FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Hands-on Workshop: Xtrinsic Sensor Fusion

FTF-ENT-F0091

Fengyi Li

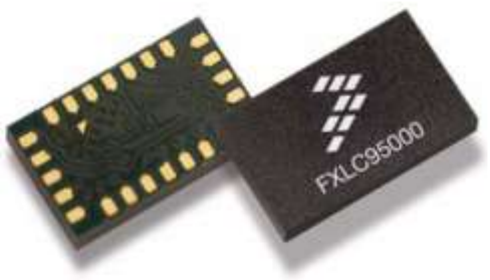
James Scott

Applications Engineer

June 2012

Session Introduction - FXLC95000L

- Motion Sensing Platform: The FXLC95000L extends 3-axis accelerometer functionality with a programmable core and on-chip memory (128K Flash, 16K RAM) to enable advance motion sensing capability.
- Decision Engine: The FXLC95000L is a highly flexible and configurable sensor decision engine that enables complex calculations and local decision making.



Session Introduction – Objective

- **Understand the target markets of the FXLC95000L**
- **Build an FXLC95000LCodeWarrior application**
- **Utilize the FXLC95000L subsystems to create simple sensor based applications**

Agenda

- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

Agenda - Lab

- Setup up the lab environment and demo project
  - Lab 1: Import, build and download demo project
- Lab 2 – Lab 5: Modify the MQX based code
  - Lab 2: Toggle LED using the RGPIOs
  - Lab 3: Acquire AFE data
  - Lab 4: Toggle LED based on Gesture Recognition
  - Lab 5: Communicate with FXLC95000L via slave I2C ports
- Demo: Use FXLC95000L as a sensor hub

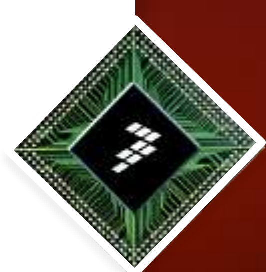
Class rules

- **Please be courteous to your fellow classmate**
- **Questions? Raise your hand**
- **Lab? Find your lab assistances**

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000L Preview

FTF-ENT-F0091



Agenda

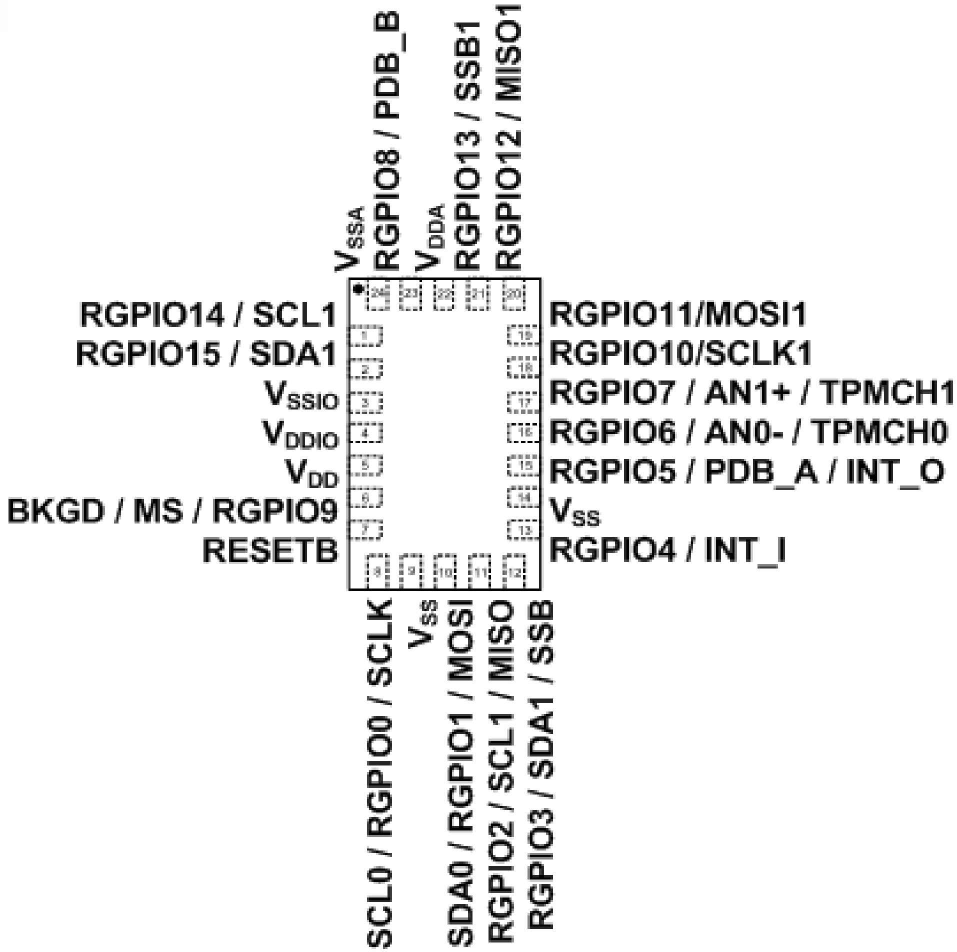
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

FXLC95000 Preview

- **Accelerometer + MCU**

- **I2C / SPI**
- **Peripherals w/ GPIO**

- **3 mm X 5mm x 1mm**
- **24 pin**

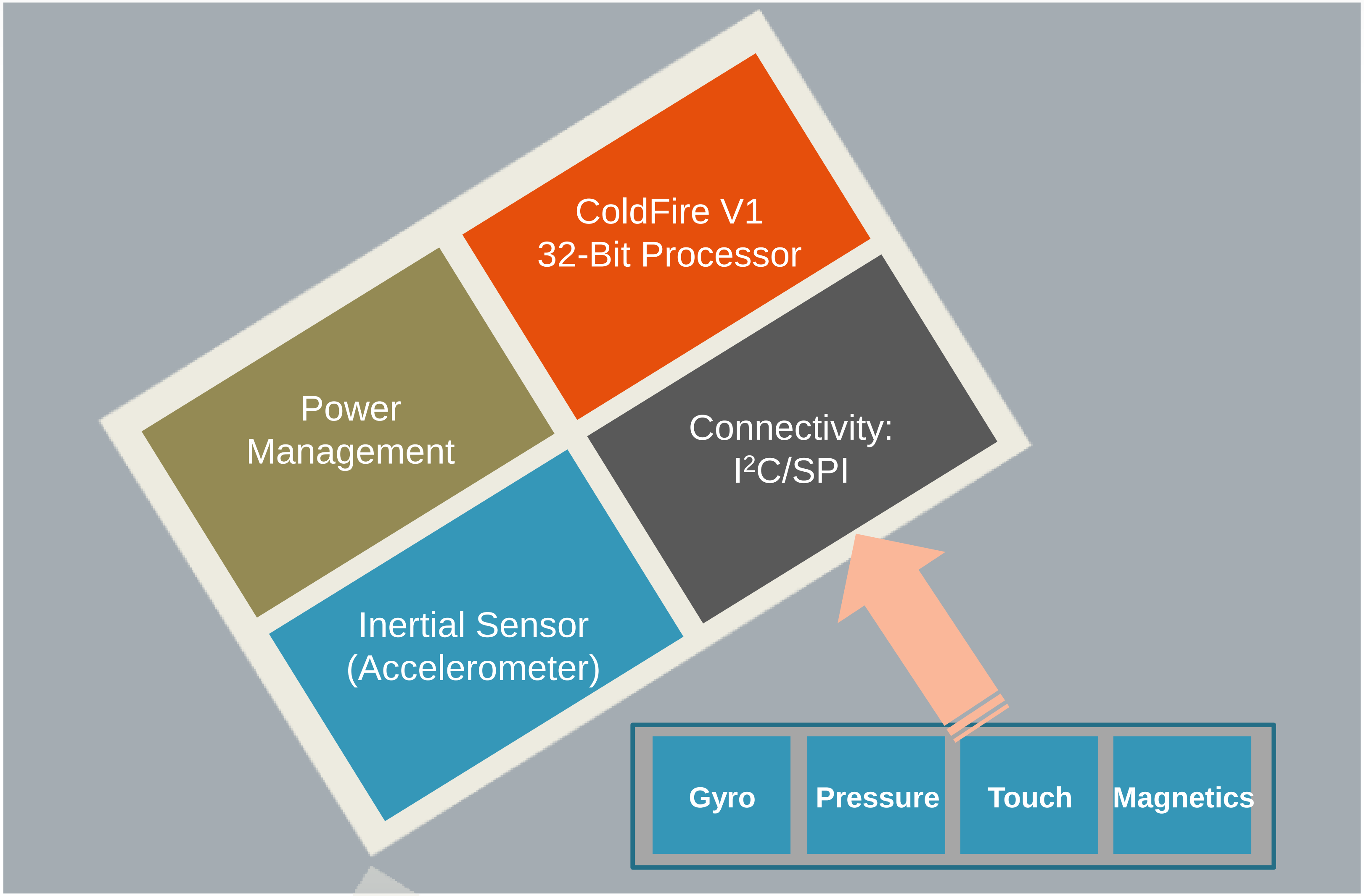


freescale™

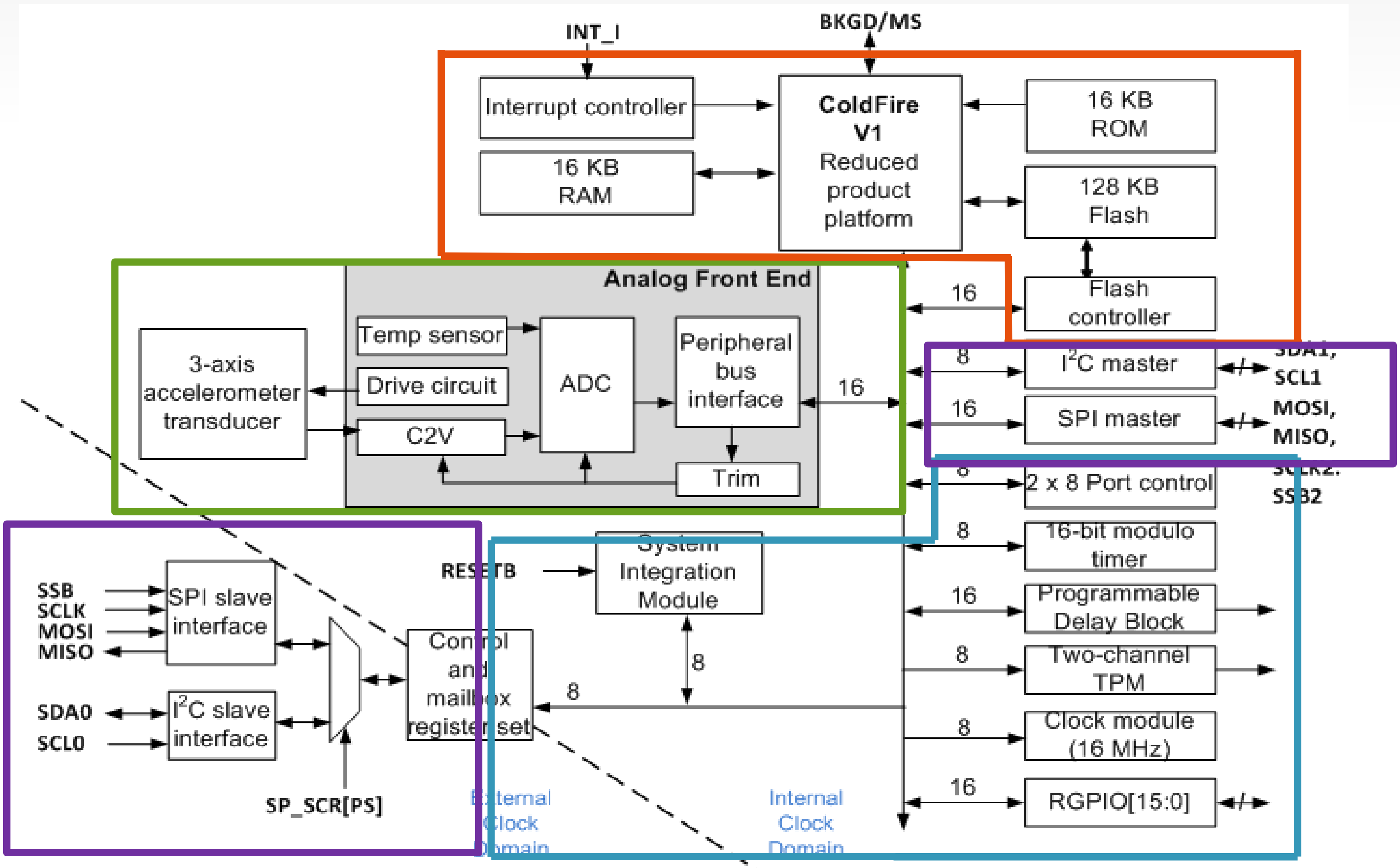
Typical Applications

- **Activity monitor**
- **Remote control**
- **Handheld scanner**
- **Tablet**
- **Smart Phone**
- **Health Monitor**

