Freescale TWR-LCD Lab Tutorial

Table of Content

[Overview 2](#_Toc288334689)

[Installing Processor Expert TWR-LCD Embedded Components 2](#_Toc288334690)

[Demo Licensing Limitations 3](#_Toc288334691)

[Lab 1: TWR-LCD Stand-alone 3](#_Toc288334692)

[Lab 2: TWR-LCD Bootloader 5](#_Toc288334693)

[Building the MCF51JM128 Bootloader 5](#_Toc288334694)

[Using the MCF51JM128 Bootloader 6](#_Toc288334695)

[Memory Map for Bootloader and Application 8](#_Toc288334696)

[Bootloader Build Options 9](#_Toc288334697)

[Application Build Options 13](#_Toc288334698)

[Lab 3: TWR-LCD Processor Expert Embedded UI Demo 16](#_Toc288334699)

[Configuring the demo amount 17](#_Toc288334700)

[Lab 4: TWR-LCD with TWR-MCF51CN128 with Accelerometer 18](#_Toc288334701)

[Power the system 19](#_Toc288334702)

[Building and downloading the demo 19](#_Toc288334703)

[Using the switches on the TWR-MCF51CN128 19](#_Toc288334704)

[Using the navigation switch with the TWR-MCF51CN128 20](#_Toc288334705)

[Lab 5: TWR-LCD Display Orientation 21](#_Toc288334706)

# Overview

There are following projects provided:

C:\Users\ERICHS~1\AppData\Local\Temp\SNAGHTML206ab46.PNG

Figure : Demo Projects

* **TWR-LCD CN128**: Demo where a TWR MCF51CN128 CPU card uses the display with a FlexBus interface. The MCF51JM128 can be flashed with the bootloader with an application to send navigation switch messages over I2C to the CN128.
* **TWR-LCD JM128**: This project comes with 3 different Processor Expert configurations: one to demonstrate the LCD without the bootloader (JM128\_RevA\_noBL\_ECUI\_SPI), one doing the same, but enabled for the bootloader ((JM128\_RevA\_BL\_ECUI\_SPI) and a bootloader application which does not use the LCD, but routes the navigation switch messages over I2C to another Tower CPU card (JM128\_RevA\_BL\_TWR\_I2C)
* **TWR-LCD JM128 Bootloader**: Project to build the bootloader. Using the bootloader you can load any bootloader enabled project over the USB port of the MCF51JM128

The projects are sharing sources, so you need to have all projects open and present in your workspace.

## 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’ > ‘Import Package’

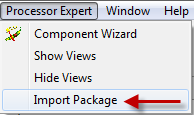


Figure : Updating Processor Expert from Package

* Browse to the ‘TWR-LCD\_Components\*.PEupd’ file located in the TWR-LCD JM128 CodeWarrior demo project folder and import all the components
* Close CodeWarrior IDE and restart CodeWarrior: this will make sure the new components are recognized
* Import the CodeWarrior project into eclipse

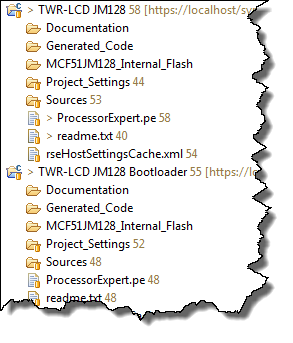


Figure : CodeWarrior projects

## Demo Licensing Limitations

CodeWarrior comes after installation with a code size limited Special Edition License. The Processor Expert demo requires a Professional License to generate code for all configurations. As such, the demos are provided as well in non-Processor Expert Format. Additionally you can configure (increase/decrease) the amount of demos using the platform.h header file for the Embedded Component demos.

S19 (S-Record) files are provided of full demos so you can download or use them with the bootloader without any restrictions.

# Lab 1: TWR-LCD Stand-alone

In this demo we are going to demonstrate the LCD in standalone mode (the MCF51JM128 is using the LCD without the bootloader).

