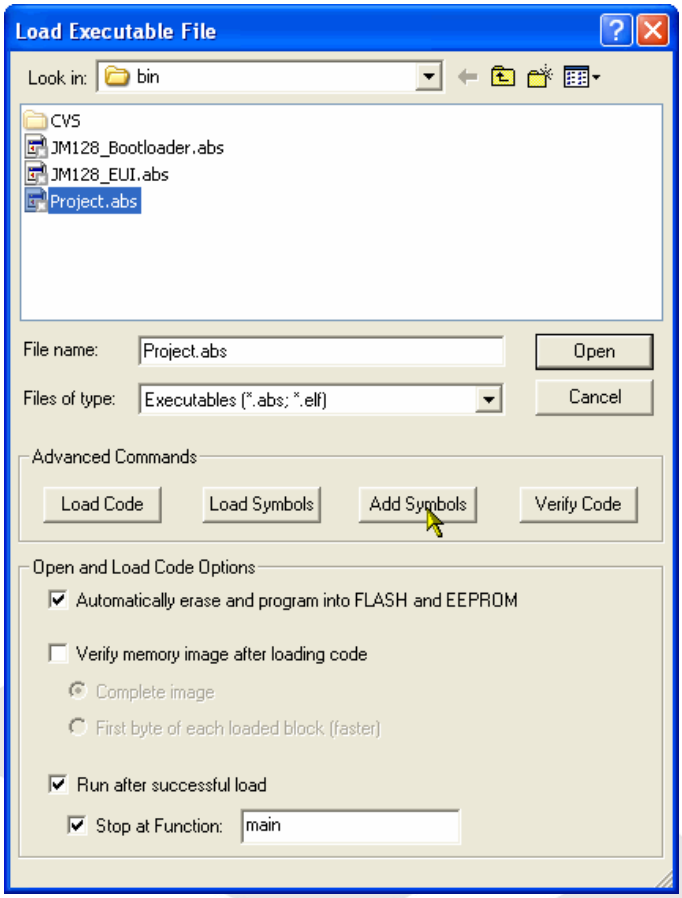

Figure 24: Adding application symbols

2.5

Memory Map for Bootloader and Application

It is important to know the memory mapping both for the bootloader and the application on top of the bootloader.

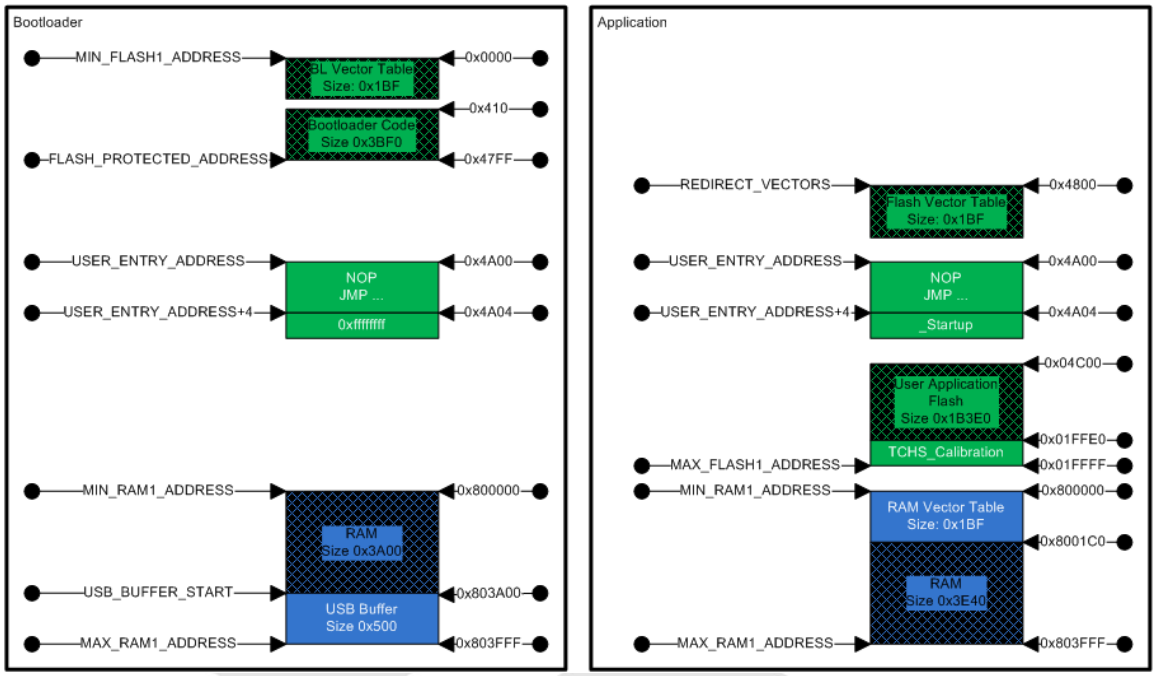

Figure 25: Bootloader and memory map

The application flash needs to be above FLASH_PROTECTED_ADDRESS.

2.5.1

Bootloader Build Options

The easiest way is to configure the bootloader memory configuration in Build Options:

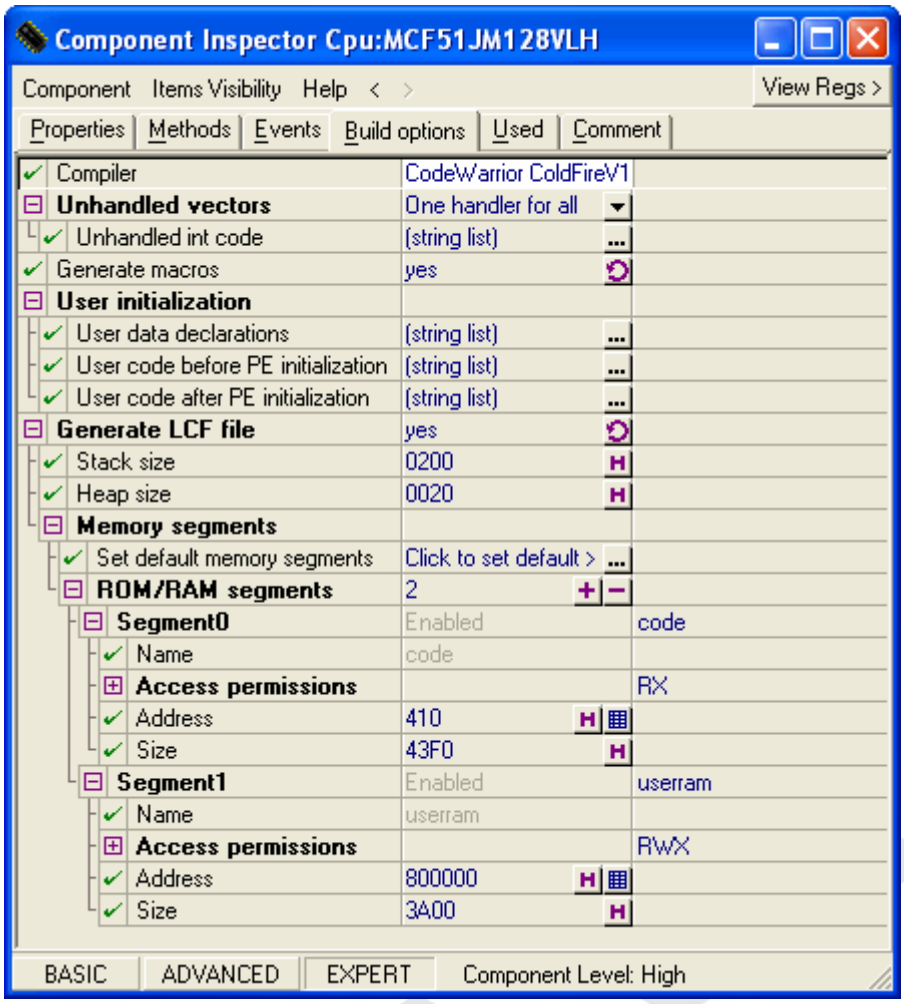

Figure 26: Bootloader Build Options

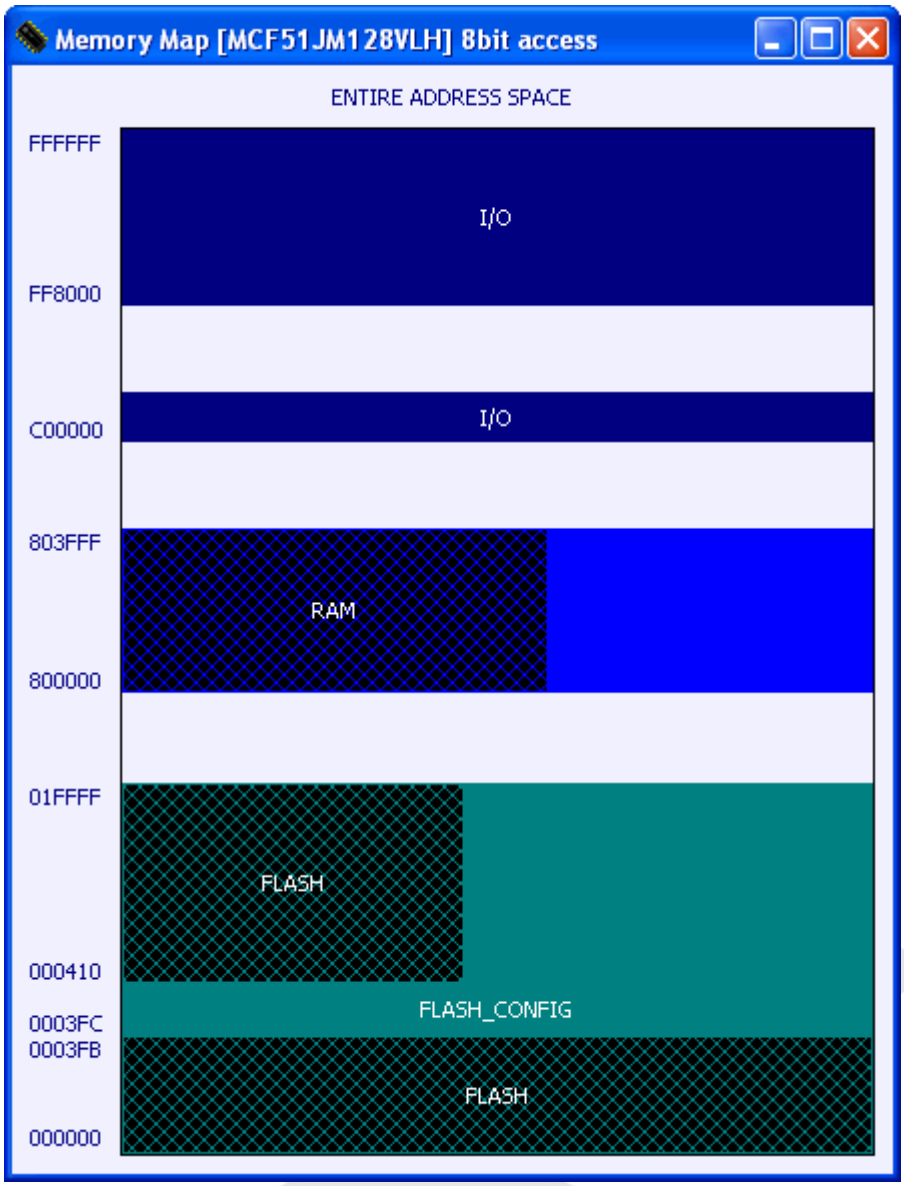

Figure 27: Bootloader memory map

Additionally, the bootloader needs to do an early check if the boot loader or application mode shall be entered. This needs to be done as part of the _Startup(), just at the beginning of _initialize_hardware().