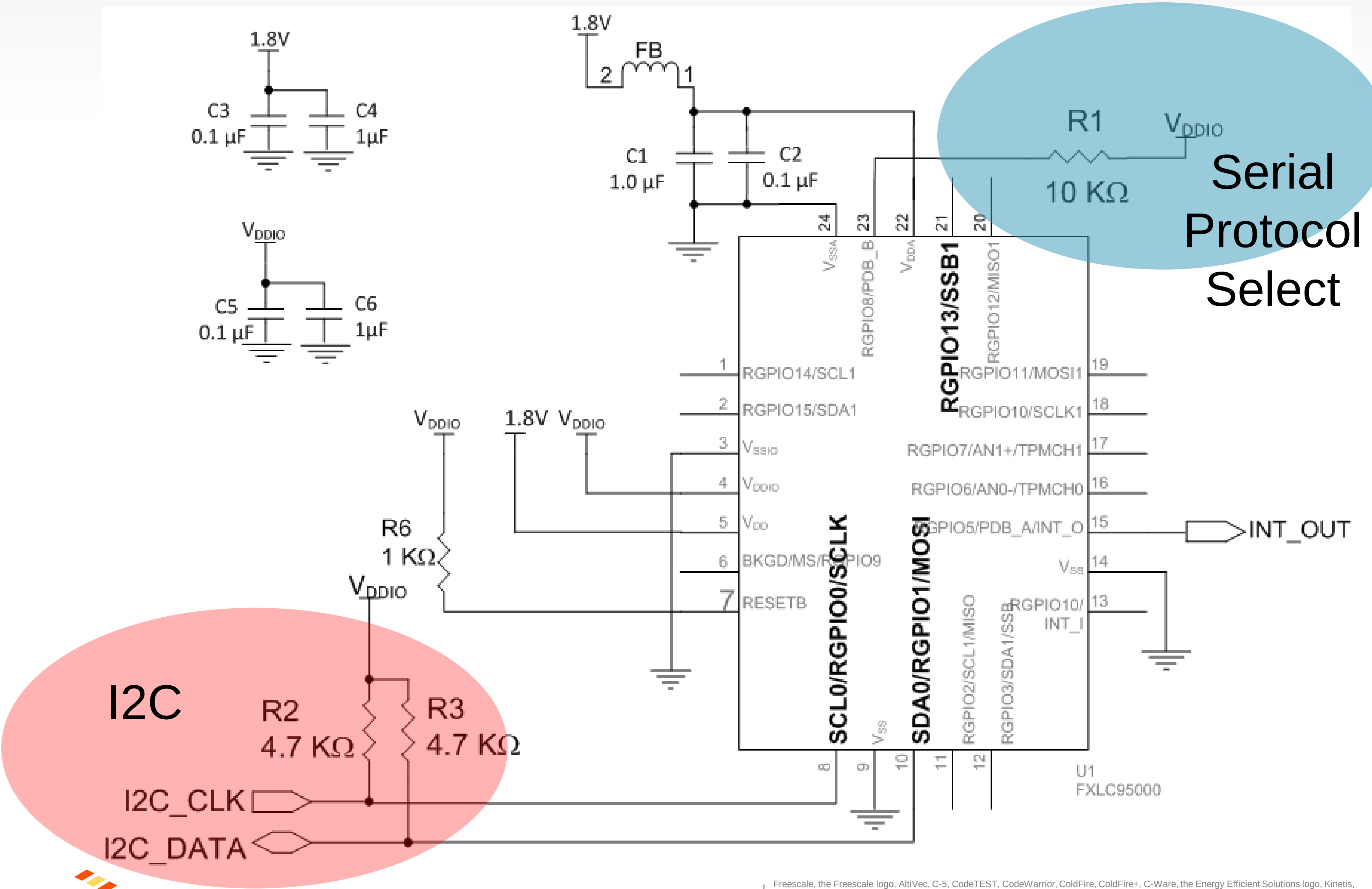
High Level FXLC95000L



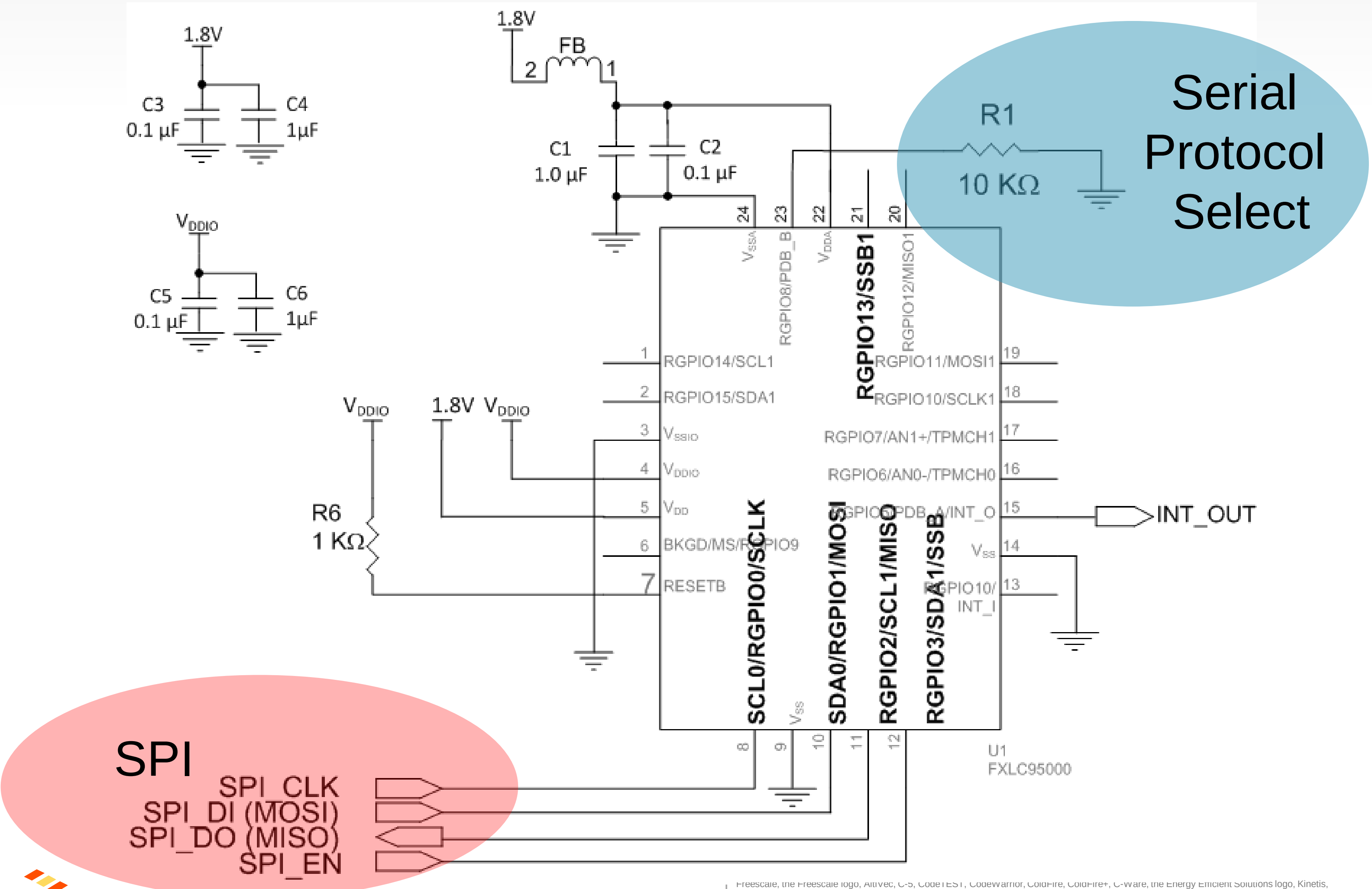
FXLC95000L Block Diagram



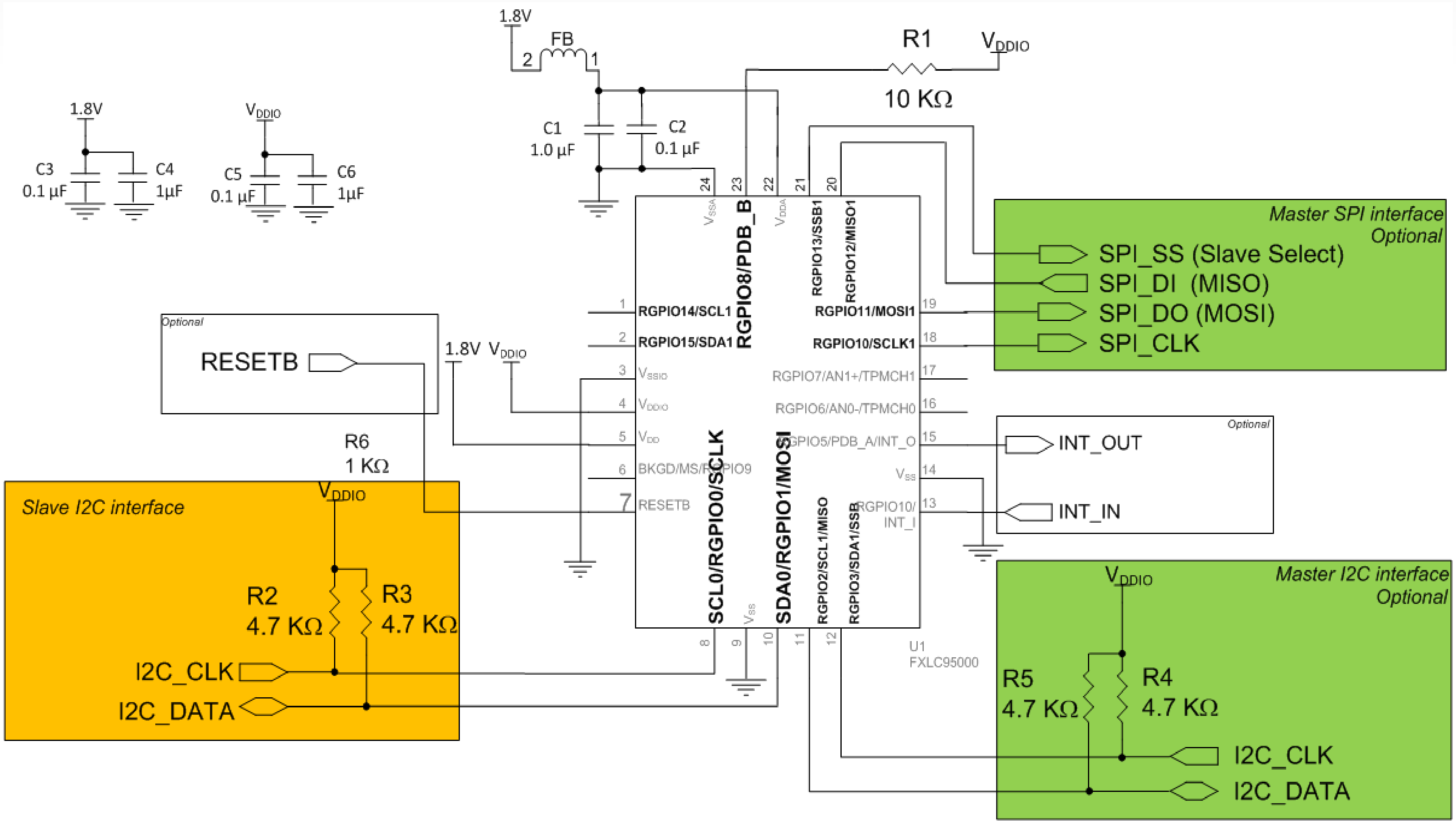
Platform as a Slave – I2C Interface Selected