Make sure you are using following jumper settings:

1:OFF (SPI to display)

2:ON (SPI to display)

3:OFF (SPI from JM128)

4:OFF (uSD to JM128)

5:OFF (SPI0 CS0 as chip select)

6:OFF (Touch screen by JM128)

7:ON (LCD backlight on)

8:OFF (Buzzer to JM128)

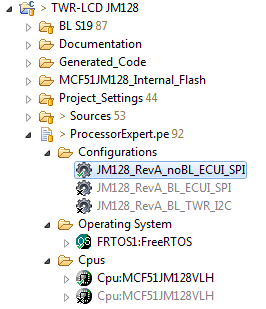


Figure : Standalone project

* Load the TWR-LCD JM128 project
* Make sure JM128\_RevA\_noBL\_ECUI\_SPI is selected as configuration
* Generate Processor Expert code
* Build and flash the project
* Run the application: it will show as the first thing a touch screen calibration

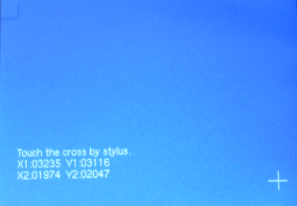


Figure : Touch Calibration

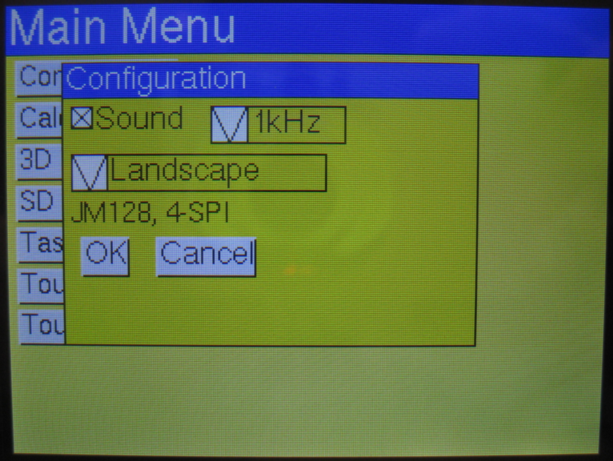


Figure : Processor Expert Embedded Component GUI Demo

# Lab 2: TWR-LCD Bootloader

## 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 you host system, as the bootloader is getting the S19 files from the host through a USB connection.

The bootloader project is the ‘TWR-LCD JM128 Bootloader’ one:

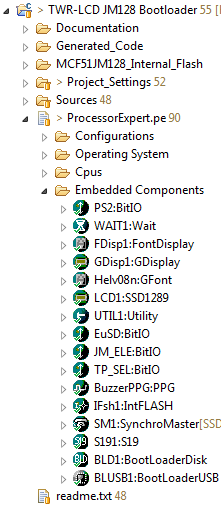


Figure : Bootloader Project

Generate Processor Expert code and download the application to the TWR-LCD MCF51JM128.

## Using the MCF51JM128 Bootloader

The bootloader allows you to load applications to the target without the need for an BDM cable. 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) and press within about 200ms as well the BTLD button

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

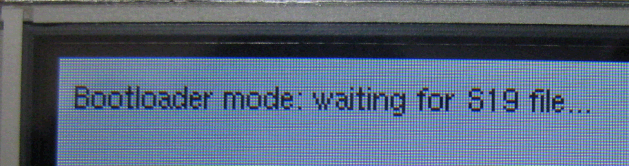


Figure : Bootloader LCD message

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

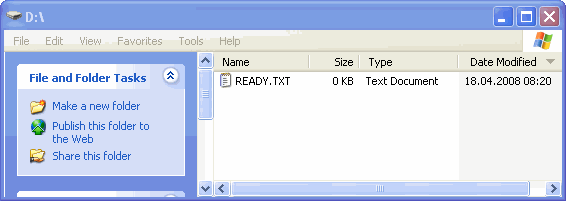


Figure : Bootloader recognized as mass storage device

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

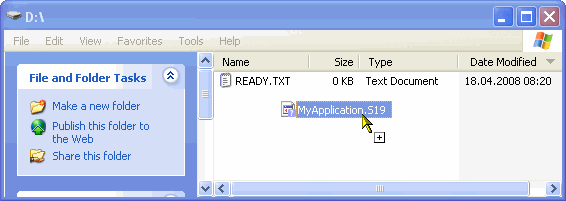


Figure : Drag&Drop S19 file to bootloader

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 a beep.