As such, an include to “Bootloader.h” has been added to the ‘User data declarations’:

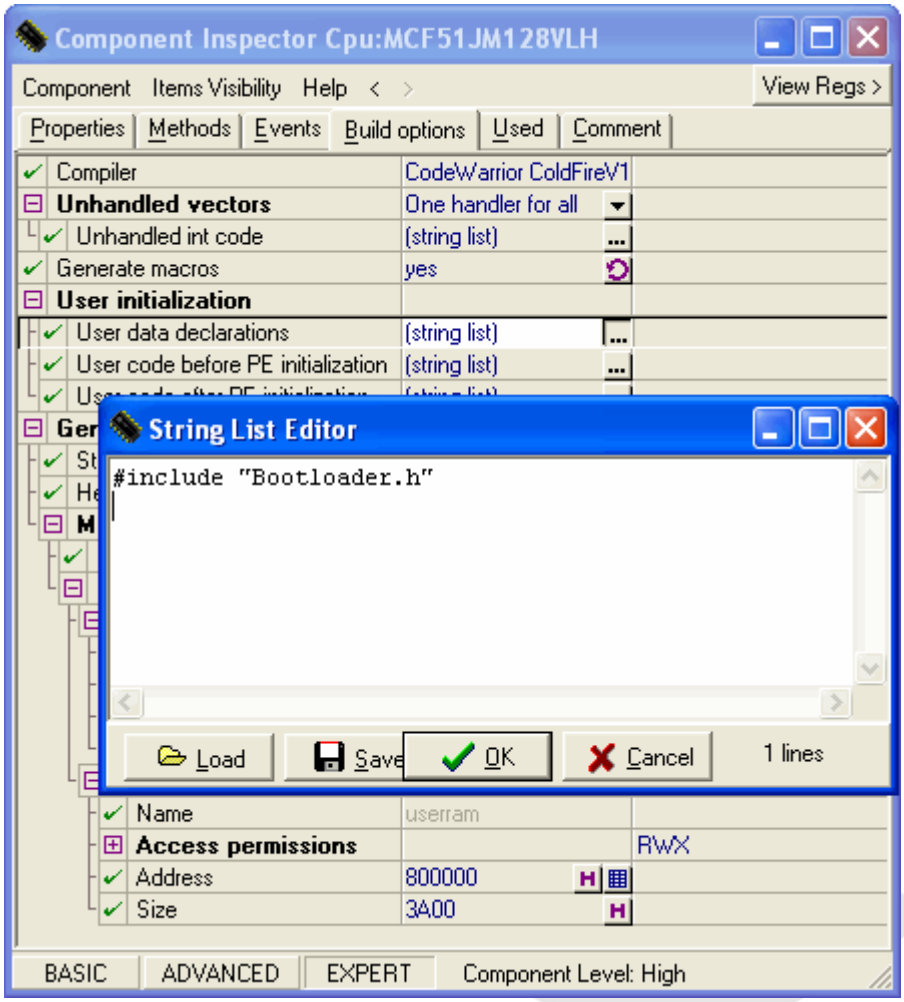

Figure 28: Bootloader User Data Declarations

And in order to call the Bootloader function which performs the check on the BTLD switch, a call to BL_CheckForUserApp() has been added to 'User ode before PE initialization'.

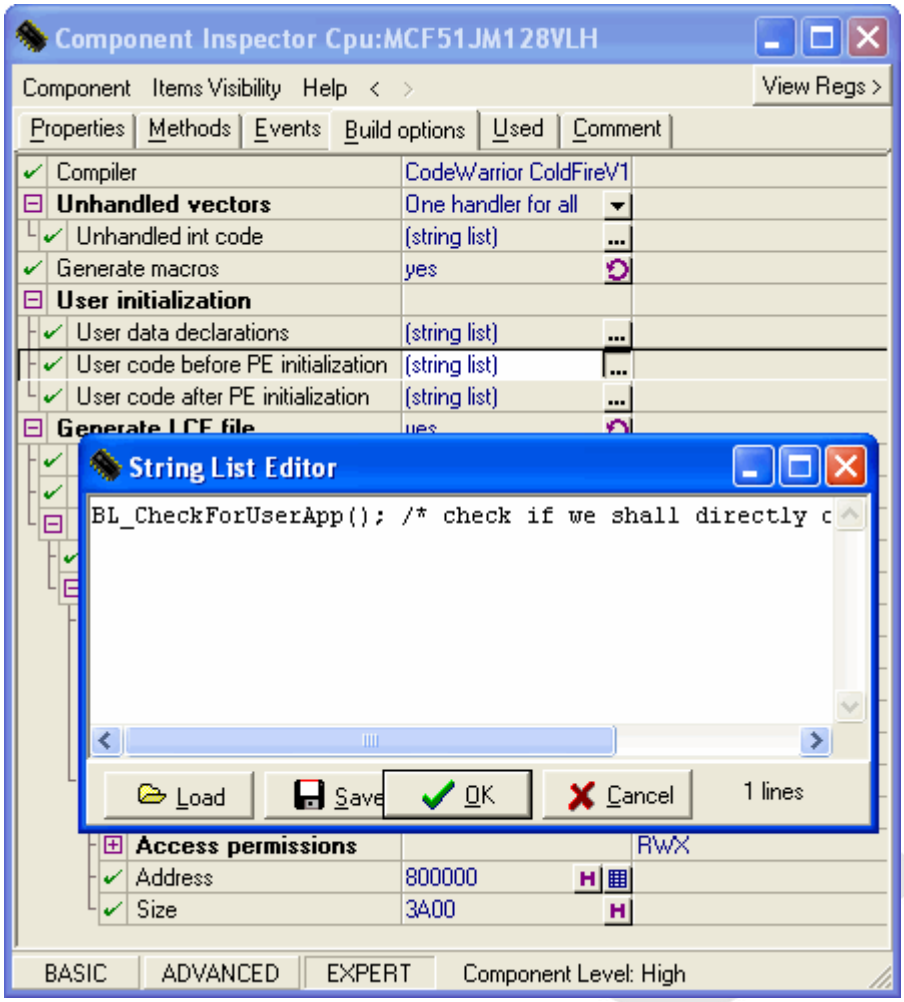

Figure 29: Bootloader Use code before PE initialization

2.5.2

Application Build Options

In a similar way you can configure the memory map for the user application:

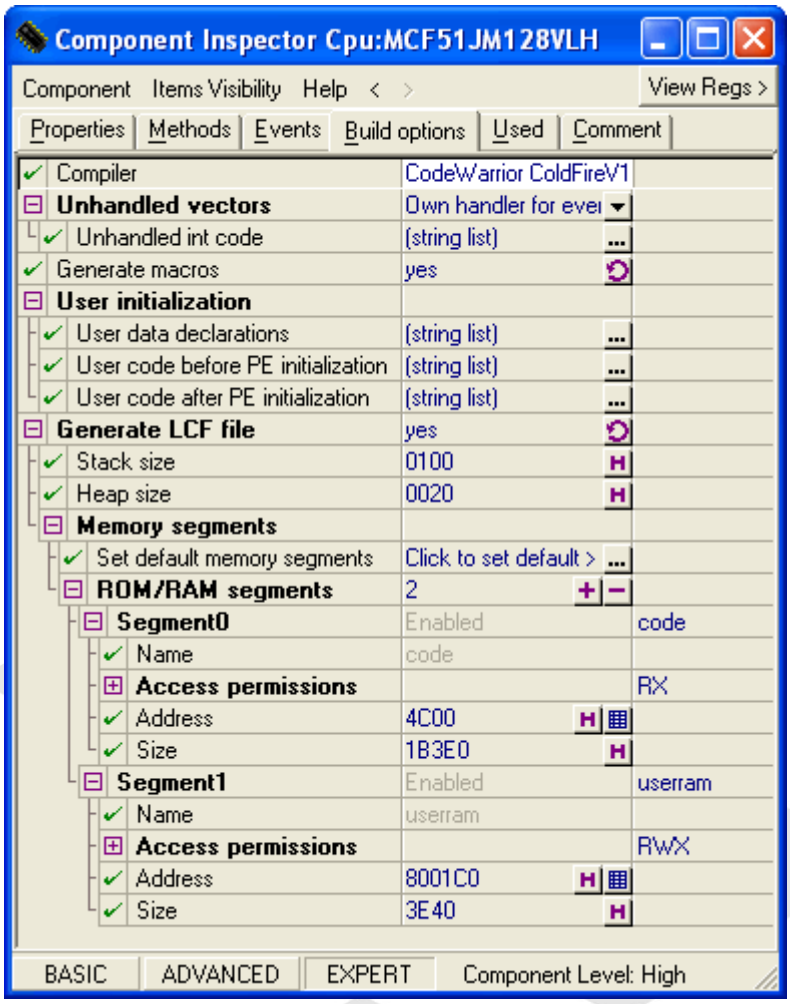

Figure 30: Application Build Options

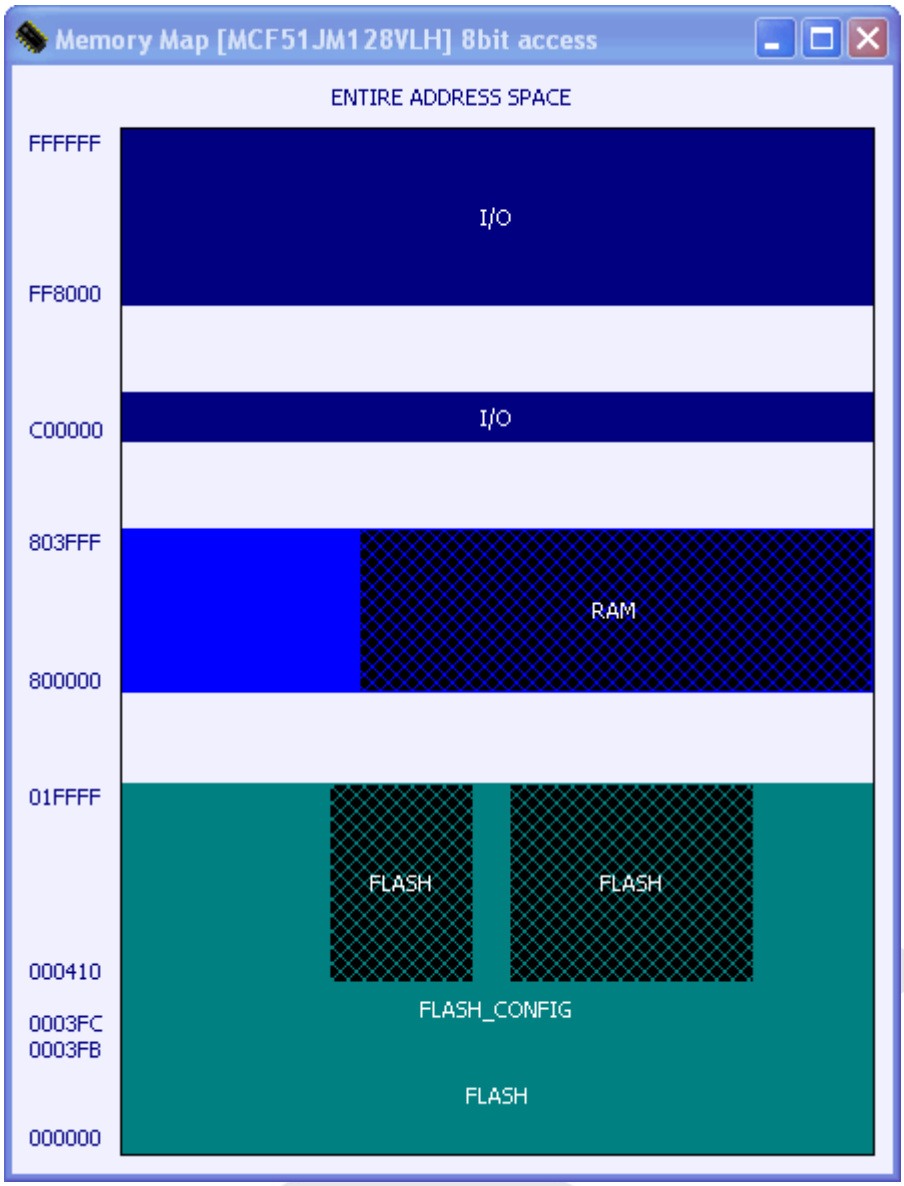

Figure 31: Application memory map

The application needs to use its own vector table. For this you need to allocate the vector table at REDIRECT_VECTORS address:

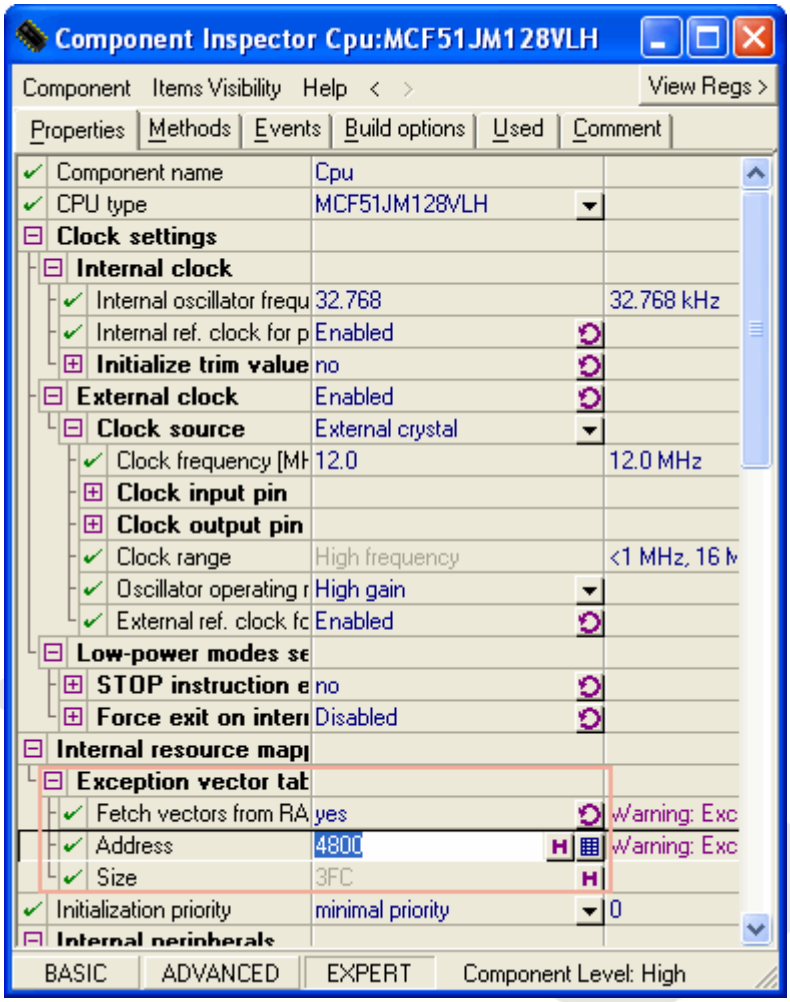

Figure 32: Application vector table settings

Then the application needs to copy the FLASH vector table to RAM:

```
#if PL_HAS_BOOTLOADER
void BL_RedirectUserVectors(void) {
  /* !! This section needs to be here to redirect interrupt vectors !! */
  dword *pdst,*psrc;
  word i;

  asm (move.l  #BL_APP_RAM_VECTOR_ADDRESS,d0);
  asm (movec   d0,vbr);

  pdst=(dword*)BL_APP_RAM_VECTOR_ADDRESS; /* copy to RAM */
  psrc=(dword*)REDIRECT_VECTORS; /* The vector table has been placed here */

  for (i=0;i<111;i++,pdst++,psrc++) {
    *pdst=*psrc;
  }
}
#endif /* PL_HAS_BOOTLOADER */
```

Figure 33: Application vector table copy and redirection

Lab 3: TWR-LCD Freescale Embedded GUI Demo

In order to build the Freescale Embedded GUI Demo, load the CodeWarrior project with the Embedded Components (see Lab 2/Bootloader).
The demos in this and the next lab sessions are using many embedded components. The following block diagram gives an overview of the system:

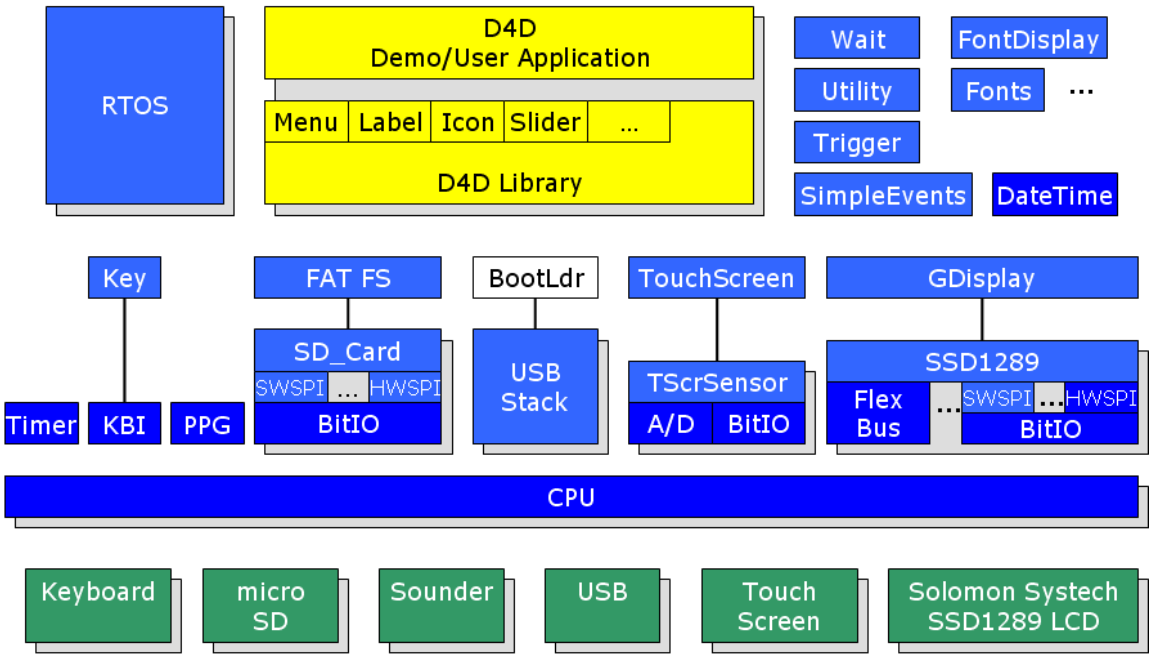

Figure 34: System Block Diagram

3.1 Selecting Configuration

Make sure your current CPU is set correctly:

Figure 35: Change MCU/Connection

'Change MCU/Connections...' opens the following dialog:

Verify that your target CPU is the MCF51JM128:

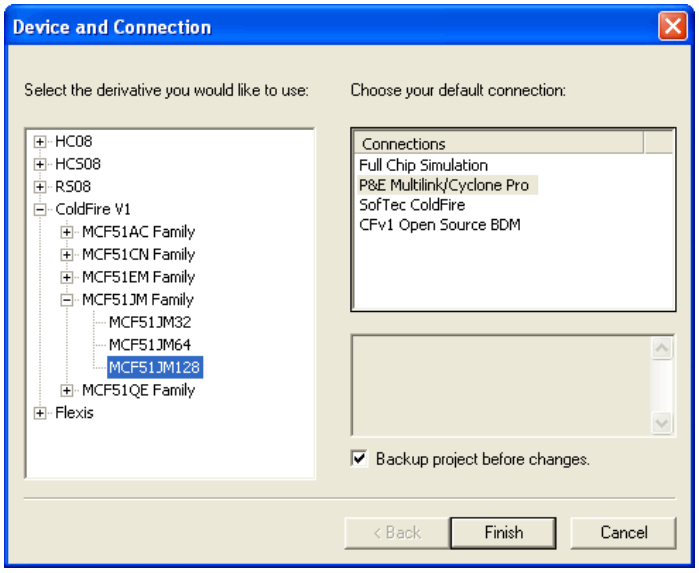

Figure 36: Change MCU/Connection

Verify that your current configuration is the 'JM128_RevA_BL_EGUI_SPI' one. If not, select it as the active configuration:

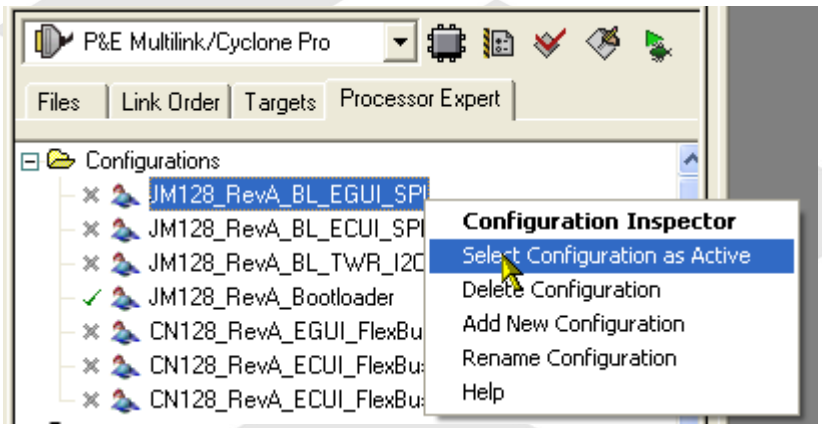

Figure 37: Selecting the Freescale Embedded GUI Demo

As the Configuration name indicates, the demo is for the JM128 and the RevA TWR-LCD board. The 'BL' indicates that the application needs to be loaded by the bootloader. 'EGUI' indicates that the Freescale Embedded GUI is used, and 'SPI' indicates that the serial SPI communication protocol is used to communicate with the display.

3.2 Inspecting CPU

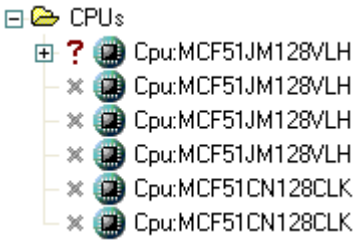

Figure 38: CPU components

Each configuration has a CPU component associated with it. The CPU component defines, which CPU has to be used, and configures things like stack usage memory map.
As we are using a special memory map and exception vector table for the bootloader, this is indicated with a '?' mark sign and a message from the Processor Expert system:

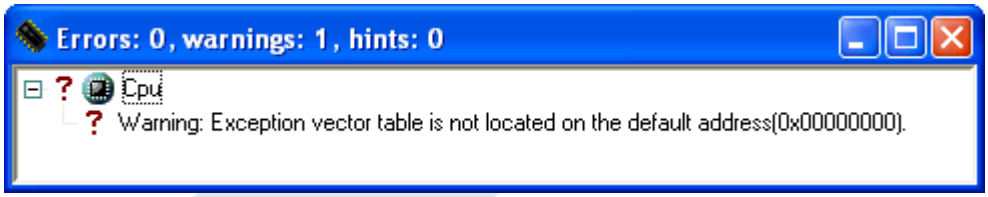

Figure 39: Information about vector relocation

This means that this application will only run successfully with the bootloader, loaded by the bootloader.
You can verify the settings in the CPU Inspector:

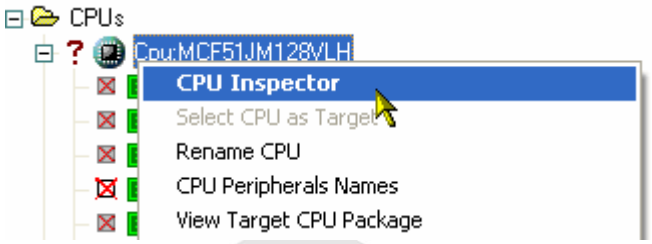

Figure 40: Launching CPU inspector

In the ‘Properties’ of the CPU you can see that the vectors are allocated at address 0x4800 and will be fetched from RAM. Make sure that you have the viewer in ‘EXPERT’ mode to see all details.

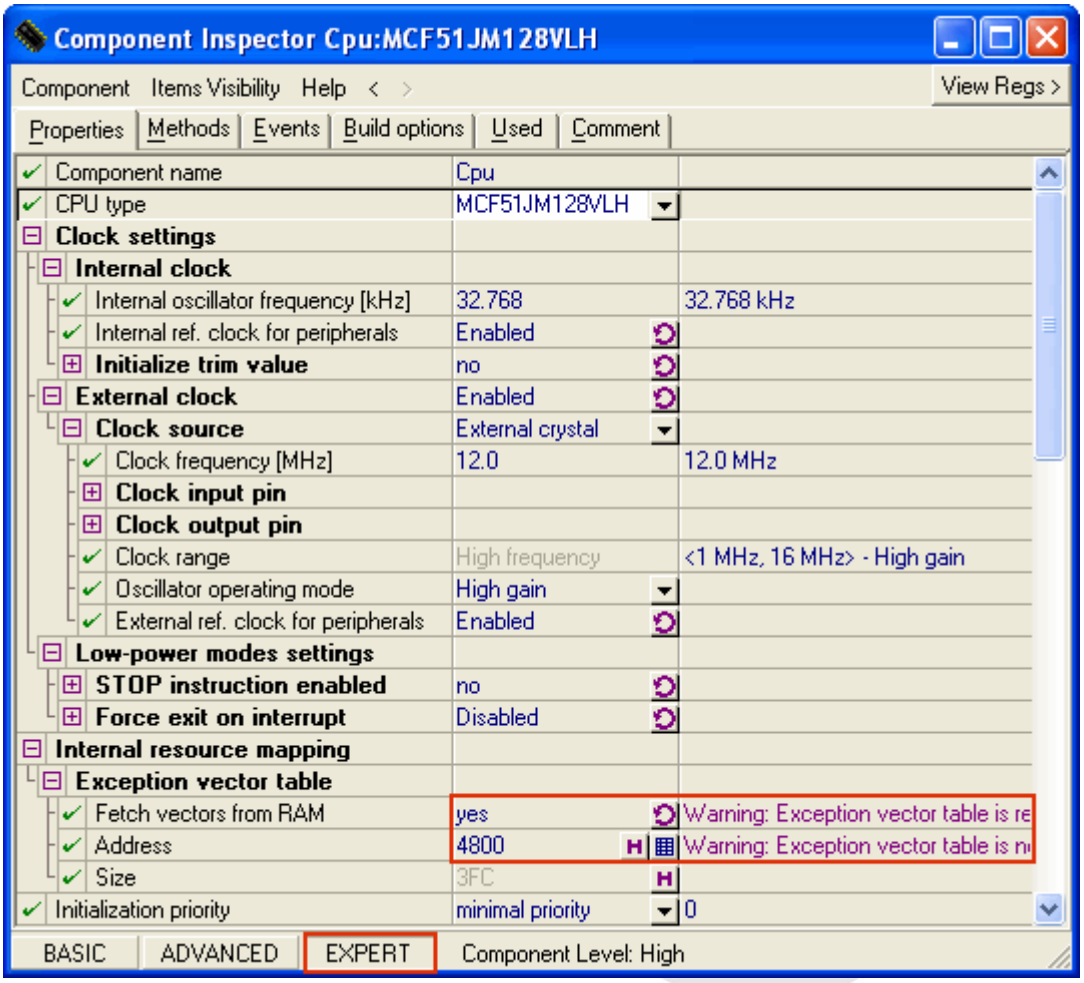

Figure 41: Application Vector Redirection table

3.3

Inspecting Low Level Display Driver Component

The Processor Expert tab shows as well a list of components used. The components have a checkmark if they are enabled for a given configuration.

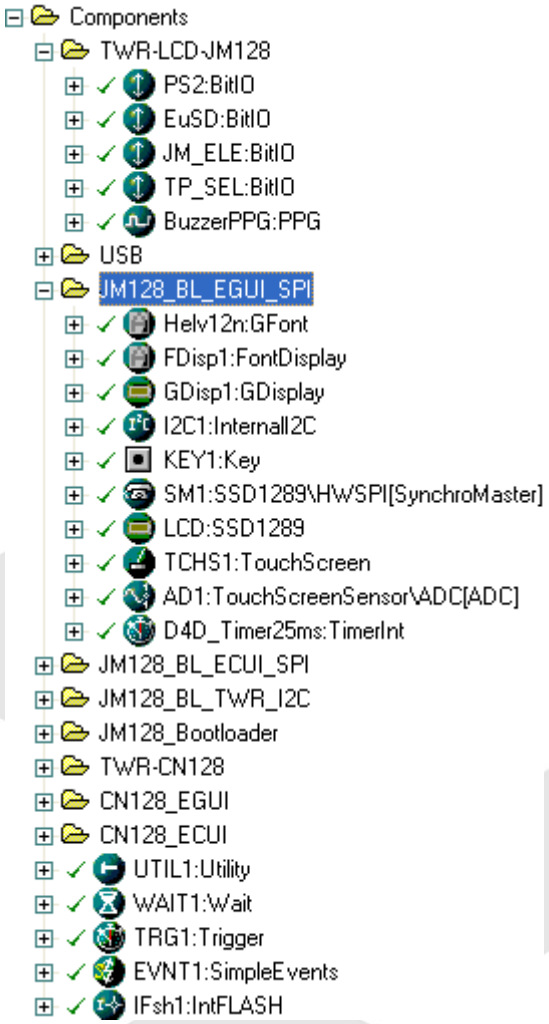

Figure 42: Embedded Components

The subfolder 'TWR-LCD-JM128' contains common components used for the TWR-LCD board. The 'JM128_BL_EGUI_SPI' contains components configured for the Freescale Embedded GUI, and at the end there are components shared for all configurations.