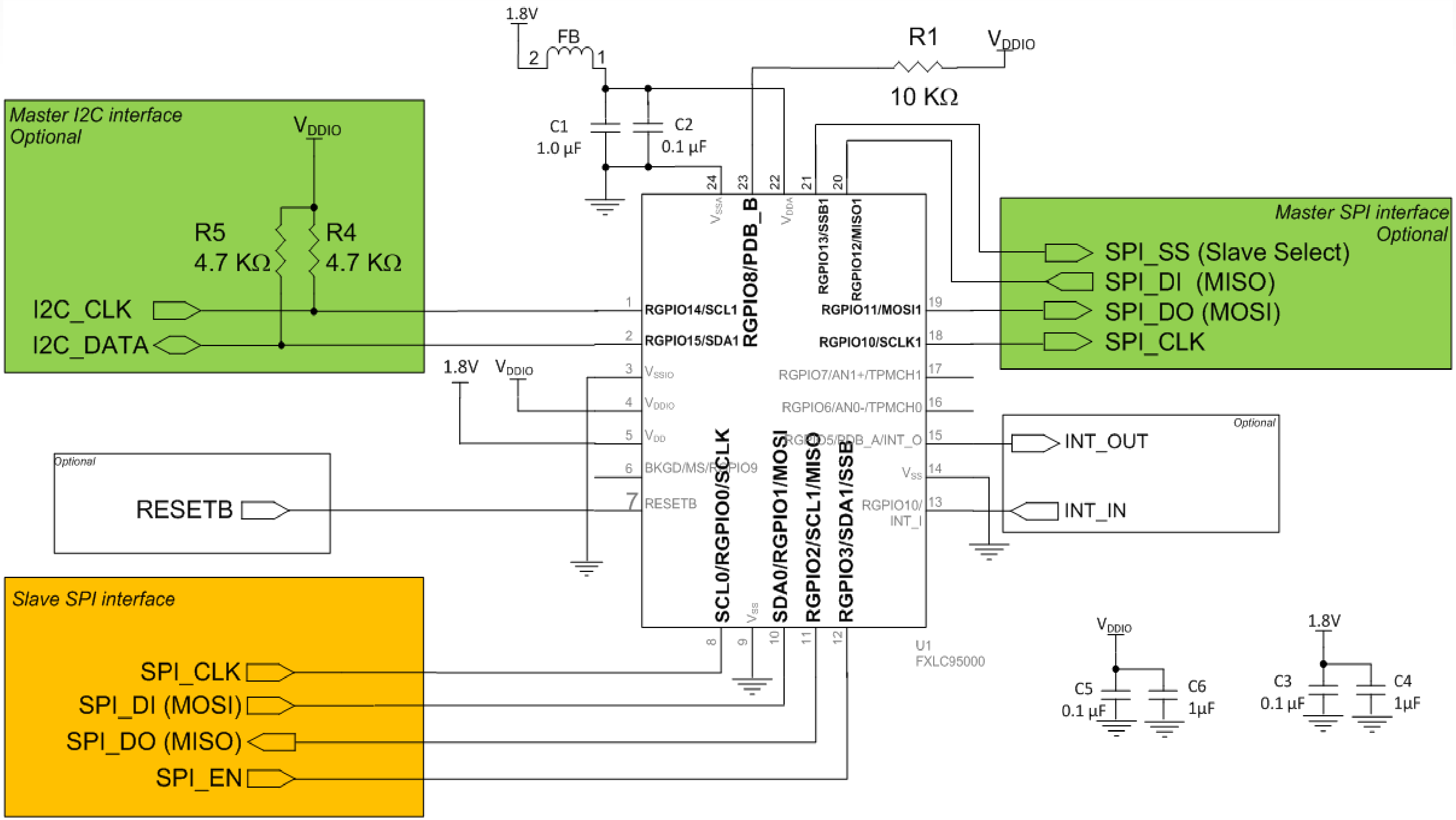
Platform as a Slave – SPI Interface Selected



Platform as a Sensor Hub (I2C)



Platform as a Sensor Hub – (SPI)



FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Designing with the FXLC95000L Platform

FTF-ENT-F0091

Agenda

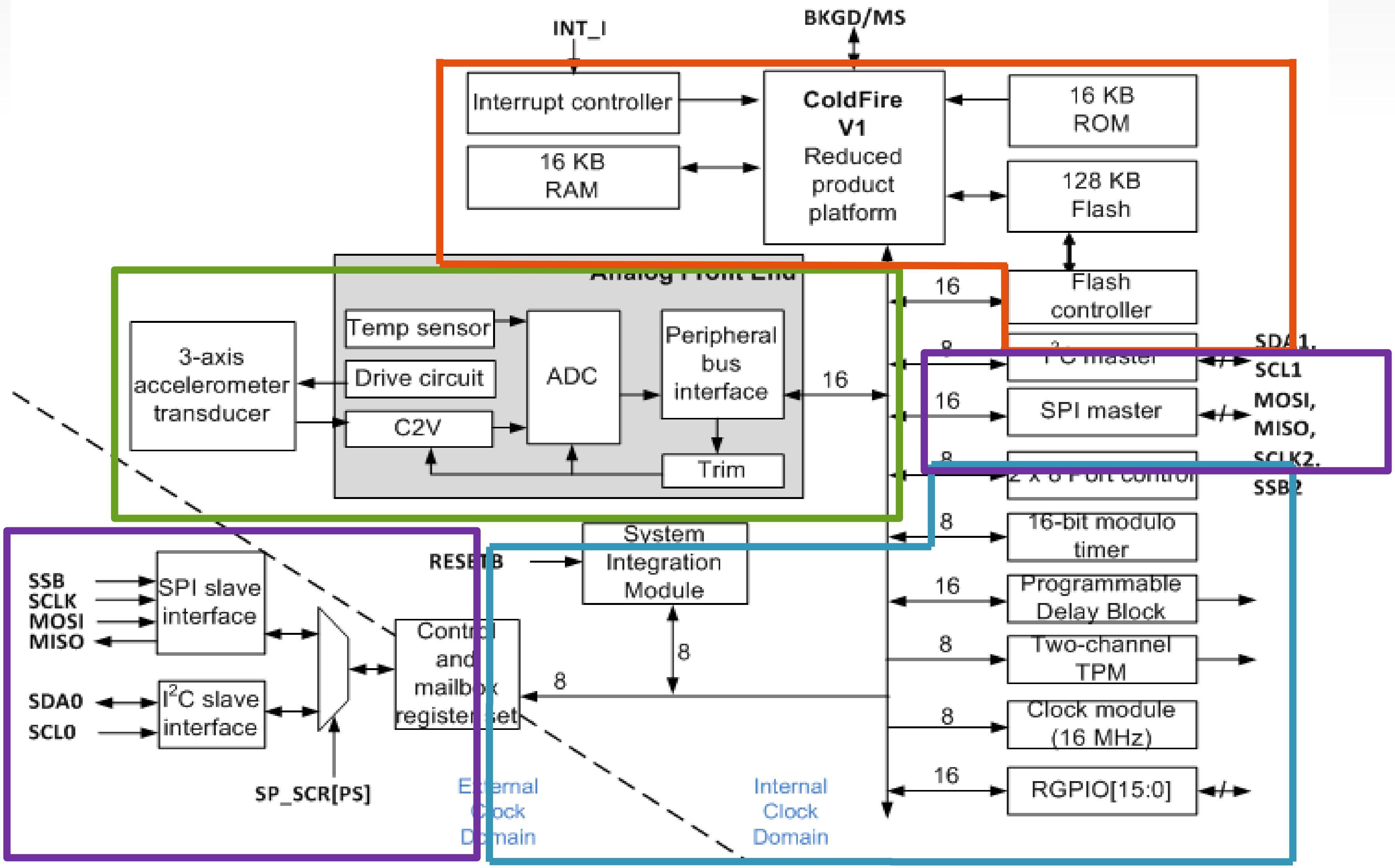
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

Designing with the FXLC95000L Platform

- **Objective**
  - Gain an overview of the FXLC95000L AFE, System Clock, GPIO, Timer and I2C modules
  - Learn the basics on how to create custom FXLC95000L smart sensing solutions project in CodeWarrior 10.1
  - Create a custom application and program the FXLC95000L smart sensor platform to toggle an LED based on a various gesture recognition

- **Pre-requisites**
  - A basic understanding of micro-controllers and inertial sensors is needed to maximize the utility of this "Designing with the FXLC95000L" training module

FXLC95000L Block Diagram



FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

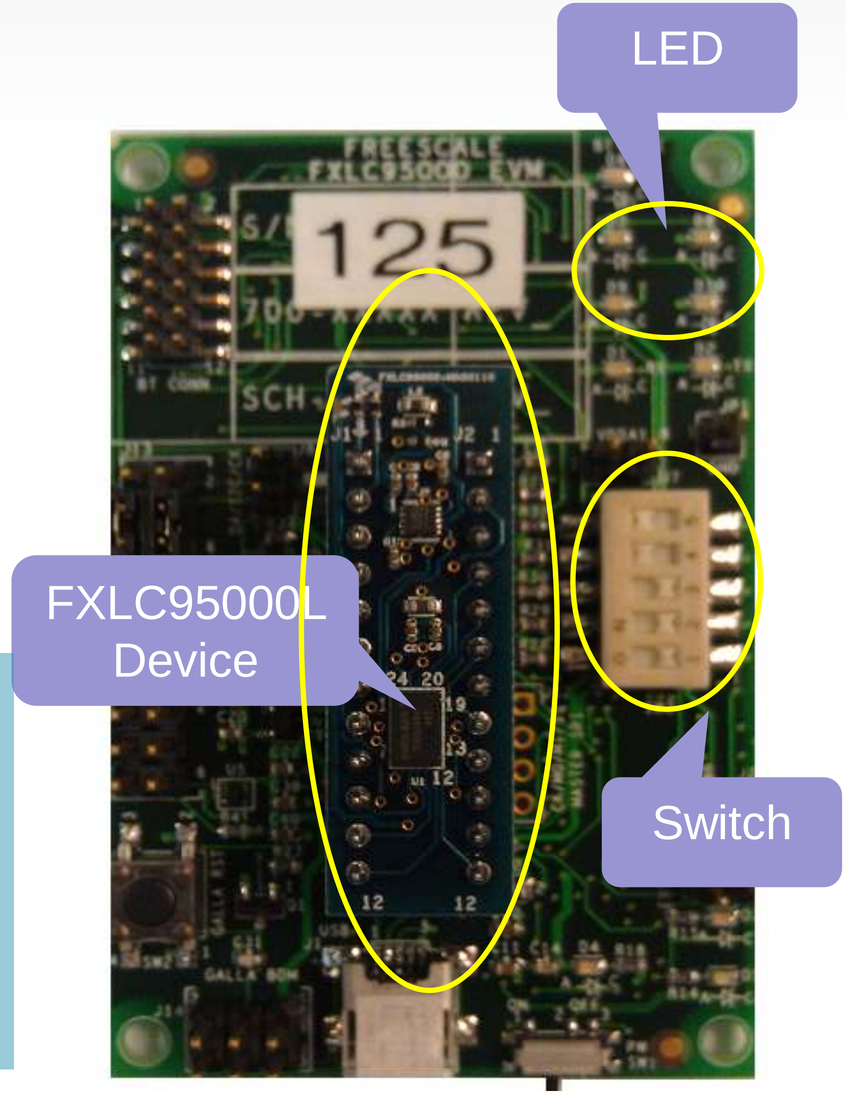
Development Hardware and Software Overview

Agenda

- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

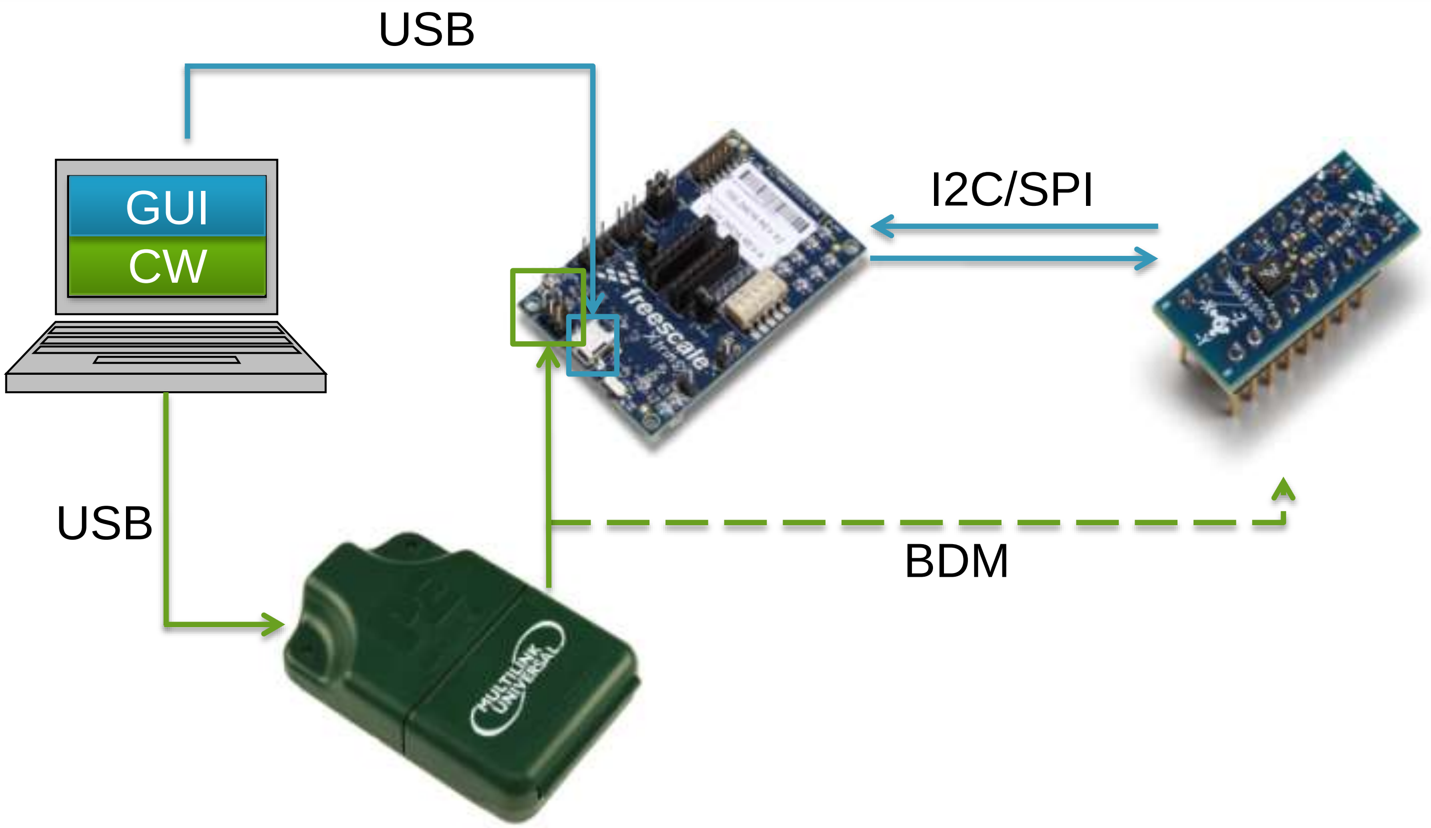
Development Hardware and Software Overview