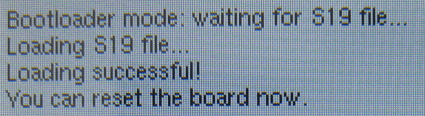


Figure : Bootloader has flashed S19 file

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

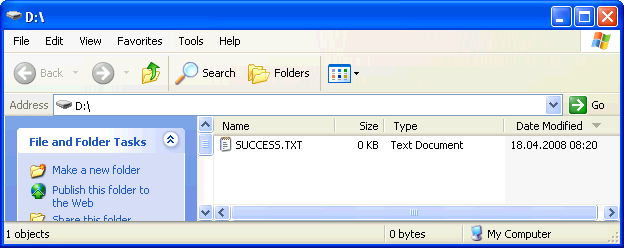


Figure : Bootloader successful MSD message file

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

## 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.

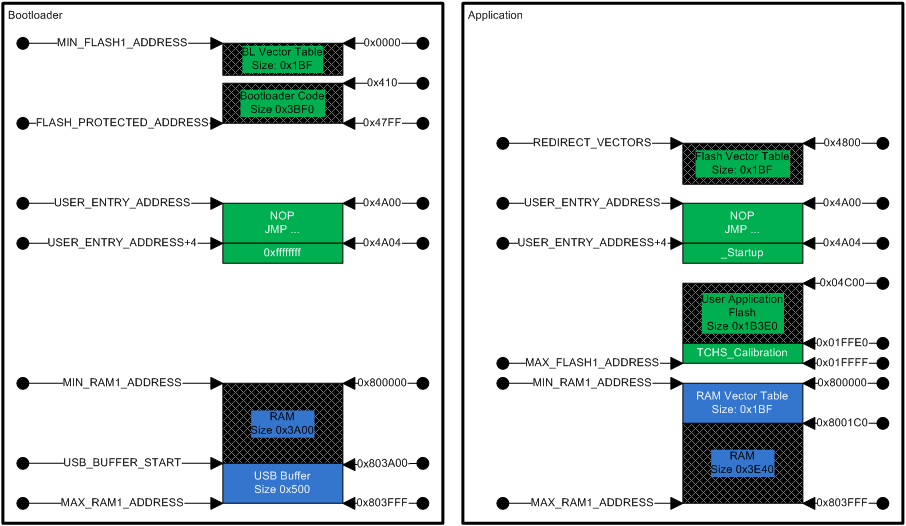


Figure : Bootloader and memory map

The application flash needs to be above FLASH\_PROTECTED\_ADDRESS.

## Bootloader Build Options

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

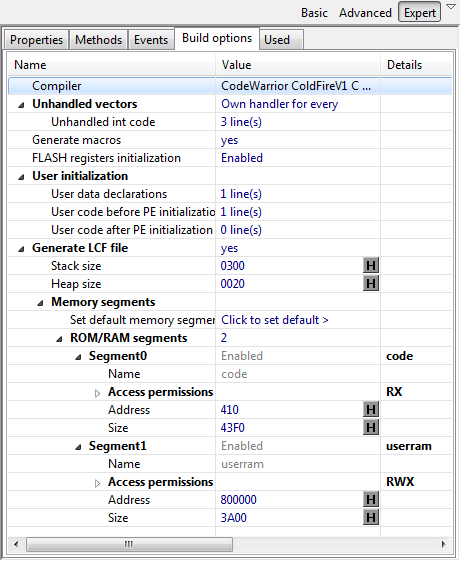


Figure : Bootloader Build Options

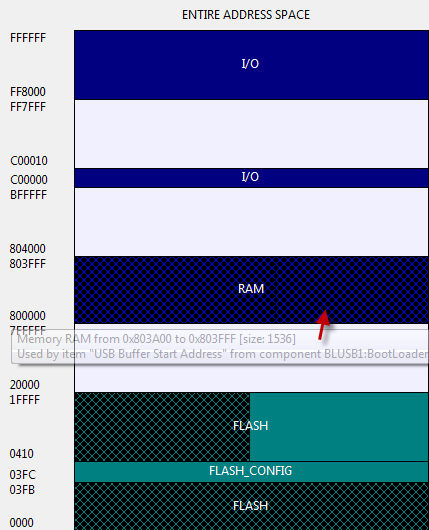


Figure : Bootloader memory map

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’:

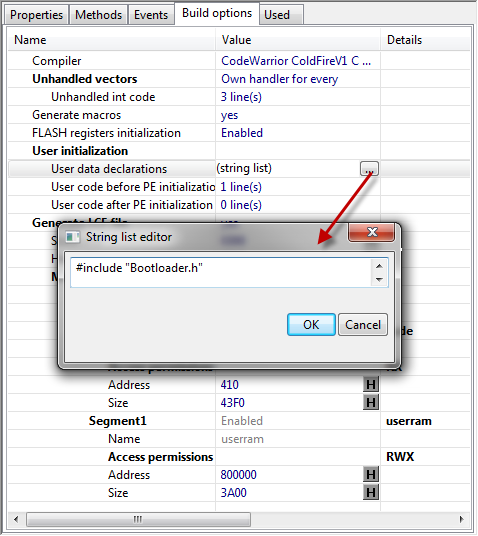


Figure : Bootloader User Data Declarations

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’.

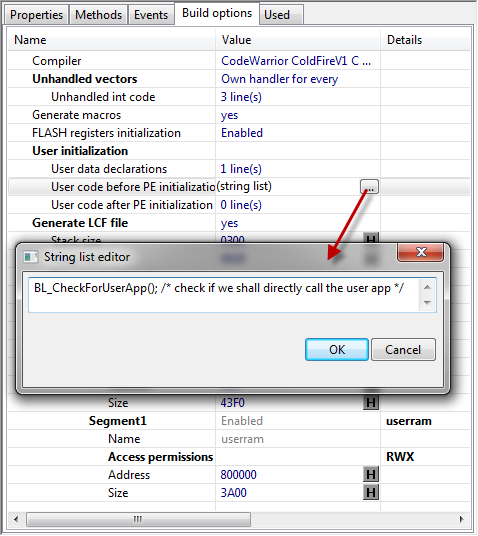


Figure : Bootloader Use code before PE initialization

## Application Build Options

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

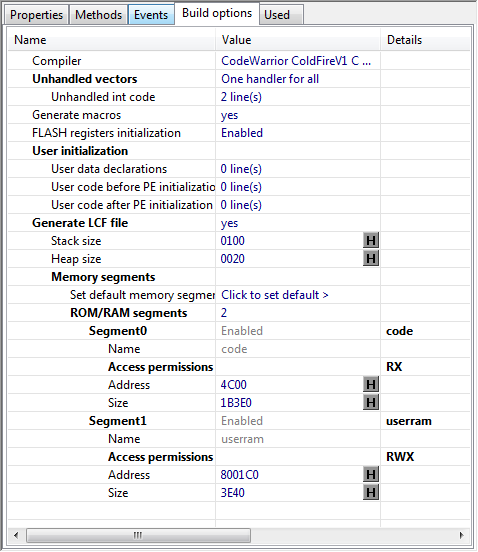


Figure : Application Build Options

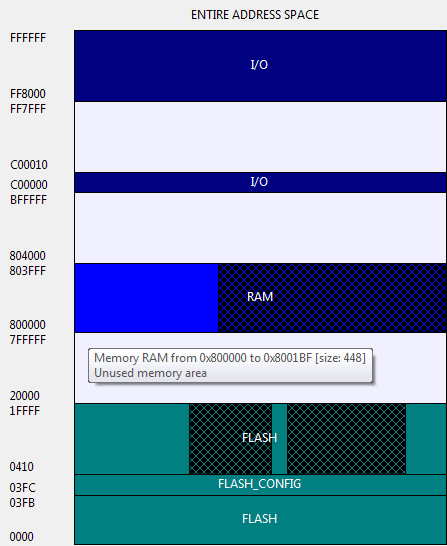


Figure : Application memory map

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

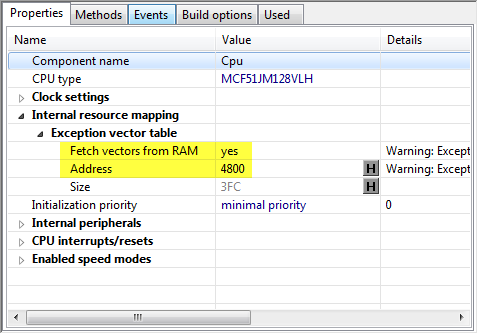


Figure : Application vector table settings

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

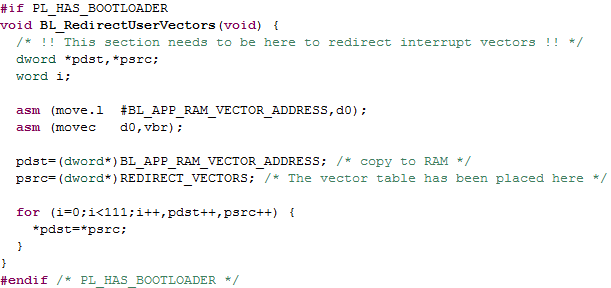


Figure : Application vector table copy and redirection