You can hover over a component to show details about the component. If you unfold it, you get a list of inherited components (like BitIO) and Methods Methods provided by that component.

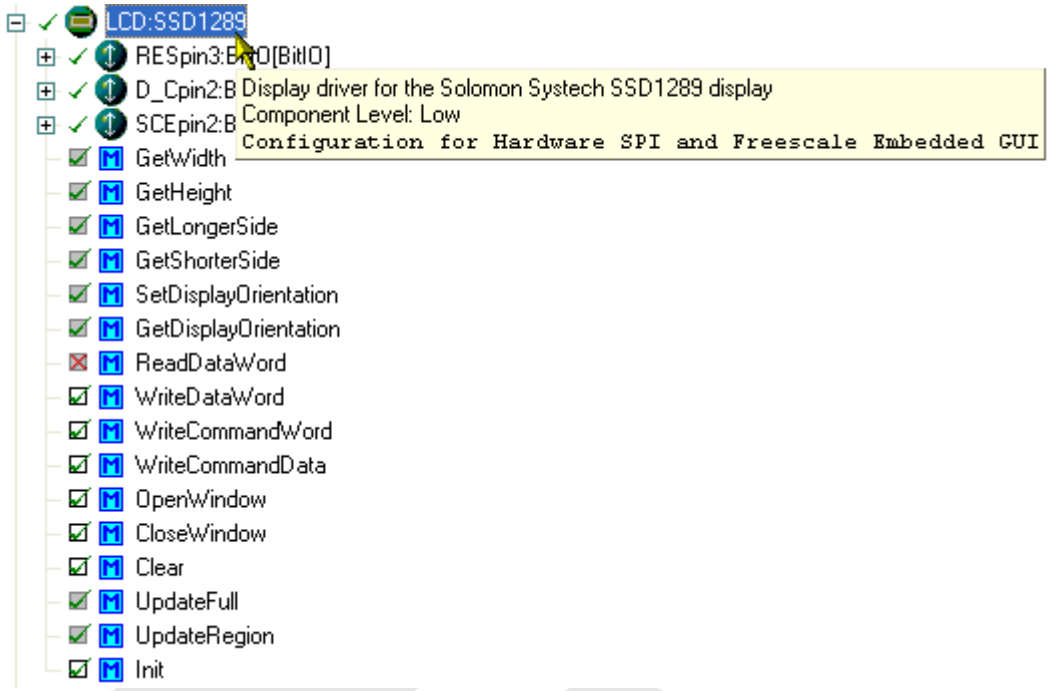

Figure 43: Embedded Component Details

To inspect the settings of a component, double click on it or use the 'Component Inspector' menu:

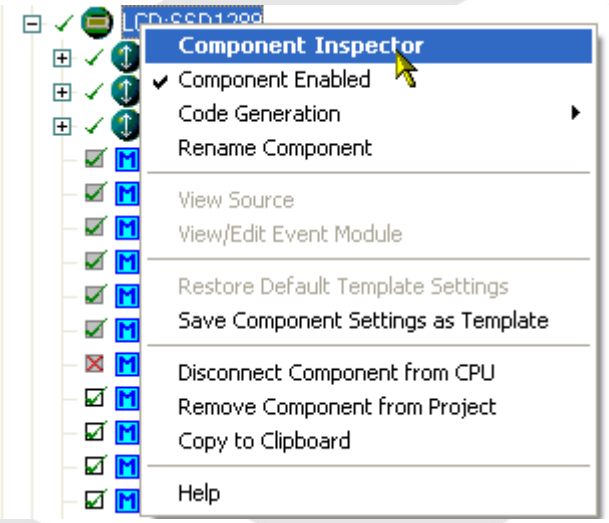

Figure 44: Embedded Component Inspector Menu

If you inspect the RES (Reset pin) component of the SSD1289 display driver, you see it is connected to the PTE3 pin.

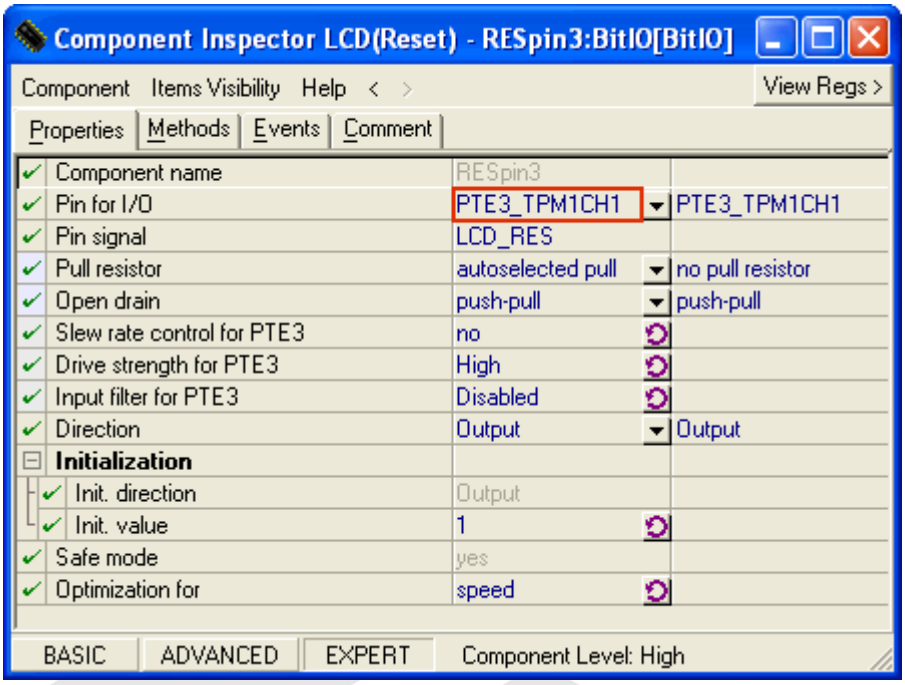

Figure 45: Display Reset Pin Properties

3.4 Building the Project S-Record File

To build the project, press the 'Make' button:

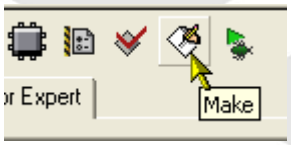

Figure 46: Make Button

This generates the source code and creates the S19 File we want to load with the bootloader.
The S-Record file has extension .S19 and will be generated in the 'bin' folder of your example project.

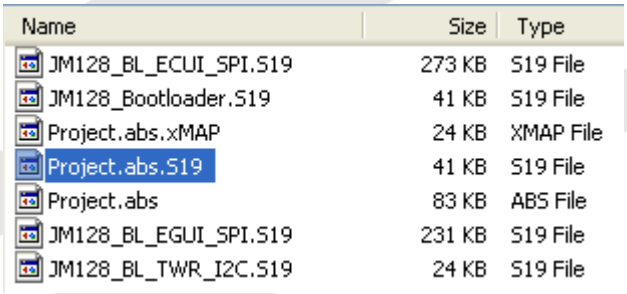

Figure 47: Generated S19 File

Enter now the bootloader mode on your board as shown in the previous Lab and load your application to the target to run it.

Lab 4: TWR-LCD Processor Expert Embedded UI Demo

Exactly as in the previous demo, you can build and load the demo using Processor Expert UI components. This demo is using the same low level drivers, but has implemented a different demo completely written with Processor Expert Embedded Components.
In order to switch to this demo, select the 'JM128_RevA_BL_ECUI_SPI' Configuration:

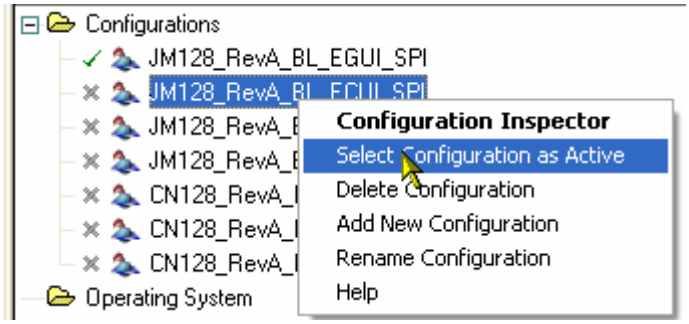

Figure 48: Selecting the Embedded Component UI Demo

Then you can build and load the demo in the same way as in the previous demo.

4.1 Configuring the demo amount

In order to configure the demos and the amount of demos, the file 'platform.h' is used:

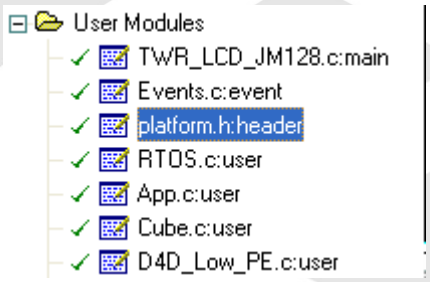

Figure 49: platform.h header file

In order to enable a demo, place a '1' in front of the #define condition. To disable a demo, place a '0' in front of it. Then recompile your project.

```
/* demo configuration for Embedded Component UI */
#define PL_HAS_TOUCHSCREEN_DEMO    (1 && PL_USE_UI_EUI && PL_HAS_HW_TOUCHSCREEN)  /* if we inclu
#define PL_HAS_CALIBRATION_DEMO    (1 && PL_USE_UI_EUI && PL_HAS_HW_TOUCHSCREEN)  /* if we includ
#define PL_HAS_CUBE_DEMO           (1 && PL_USE_UI_EUI)  /* if we include the 3D rotating cube de
#define PL_HAS_TETRIS_DEMO         (0 && PL_USE_UI_EUI)  /* if we inclued the tetris game demo */
#define PL_HAS_FONT_DEMO           (0 && PL_USE_UI_EUI && !PL_TETRIS_USES_BMP)  /* if we have t
#define PL_HAS_CALENDAR_DEMO       (1 && PL_USE_UI_EUI)  /* if we have the calendar demo */
#define PL_HAS_TASKLIST            (1 && PL_USE_UI_EUI && PL_USE_RTOS) /* if we show a list of R
#define PL_HAS_ACCEL_DEMO          (1 && PL_USE_UI_EUI && PL_HAS_HW_ACCELEROMETER) /* if we demo
#define PL_HAS_SD_DEMO             (1 && PL_USE_UI_EUI && PL_HAS_HW_SD_CARD) /* if we demo the
#define PL_HAS_PNG_DEMO            (1 && PL_HAS_SD_DEMO) /* if we                 PNG fil
```

Figure 50: platform.h demo configuration

Keep in mind that depending on your target you may not have enough RAM and ROM space to run all demos at the same time.