- Device Programming Hardware
  - BDM programmer:
    - P&E USB Multilink or Cyclone Pro
  - FXLC95000L Evaluation Kit
    - Programming platform
    - (In this lab) Program evaluation tool



USB-to-I²C or USB-to-SPI, and BDM connectivity between a PC and the FXLC95000L device

Connections



Connect Hardware

- Connect **Multi-link** to PC
- Connect **Multi-link** to FXLC95000L programming board
- Connect FXLC95000L board to PC for power
- Turn on SW1 to power the FXLC95000L board

Development Hardware and Software Overview

- Computer Hardware / Software
  - Windows XP or newer
  - Code Warrior v10.1
  - CodeWarrior10.1 FXLC95000L service pack
  - MQX 3.7
  - MQX FXLC95000L service patch
  - FTDI driver

File Needed for the Lab Exercises

- Fxlc95000_Mqx_Template_Multithread.zip
- FTF_fxlc95000_demo_solution.zip

  – These files are in the folder on Desktop → FTF-ENT-F0091

NXP

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Lab 1: Import, Build and Download Demo Project

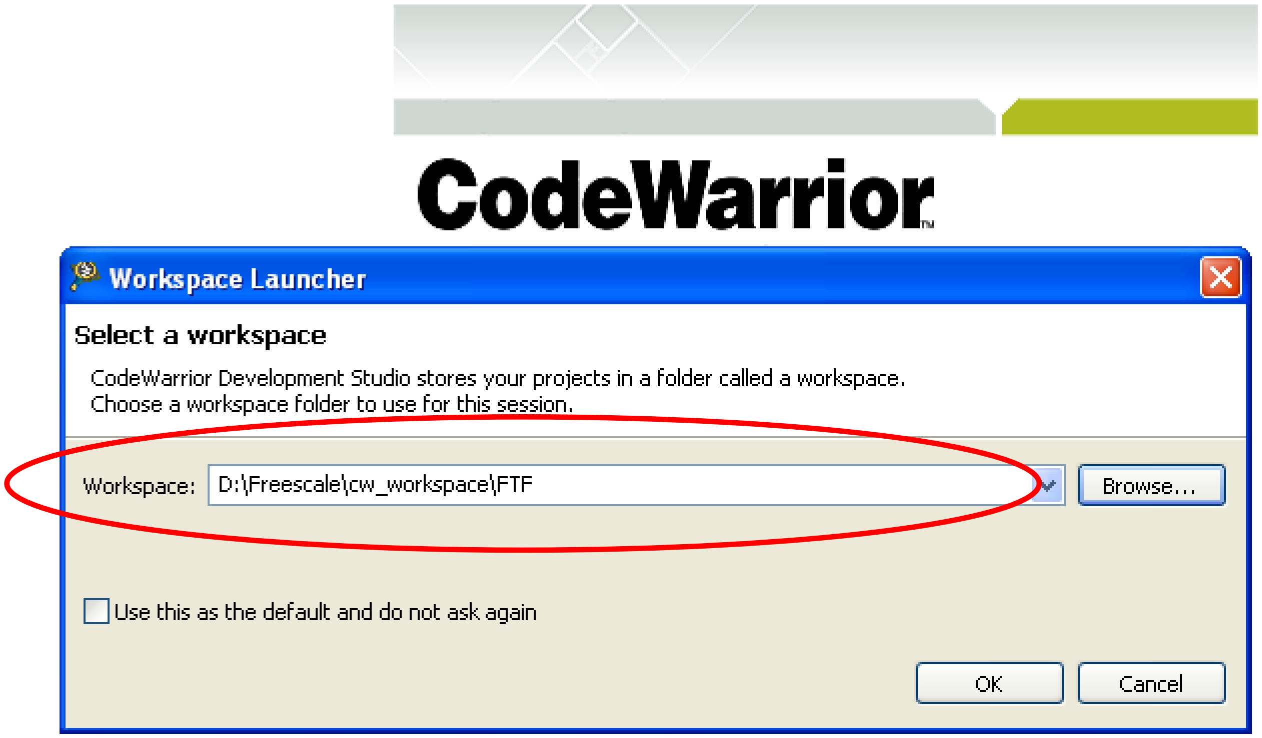
FTF-ENT-F0091

Lab 1: Import, Build and Download Demo Project

- **Object:**
  1. Verify CW 10.1 development environment
  2. Import, build and download demo project
  3. Observe: All LEDs toggle when SW1 is at ON position