# Lab 3: 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.

In order to switch to this demo, select the ‘JM128\_RevA\_BL\_ECUI\_SPI’ Configuration:

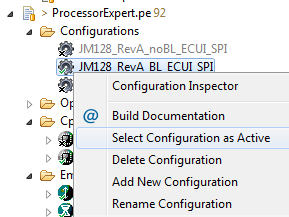


Figure : Selecting the Embedded Component UI Demo

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

## Configuring the demo amount

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

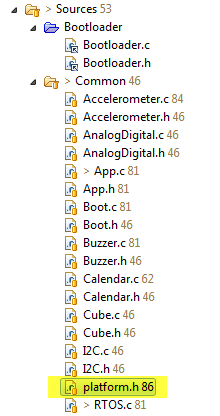


Figure : platform.h header file

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.

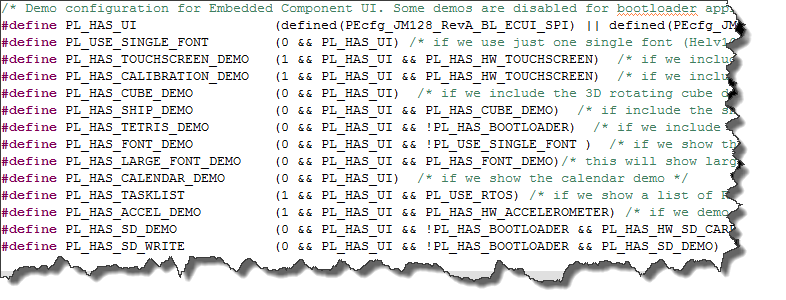


Figure : platform.h demo configuration

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 4: TWR-LCD with TWR-MCF51CN128 with Accelerometer

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

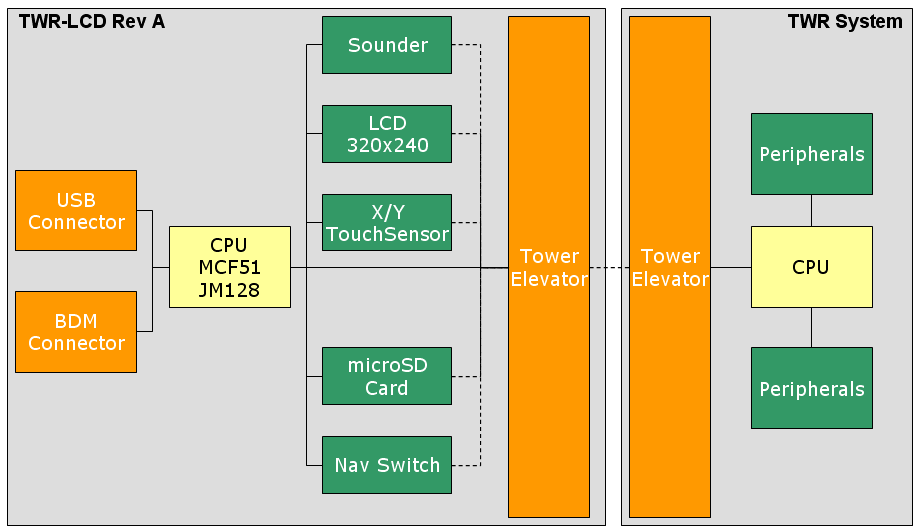


Figure : System Configuration with a TWR System

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.