Lab 5: Freescale Embedded GUI with Accelerometer

In this Lab we are going to use the TWR-LCD board together with the TWR-MCF51CN128 board.

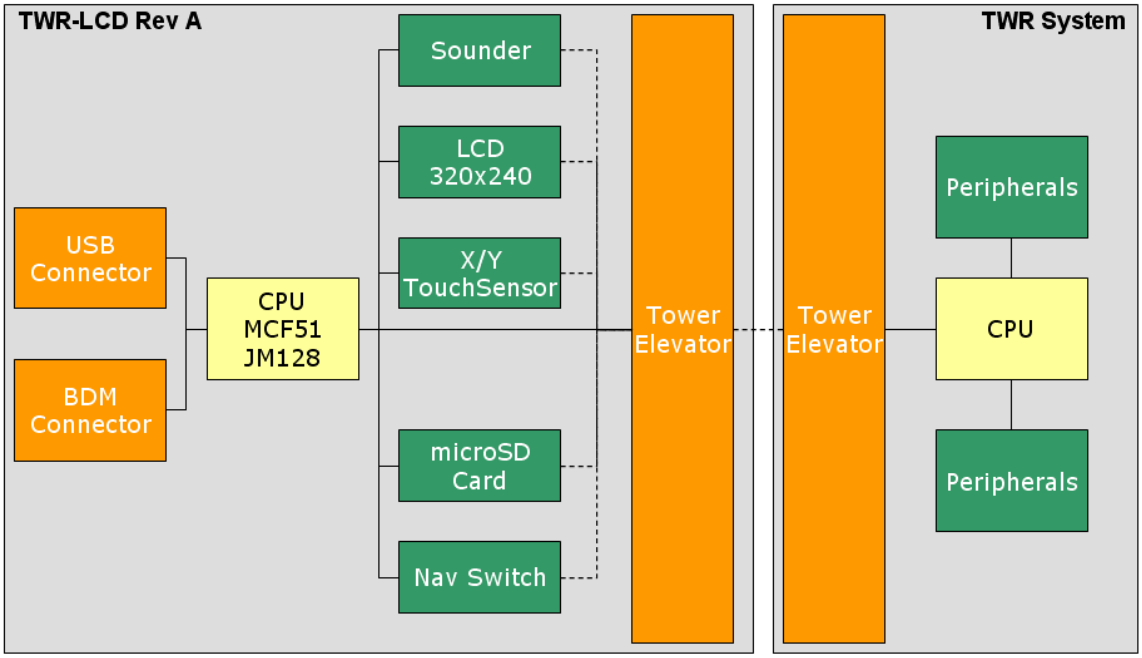

Figure 51: System Configuration with a TWR System

Through the Tower Elevator the TWR CPU has access to most peripherals on the TWR-LCD board. In this lab example we run the same demos as from previous example on the TWR-MCF51CN128. Note that there is no bootloader in this example for the CN128, but the TWR-LCD JM128 is running the bootloader. The reason to have a minimal application (in our case the bootloader) on the TWR-LCD JM128 is the need to tristate some lines and signals to the LCD module in order to have them operating correctly.

Important Note: The MCF51CN128 needs to configure the Reset pin as output pin to drive the LCD reset. As such, if you press the Reset/SW4 switch on the TWR-MCF51CN128 board, this will as well reset the LCD and put it into an initialized state. Same happens if you press the JMRST button on the TWR-LCD board if the display is controlled by the CN128. As you cannot reset the CN128 that way using the reset/SW4 switch, you need to do a power cycle using the Elevator Power On/Off switch.

5.1

Selecting Configuration

Make sure your current CPU is set correctly to the CN128:

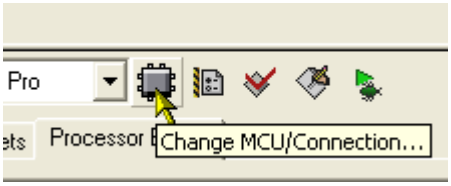

Figure 52: Change MCU/Connection

'Change MCU/Connections...' opens the following dialog:

Verify that your target CPU is the MCF51CN128:

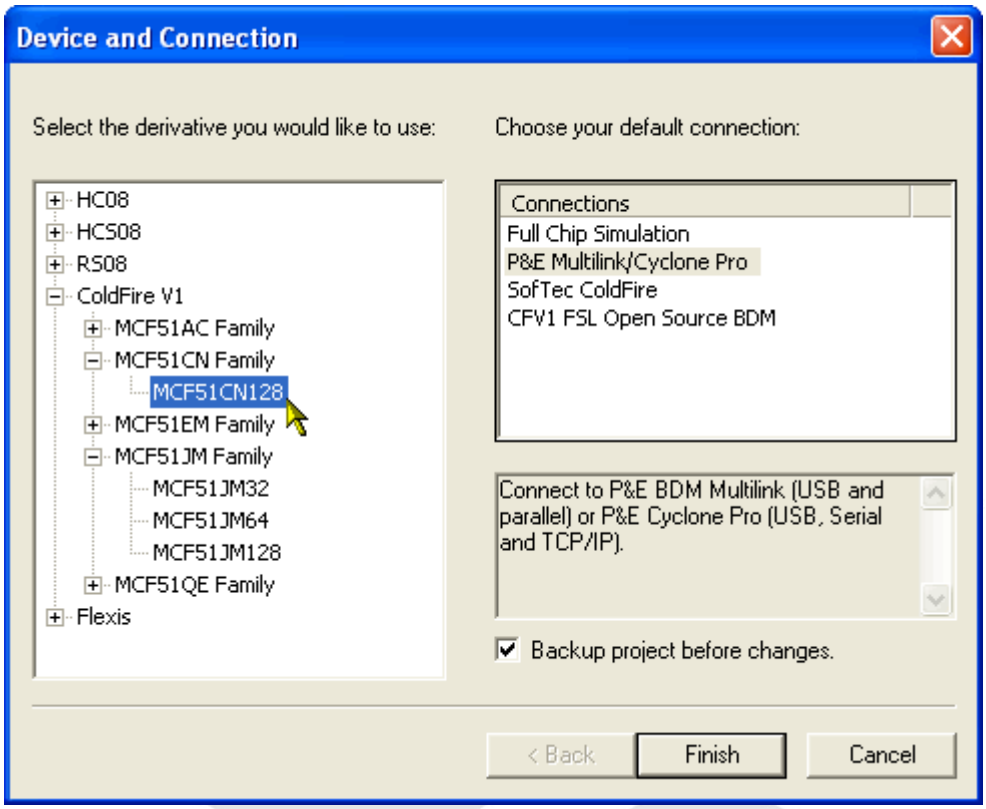

Figure 53: Change MCU/Connection

Verify that your current configuration is the ‘CN128_RevA_EGUI_FlexBus_Accel’ one. If not, select it as active configuration:

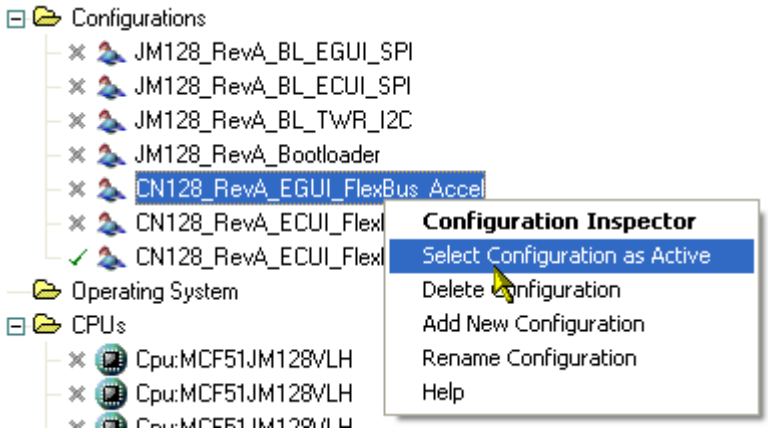

Figure 54: Selecting the Freescale Embedded GUI Demo for CN128

5.2 Installing Hardware Inspecting Jumper Settings

For the demo you need to attach the TWR-LCD to the TWR-ELEVATOR board. Make sure you move all switches of the TWR-LCD SW5 DIP switches to the ON position, as the DIP switch will be hardly accessible after attaching the TWR-LCD board to the TWR-ELEVATOR.
The demo requires that you set up the jumpers both on the TWR-LCD and TWR-CN128 board correctly.

The configuration Inspector gives you a list of required settings.