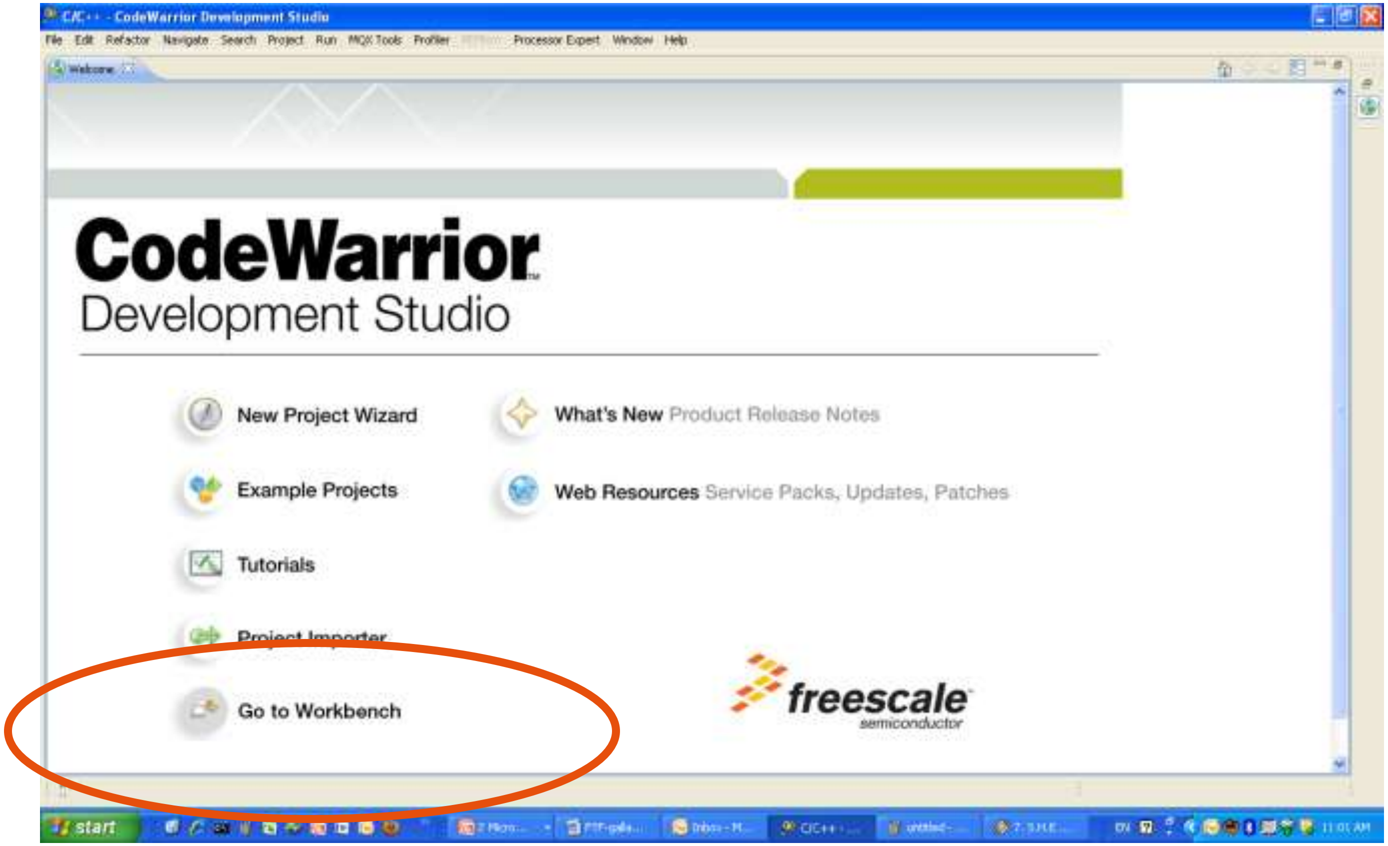
Lab1: CW Setup

- **Open → setup workspace**



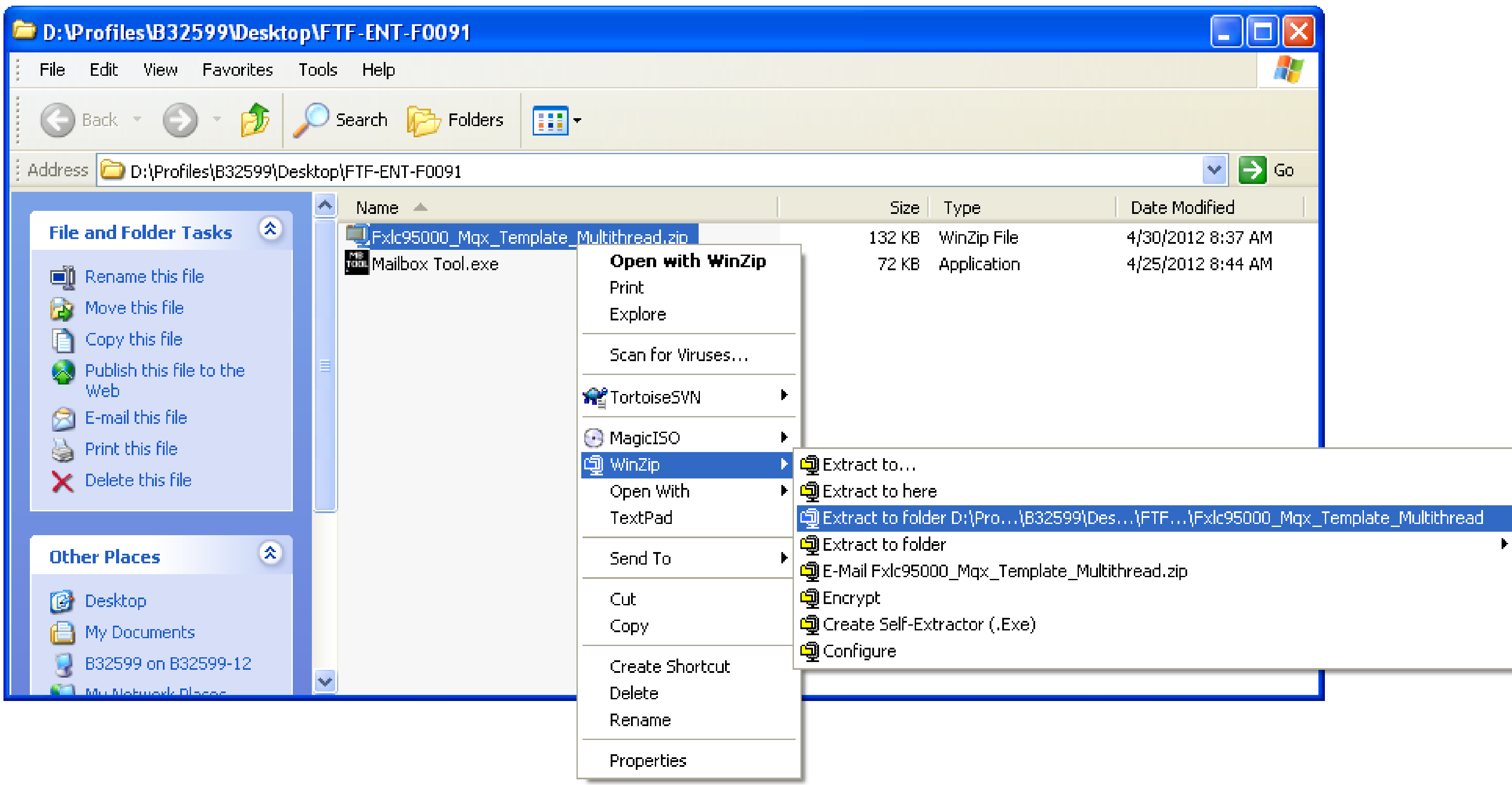
Lab1: CW Setup

- Choose **Go to Workbench** on the welcome page



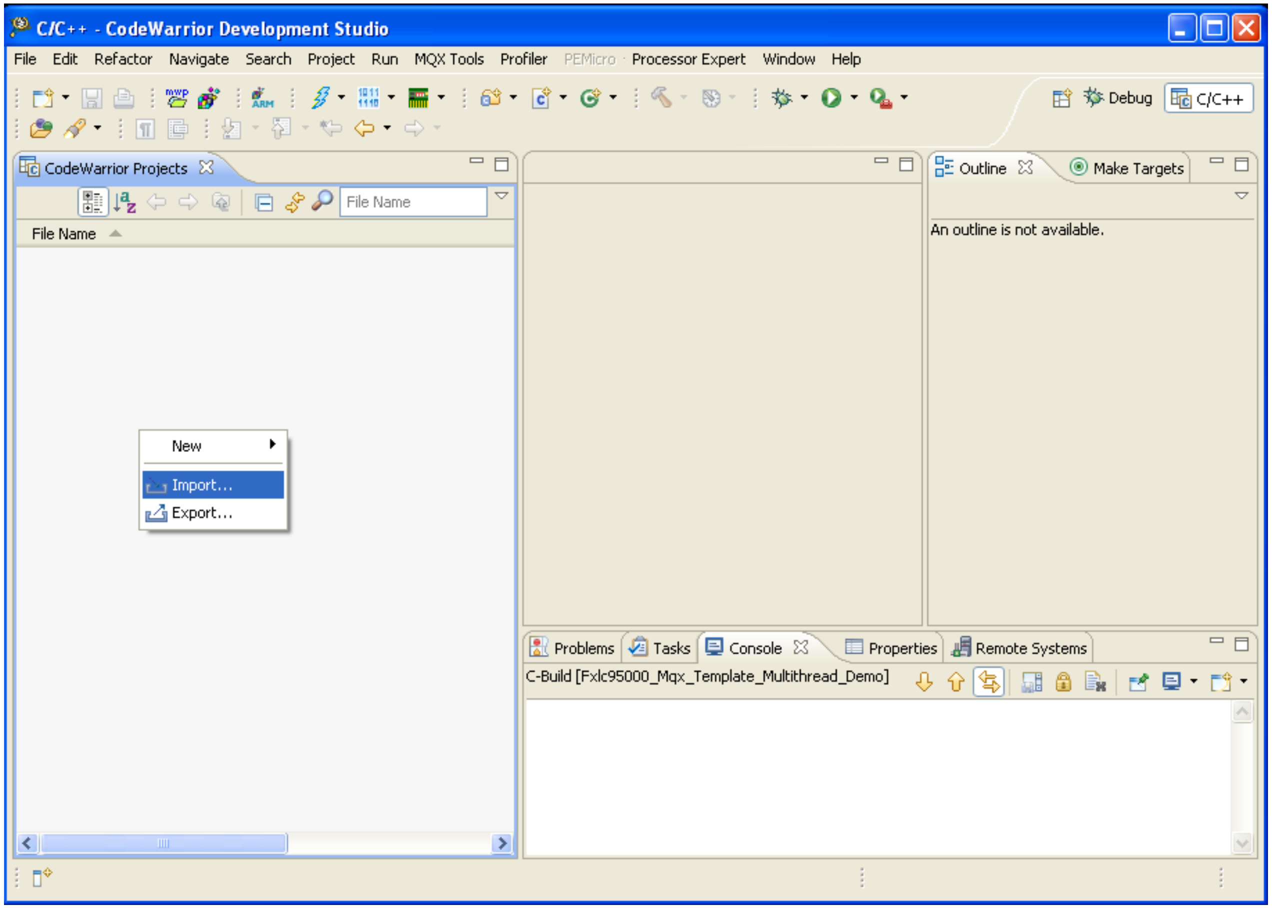
Lab1: Unzip the Project

- Go to project directory **FTF-ENT-F0091**, unzip demo file **Fxlc95000_Mqx_Template_Multithread.zip**



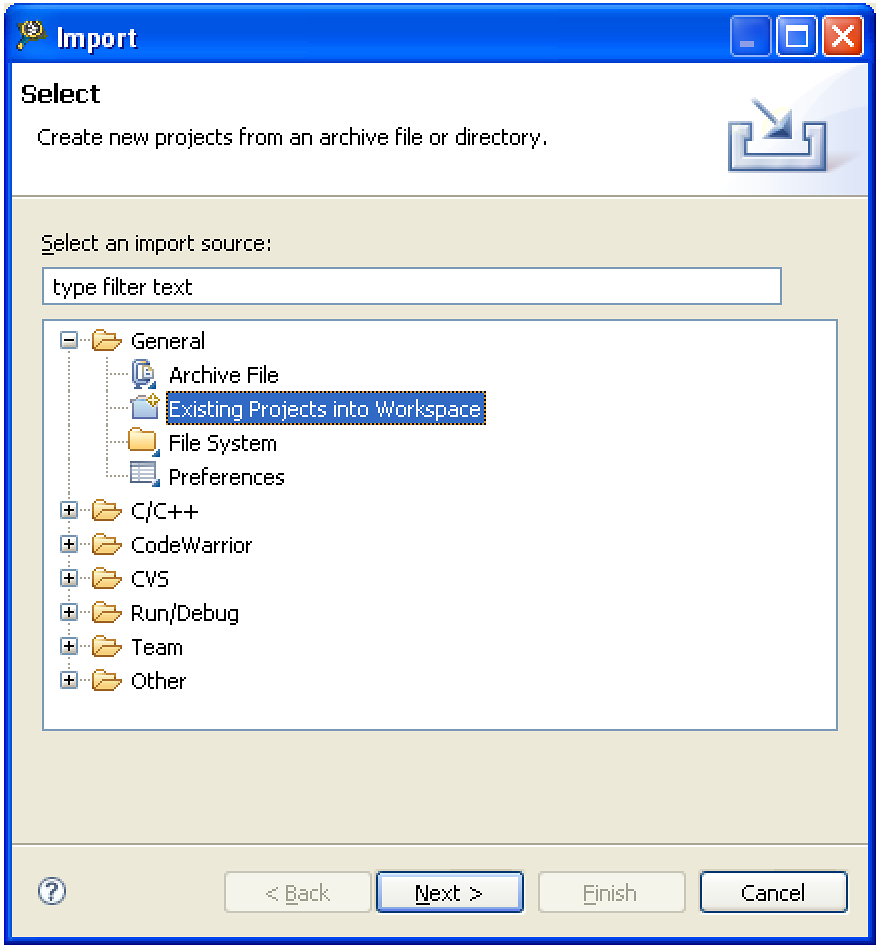
Lab1: Import Project

- In the **CodeWarrior Projects** window, right click → select **Import Project**