Make sure your project used is the TWR-LCD CN128 one:

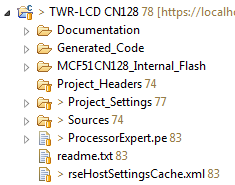


Figure : TWR MCF51CN128 LCD Project

The readme.txt in the project contains important information and jumper settings.

## Power the system

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

1. use the TWR-ELEVATOR as the power source for the TWR-LCD and the rest of the TWR-System
2. The mini-USB connector on the TWR-LCD you need only in addition to the TWR-ELEVATOR one in case you need access to the USB bus of the JM128 (e.g. to flash an application with the bootloader)

## Building and downloading the demo

Generate Processor Expert code, build the application and download it to the MCF51CN128 board.

## 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.

## 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.

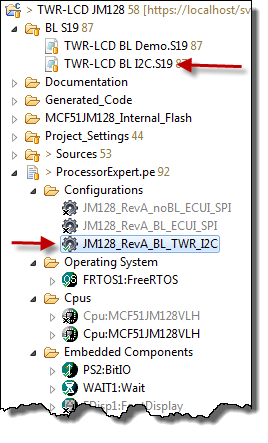


Figure : JM128 I2C Application

The BL S19 folder already contains the bootloader enabled S19 application file. Otherwise you can build the application yourself too.

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.

# Lab 5: TWR-LCD Display Orientation

Using the TWR-CN128 acceleration sensor, it is possible to change the display orientation on the fly.

In ‘platform.h’, verify that you have enabled the ‘PL\_HAS\_ACCEL\_DEMO’ enabled:

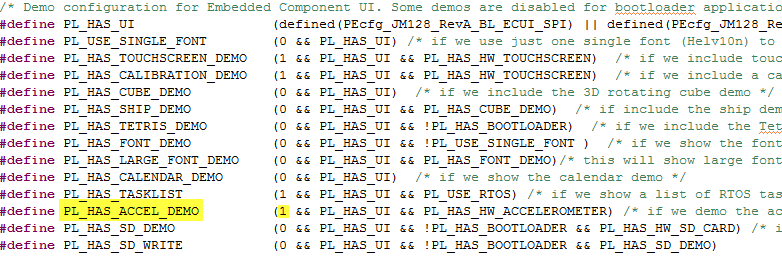


Figure : Platform.h with Accelerometer demo enabled

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

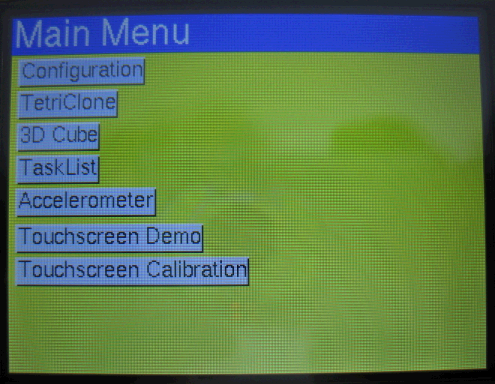


Figure : Demo main menu

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

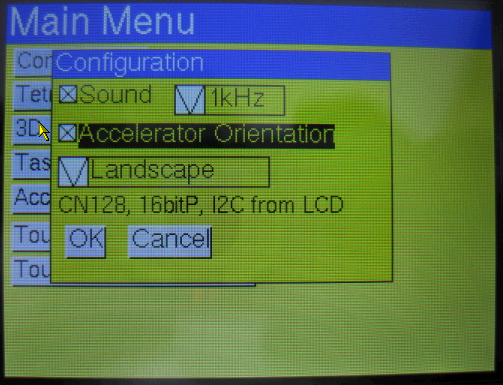


Figure : Configuration using Accelerometer Orientation

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