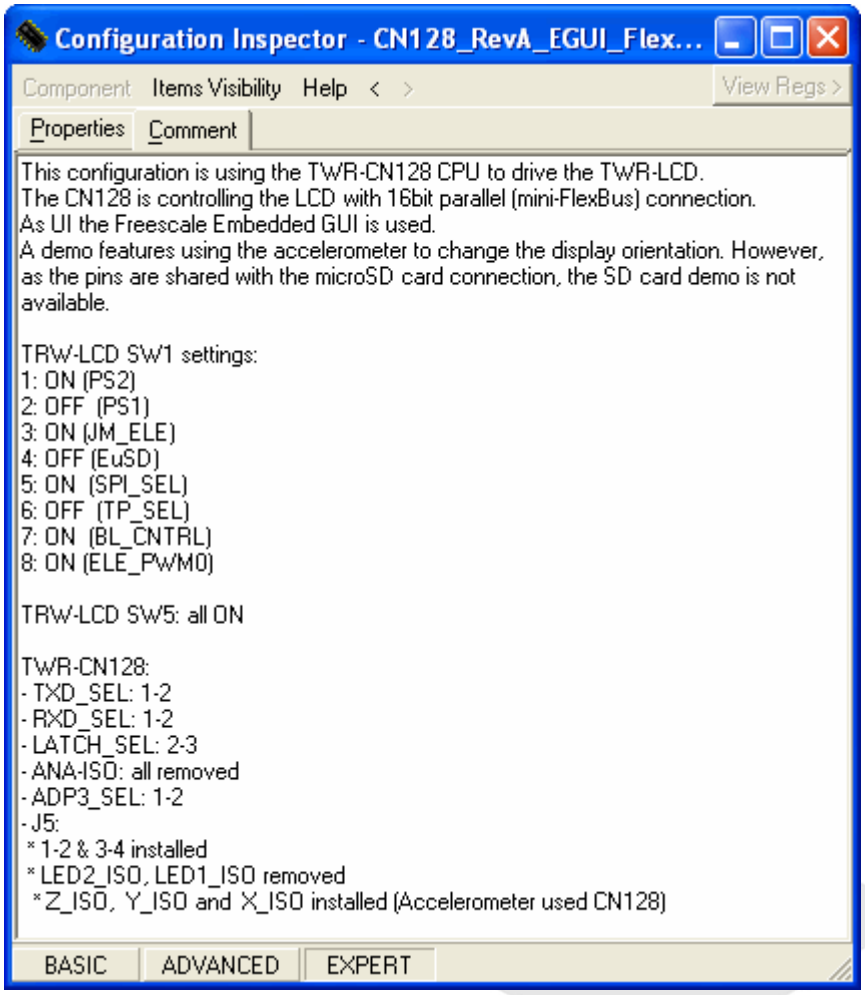

Figure 55: Configuration Inspector Comment section for CN128 Demo

5.3 Power the system

With the TWR-LCD attached to the TWR-ELEVATOR, there are now two mini-USB connectors which can be used:

a) use the TWR-ELEVATOR as the power source for the TWR-LCD and the rest of the TWR-System
b) optionally, the mini-USB connector on the TWR-LCD can also be connected in case you need access to the USB bus of the JM128 (e.g. to flash an application with the bootloader)

5.4

Inspecting mini-FlexBus Settings

If you inspect the CPU properties, then you can see how the mini-Flexbus is configured to access the TWR-LCD module:

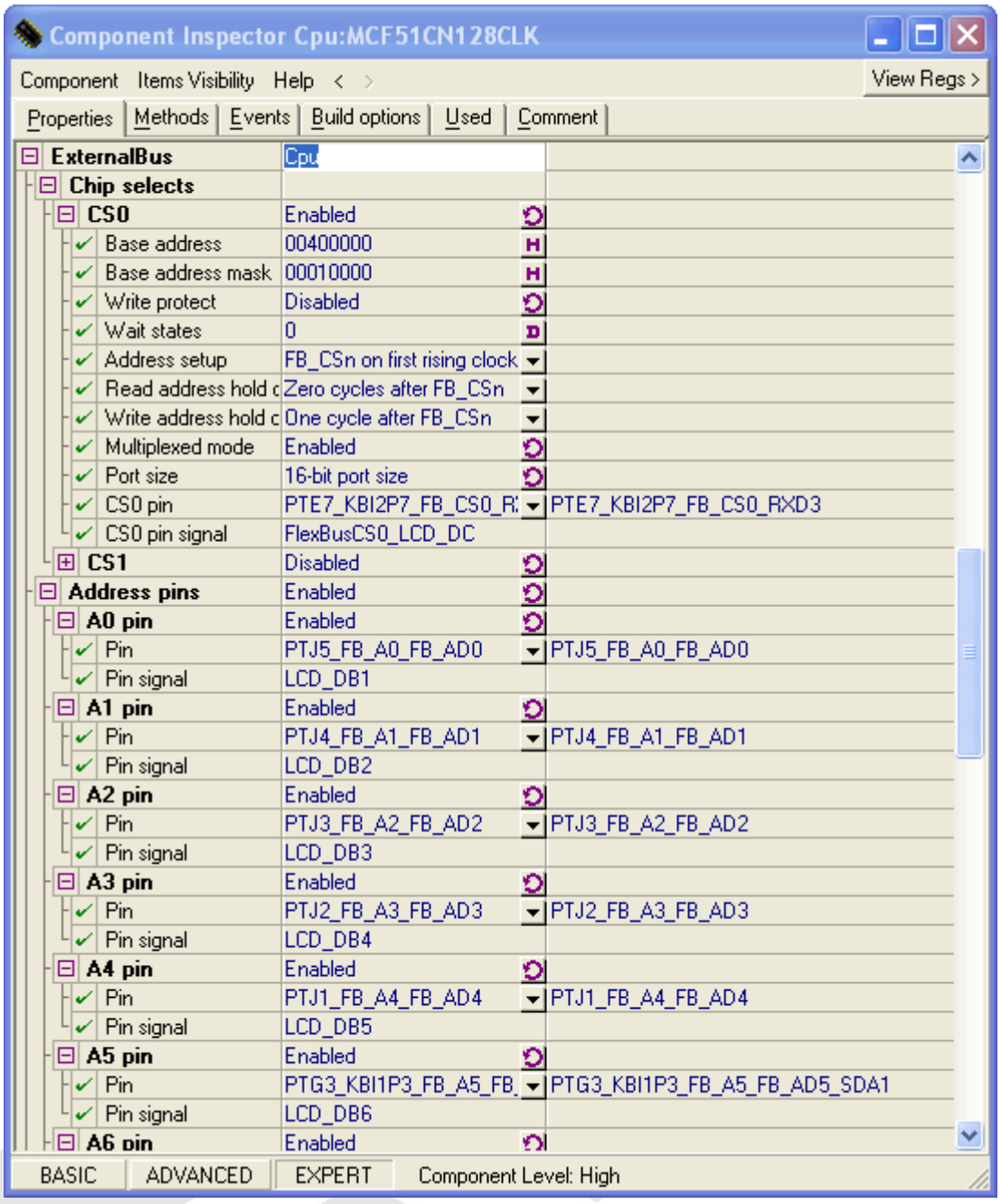

Figure 56: mini-Flexbus configuration in CN128 CPU

Inspect the display driver settings for the low level LCD driver: The components for the Freescale Embedded GUI drivers are located in the 'TWR-CN128' folder:

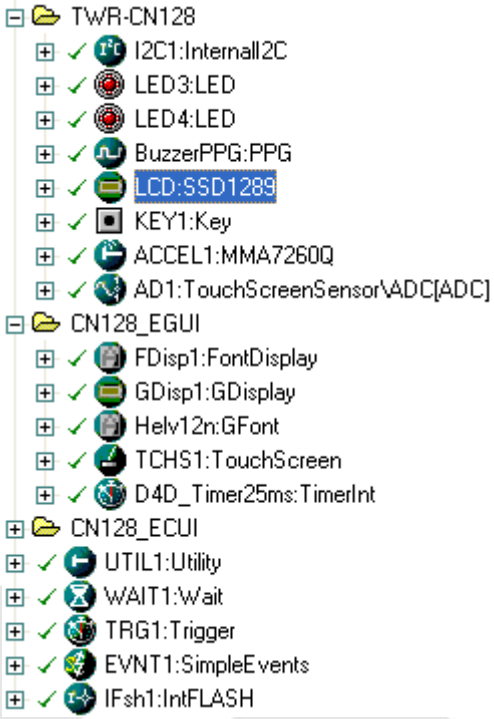

Figure 57: CN128 Freescale Embedded GUI low level drivers

In the SSD1289 driver you can see how the driver is using the parallel communication mode to the display using the ColdFire CN128 mini-Flexbus.

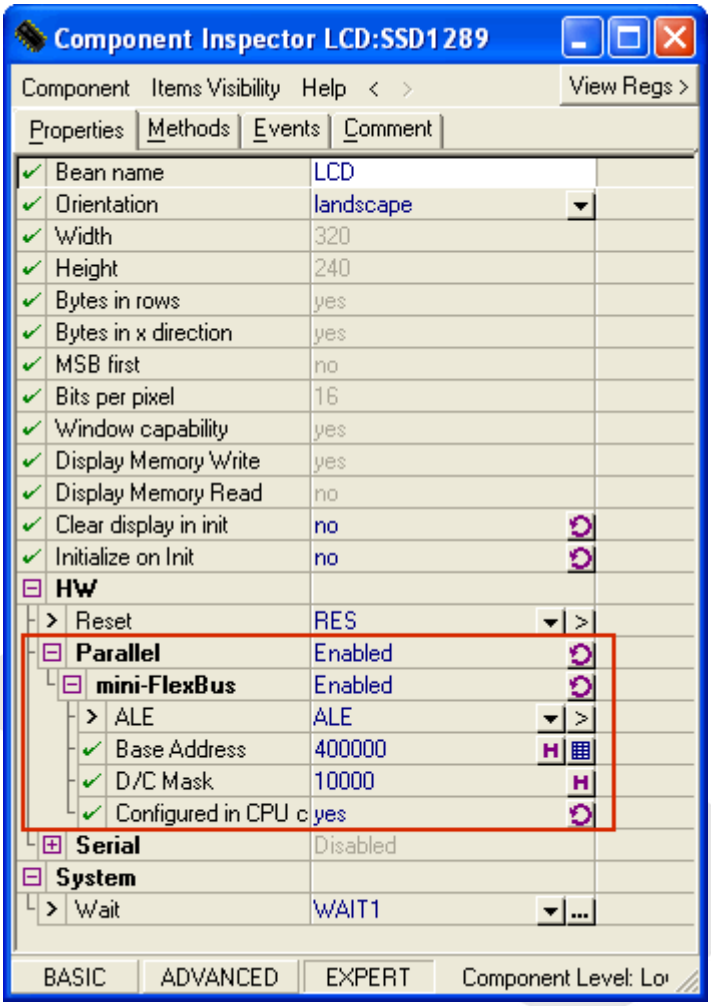

Figure 58: CN128 Freescale Embedded GUI mini-FlexBus settings

5.5 Building and downloading the demo

To build the project, press the 'Make' button:

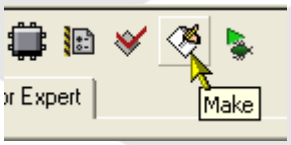

Figure 59: Make Button

This builds the project.
Make sure you have the bootloader present on the TWR-LCD JM128.