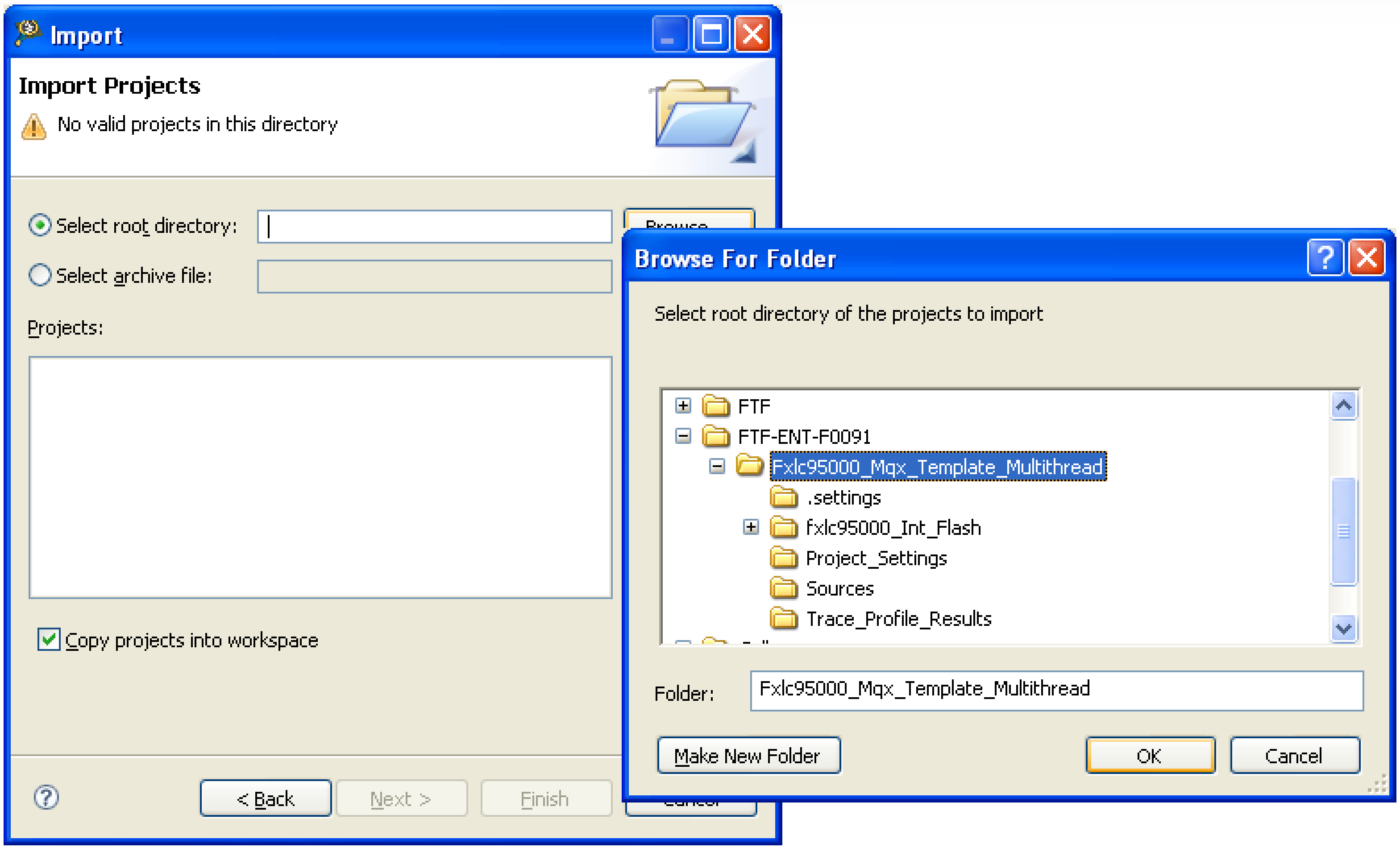
Lab1: Import Project

- Choose **Existing Projects into Workspace**

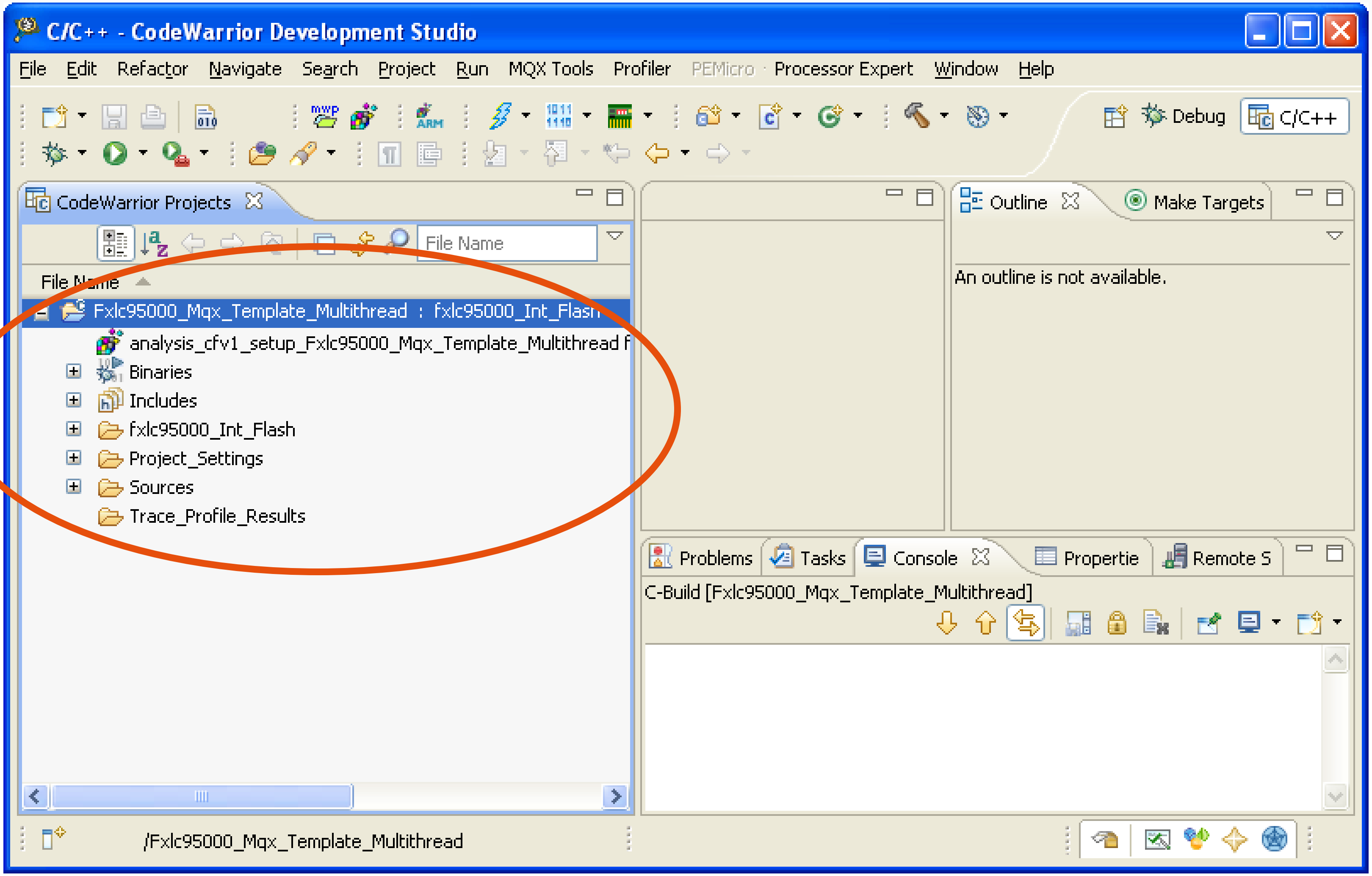


Lab1: Import Project

- Identify project root directory → click OK → click Finish

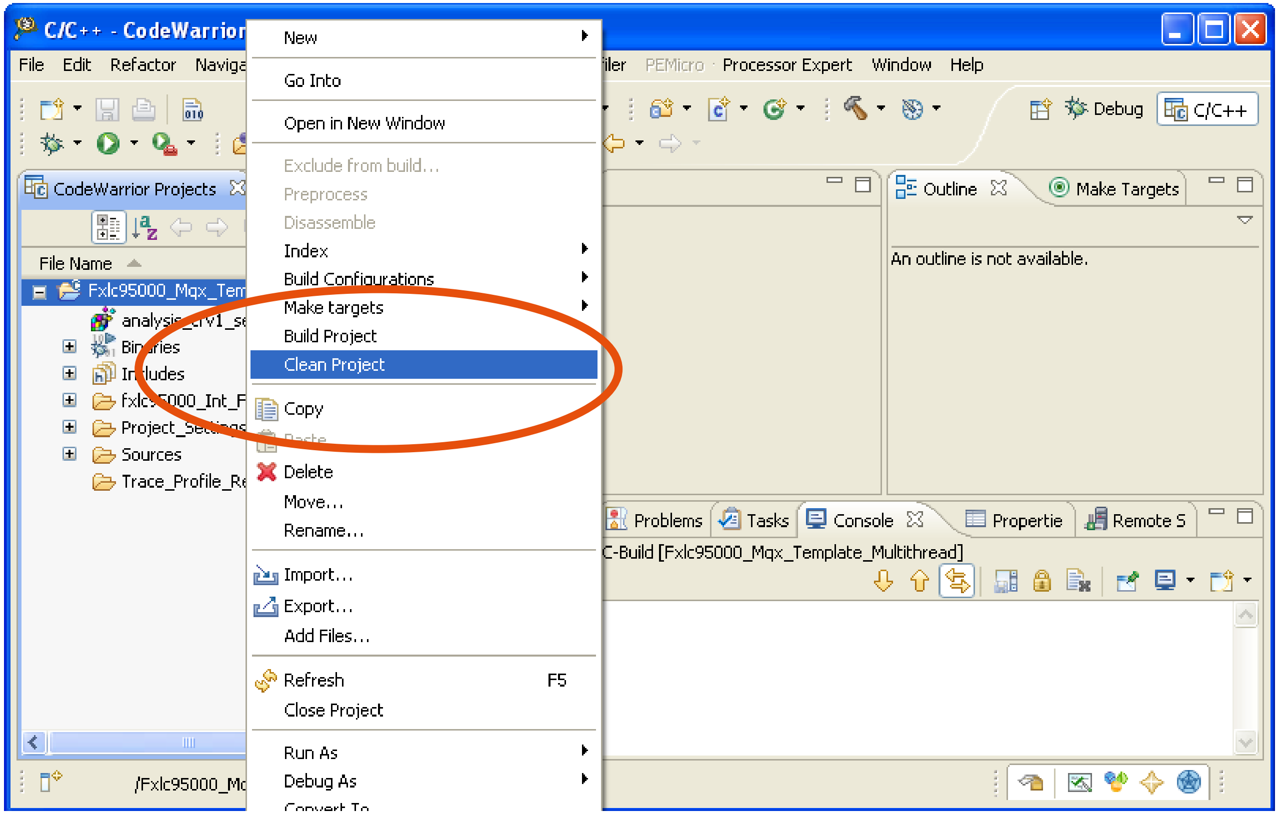


Lab1: Import Finish



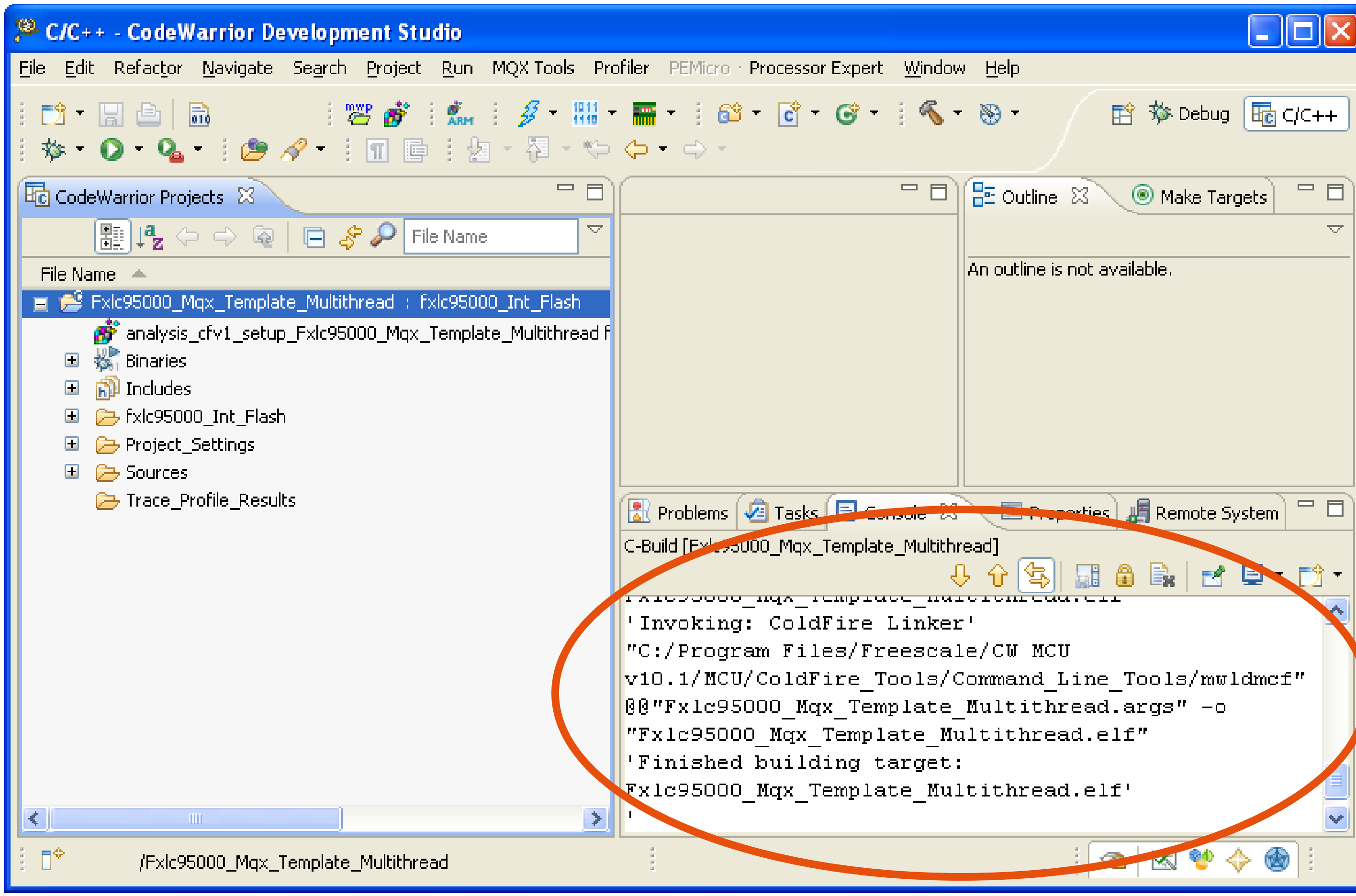
Lab1: Clean and Build Project

- Right click on the project name,
  - select **Clean Project** in the draw down menu
  - Then select **Build Project**



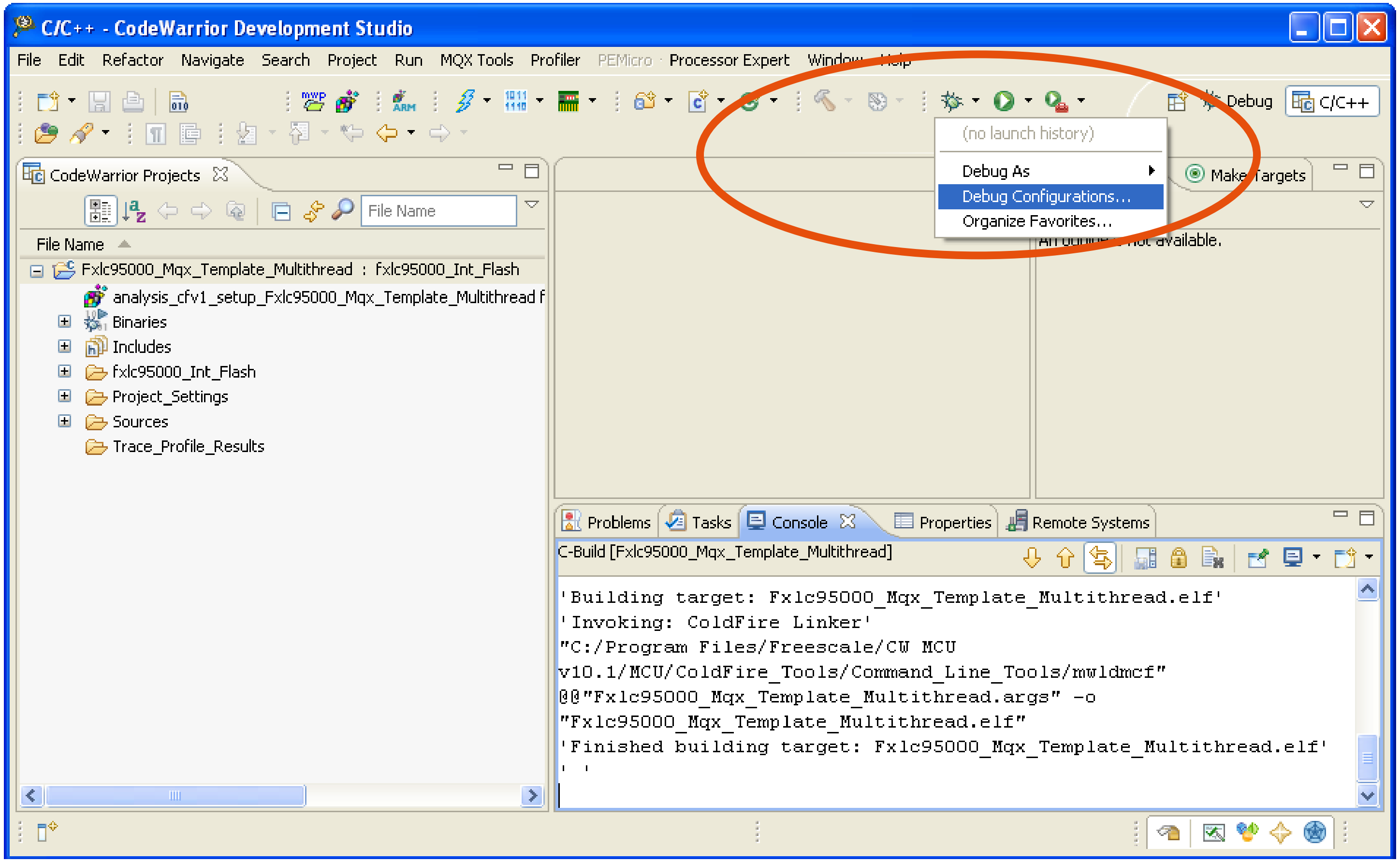
Lab1: Project Clean/Built Status

- Default C++ perspective bottom window → **Console**



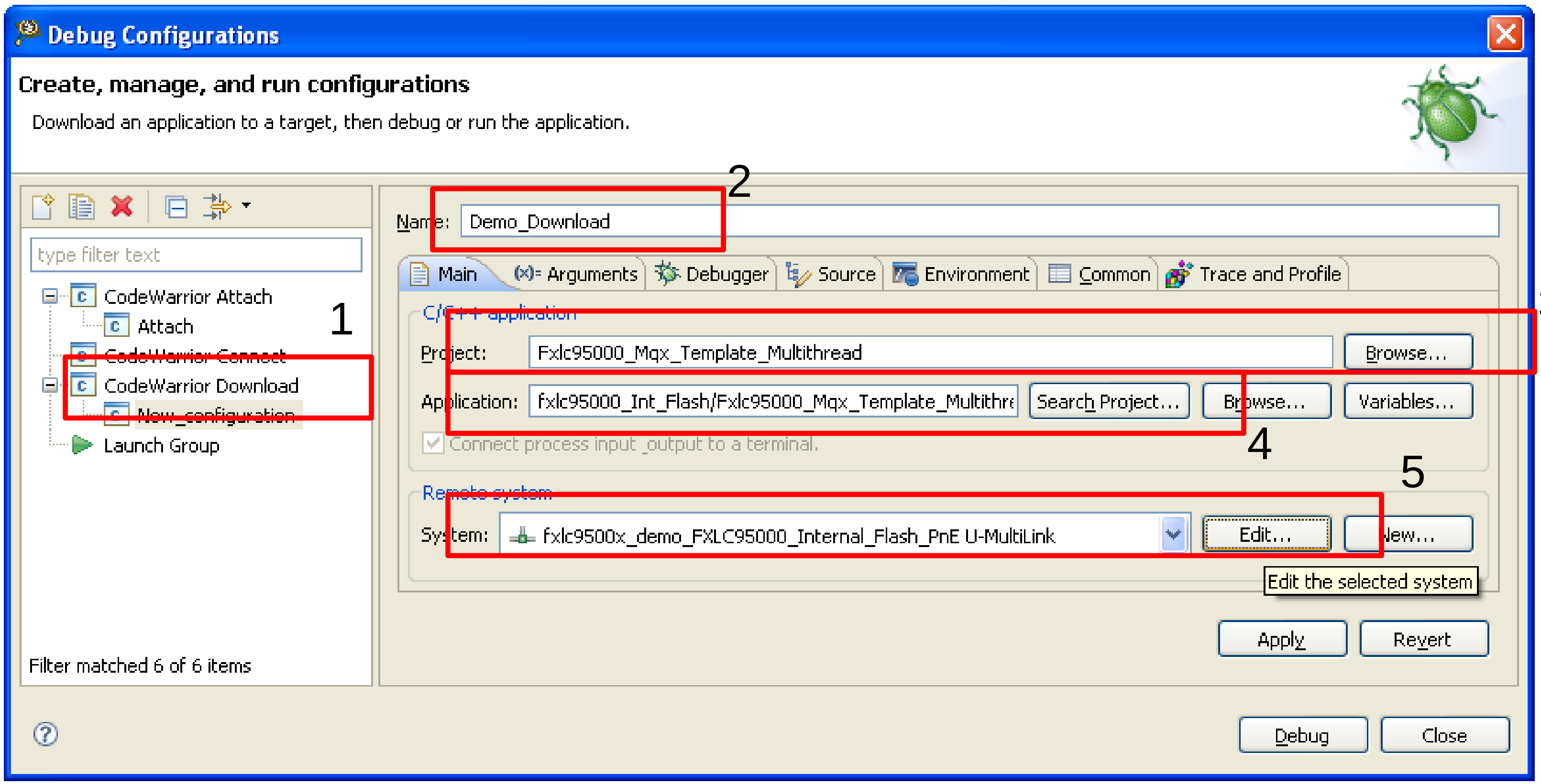
Lab1: Setup Download Options

- Find the **bug** icon on Tool bar. Click the arrow to find **Debug Configuration**



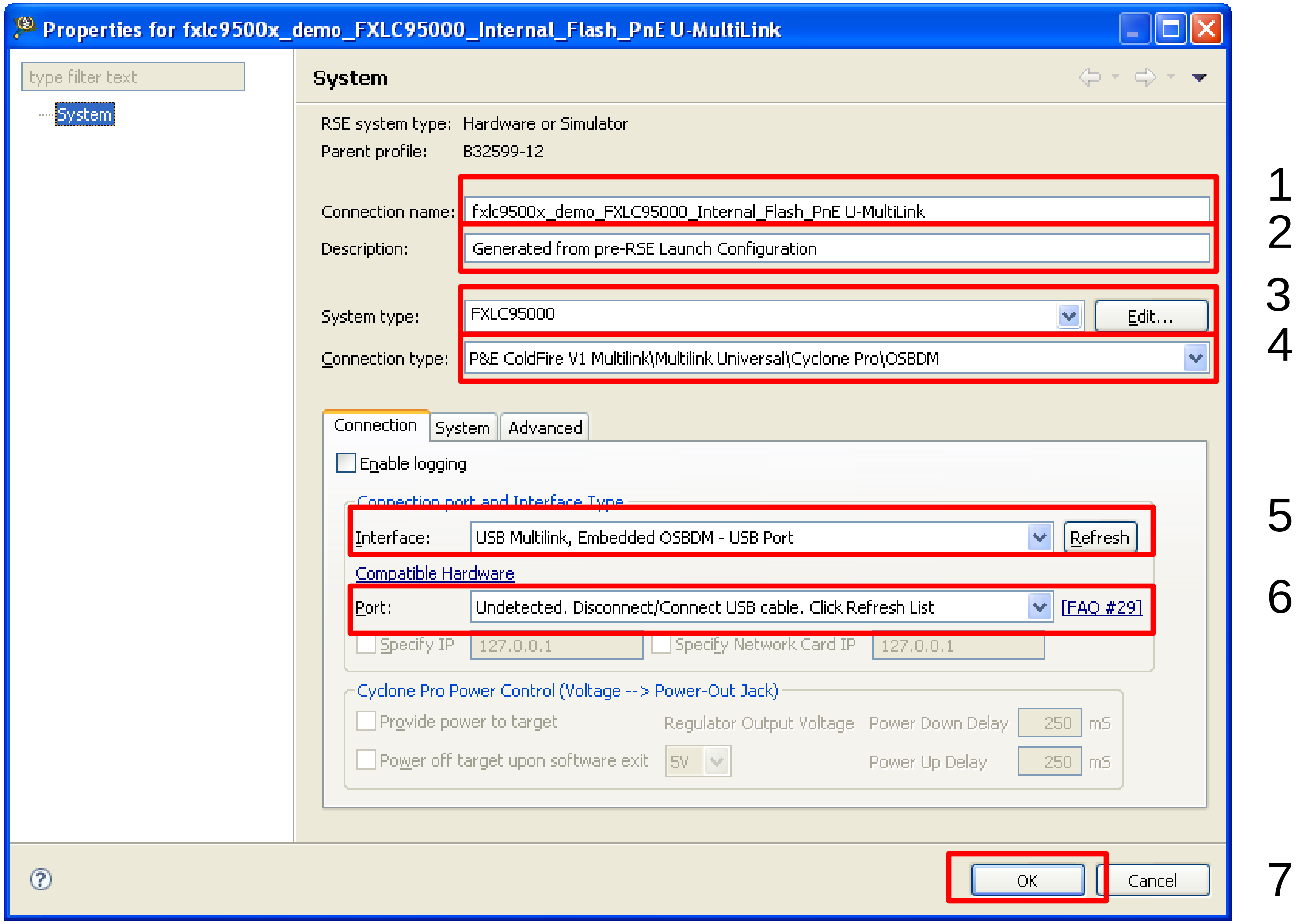
Lab1: Setup Download Options

- **Double click** the CodeWarrior Download to create a new profile
- Change the right side window accordingly