Press the debug button to download the application to the target:

Figure 60: Debug Button

In the debugger, start the application with F5 or the Start button.

Figure 61: Start the application in the debugger

This let you use the Freescale Embedded GUI demo from the CN128 CPU.

5.6 Using the switches on the TWR-MCF51CN128

The demo is using the switches on the TWR-MCF51CN128 board to navigate through the menus. You can press SW2 and SW3 on the TWR-MCF51CN128 to navigate back and forward. Pressing SW3 for more than 500ms uses the key as 'enter' key.

5.7 Using the navigation switch with the TWR-MCF51CN128

The 5-way navigation switch on the TWR-LCD board is not directly accessible through the TWR-ELEVATOR to the MCF51CN128. We are using the JM128 on the TWR-LCD board to send I2C messages to the MCF51CN128.

For this, you load an application to the TWR-LCD JM128 which captures the navigation switch interrupts and sends the events over I2C to the CN128.

You can select/build this I2C application like the previous lab examples.

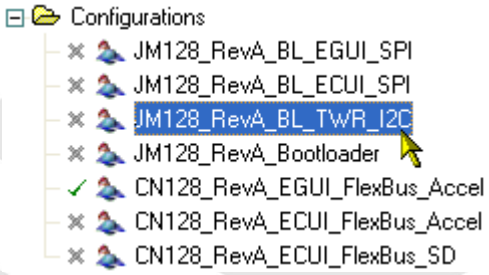

Figure 62: JM128 I2C Application

There is both an I2C component on the JM128 and the CN128. You can identify the I2C component in each of the configurations.

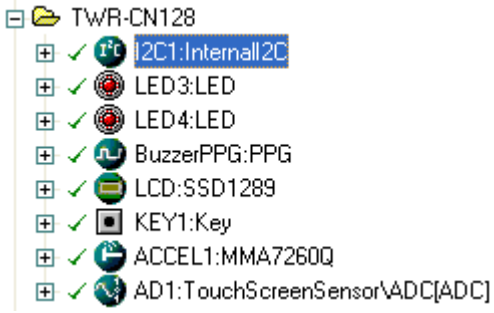

Figure 63: I2C component on the CN128 side

You can either build the JM128_RevA_BL_TWR_I2C configuration, or simply use the provided .S19 file:

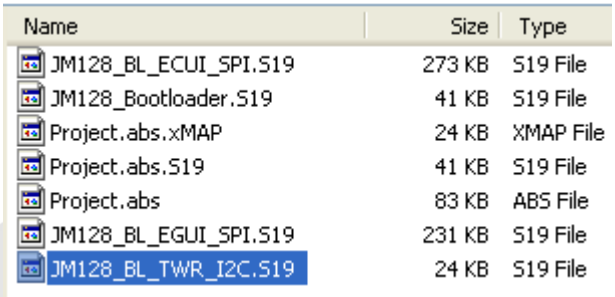

Figure 64: I2C application to send navigation switch messages

You need to load this .S19 file using the bootloader: for this plug in an additional USB cable to the TWR-LCD and press 'JMRST' and 'BTLD' on the TWR-LCD. This will launch the bootloader. Note that the bootloader will detect from the SW1 switch settings that the TWR CPU is controlling the LCD, so you will not see messages on the LCD.
Copy the above .S19 file to the bootloader device to flash the application.
Then power cycle the TWR-ELEVATOR (on/off switch) to reset the whole system properly. Now the same demo as before appears on the LCD.
Now you can use the navigation switch SW2 on the TWR-LCD to navigate through the demo as well.

5.8

Using the TWR-MCF51CN128 Accelerometer Sensor

The demo is using as well the accelerometer on the TWR-CN128 board. Check your jumper settings as specified in the configuration. Running the graph demo visualizes the accelerometer values

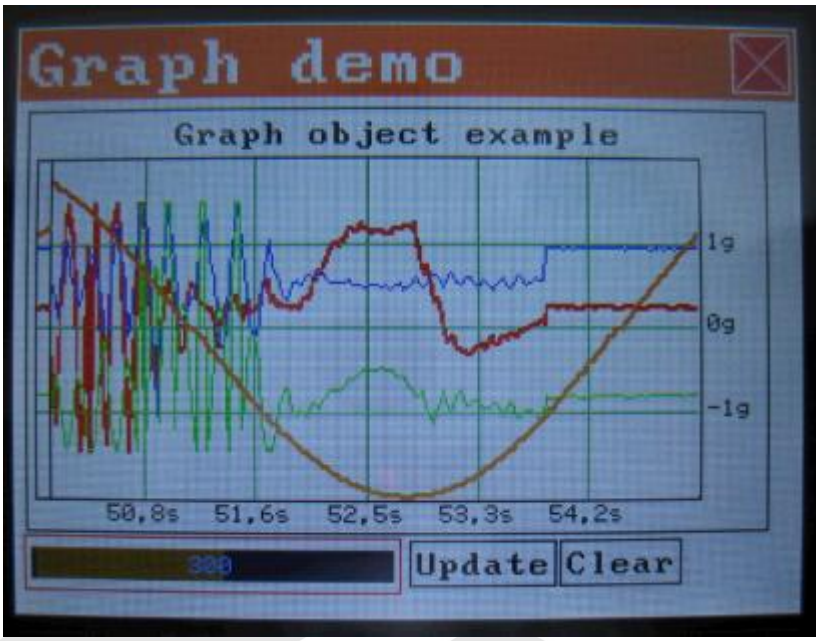

Figure 65: Graph Demo with Accelerometer

Lab 6: TWR-LCD Display Orientation

Using the TWR-CN128 acceleration sensor, it is possible to change the display orientation on the fly. For this select the configuration below:

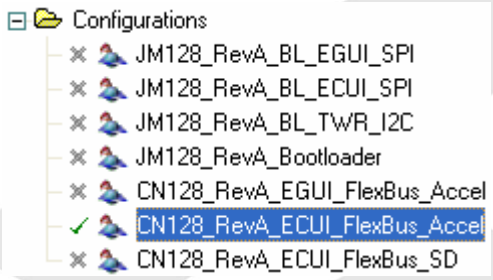

Figure 66: Accelerometer Display Orientation Configuration

Make sure that your jumper settings match the configuration comment.

In ‘platform.h’, verify that you have enabled the ‘PL_HAS_ACCELEROMETER_DEMO’ enabled:

```
/* demo configuration for Embedded Component UI */
#define PL_HAS_TOUCHSCREEN_DEMO    (1 && PL_USE_UI_EUI && PL_HAS_HW_TOUCHSCREEN)  /
#define PL_HAS_CALIBRATION_DEMO    (1 && PL_USE_UI_EUI && PL_HAS_HW_TOUCHSCREEN)  /
#define PL_HAS_CUBE_DEMO           (1 && PL_USE_UI_EUI)  /* if we include the 3D ro
#define PL_HAS_TETRIS_DEMO         (1 && PL_USE_UI_EUI)  /* if we inclued the tetri
#define PL_HAS_FONT_DEMO           (0 && PL_USE_UI_EUI)  /* if we show the font dem
#define PL_HAS_CALENDAR_DEMO       (0 && PL_USE_UI_EUI)  /* if we show the calendar
#define PL_HAS_TASKLIST            (1 && PL_USE_UI_EUI && PL_USE_RTOS) /* if we sho
#define PL_HAS_ACCEL_DEMO          (1 && PL_USE_UI_EUI && PL_HAS_HW_ACCELEROMETER)
#define PL_HAS_SD_DEMO             (1 && PL_USE_UI_EUI && PL_HAS_HW_SD_CARD) /* if
#define PL_HAS_PNG_DEMO            (1 && PL_HAS_SD_DEMO) /* if we have the PNG file
#define PL_HAS_HID_DEMO            (0 && PL_HAS_HW_USB) /* if we include the HID (m
```

Figure 67: Platform.h with Accelerometer demo enabled

Build and download with the debugger your application to the CN128 and start it.

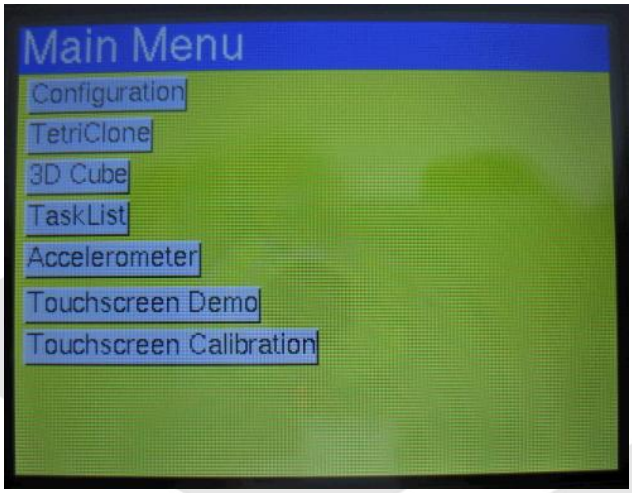

Figure 68: Demo main menu

Using the ‘Configuration’ button you open the configuration dialog:

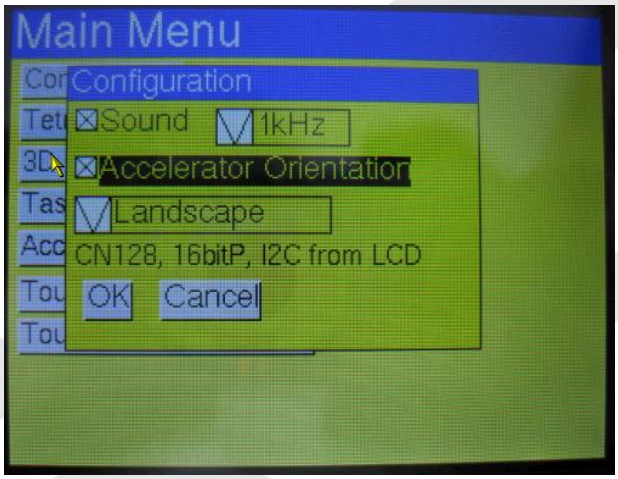

Figure 69: Configuration using Accelerometer Orientation

Enable the checkbox to change the display orientation according to the accelerometer. Press OK and watch the display changing depending on the Accelerometer orientation.