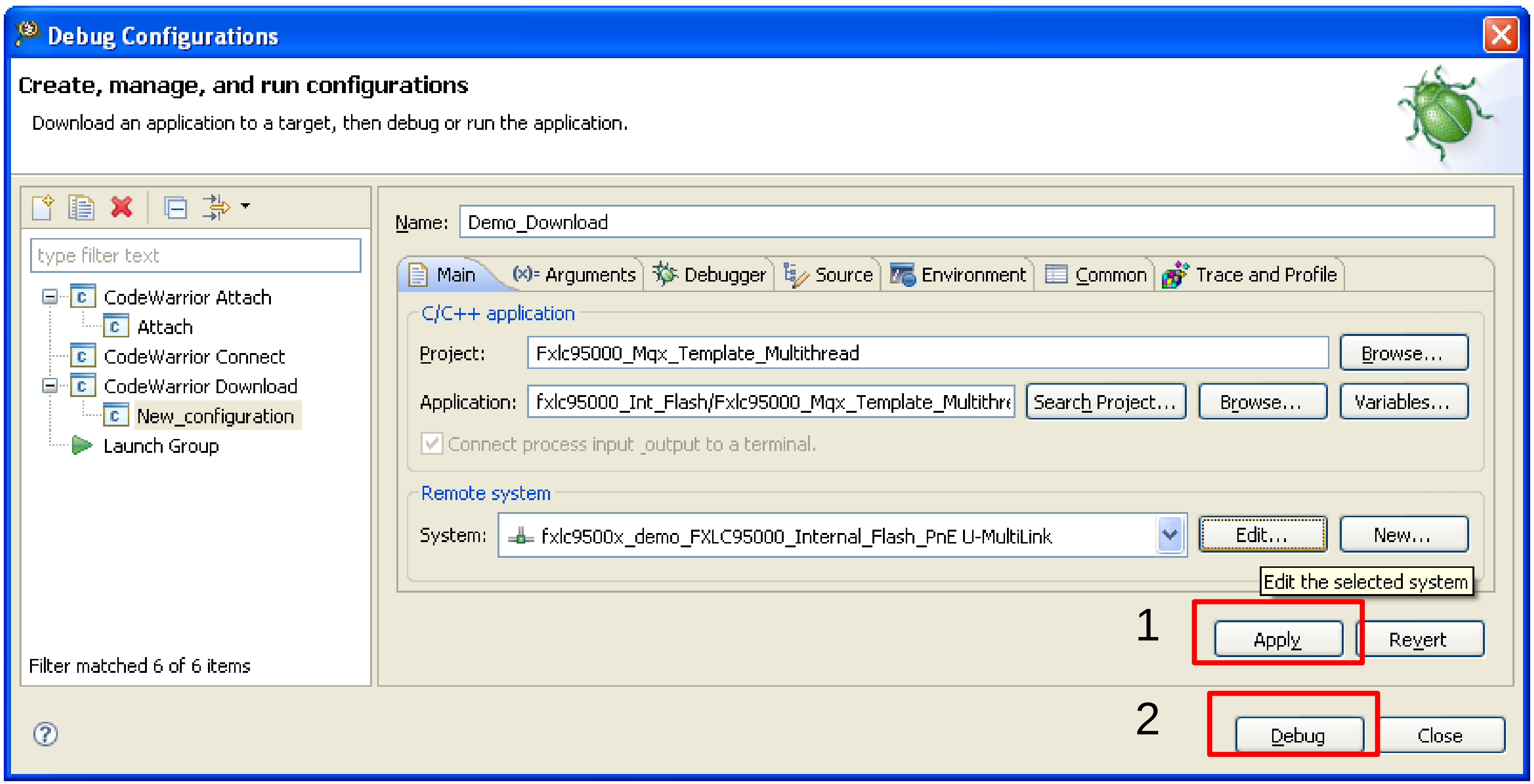
Lab1: Setup Download Options

- **Fill the system setting as below**



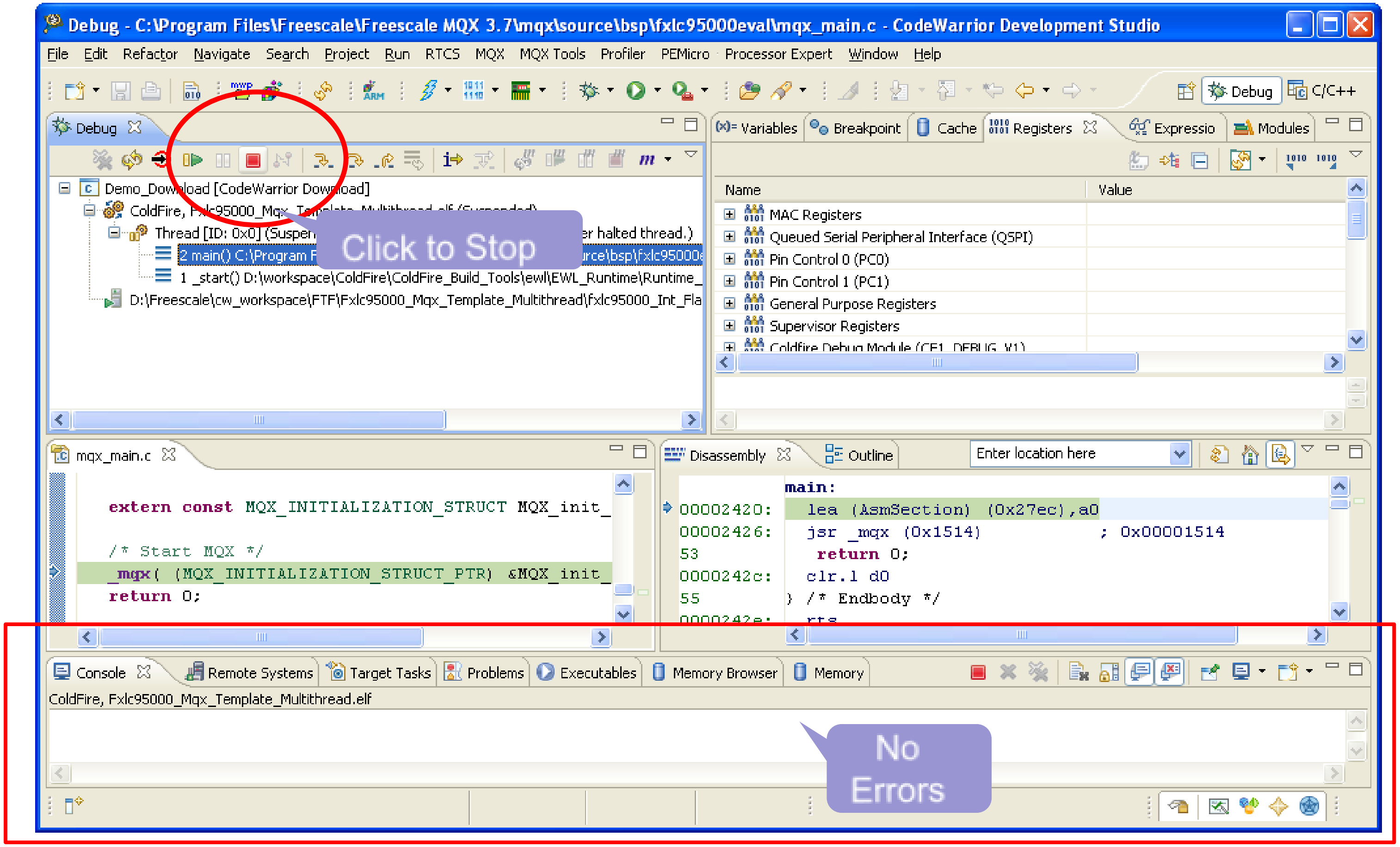
Lab1: Setup Download Options

- Click on **Apply** to adapt all changes
- Click **Debug** to start to download the program to FLASH



Lab1: Download Finish

- When the FLASH is programmed, the Debug window shows
- Click red square button to finish the download



Lab1: Setup (15 minutes)

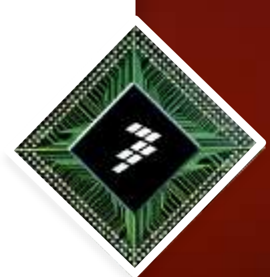
- **All LEDs (LED5-8) refresh together every 1/4 sec when**
  - programmed correctly
  - AND SW1 button is at ON position

Refer to slides: 29- 44

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

MQX Based Coding Structure

FTF-ENT-F0091

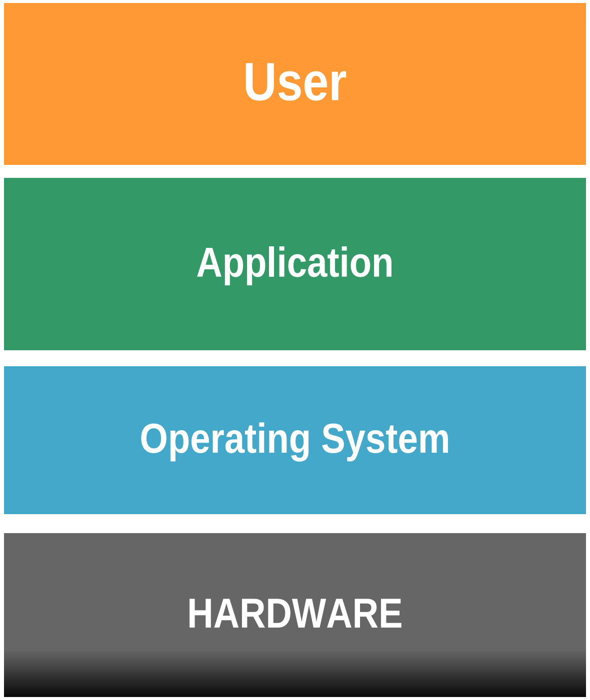


Agenda

- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

Operating Systems

- **The term "operating system" can be used to describe the collection of software that manages a system's hardware resources**
- **This software might include a file system module, a GUI and other components**
- **Often times, a "kernel" is understood to be a subset of such a collection**
- **Characteristics**
  - Resource management
  - Interface between application and hardware
  - Library of functions for the application



Real Time Operating Systems

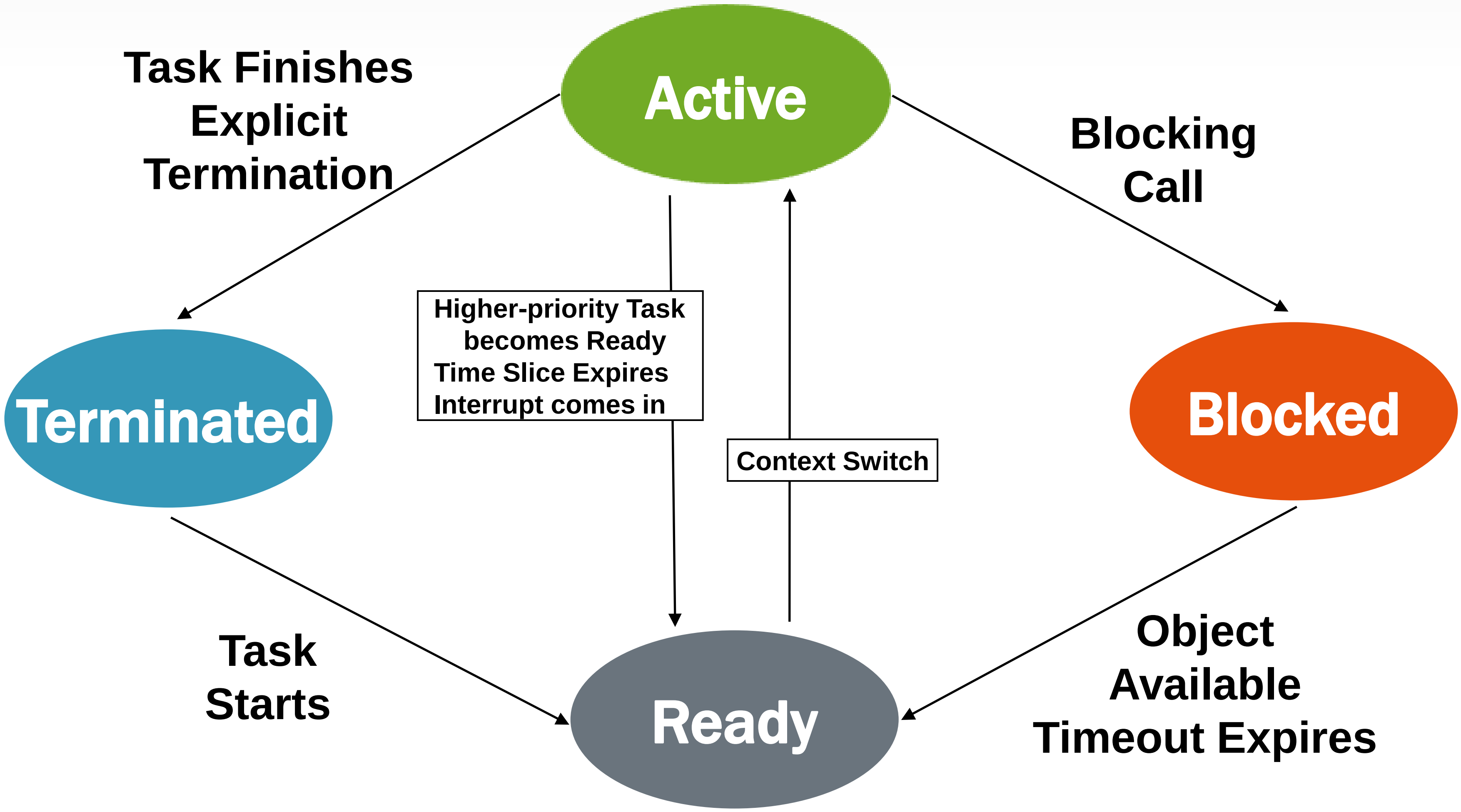
- **A real-time operating system (RTOS) manages the time of a microprocessor or microcontroller**

- **Features of an RTOS:**
  - Allows multi-tasking
  - Scheduling of the tasks with priorities
  - Synchronization of the resource access
  - Inter-task communication
  - Time predictable
  - Interrupt handling

MQX Task States

- **A task is in one of these logical states:**
  - blocked
    - the task is blocked and therefore not ready
    - it’s waiting for a condition to be true
  - active
    - the task is ready and is running because it’s the highest-priority ready task
  - ready
    - the task is ready, but it’s not running because it isn’t the highest-priority ready task
  - terminated
    - the task has finished all its work, or was explicitly destroyed

Task State Diagram



MQX - Task Management Example

{INIT_TASK, init_task, 100, 11, "init", MQX_AUTO_START_TASK, 0, 0},

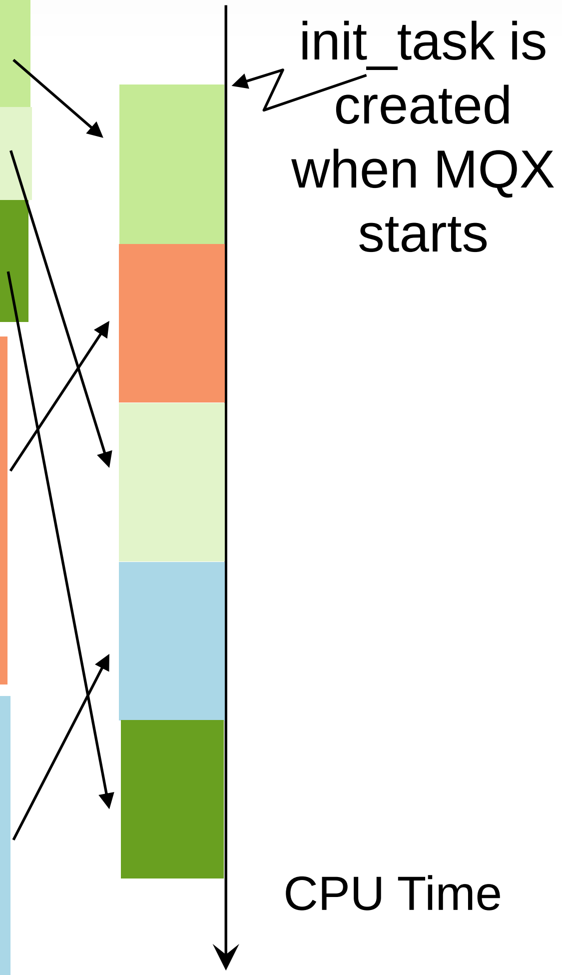
```
void init_task(void){
_task_create(0,TASK_A,0);
  ...
  _task_ready(Task_B);
  …
}
```

{TASK_A, Task_A, 100, 10, “Task A", 0, 0, 0},

```
void Task_A(void)
{
     ...
 _task_create_blocked(0,TASK_B,0);
 …
 _task_abort(TASK_A);
}
```

{TASK_B, Task_B, 100, 9, “Task B", 0, 0, 0},

```
void Task_B(void)
{

   ...
   _task_abort(TASK_B);
}
```



FXLC95000L Code Structure Quick View

1. Build the MQX task list

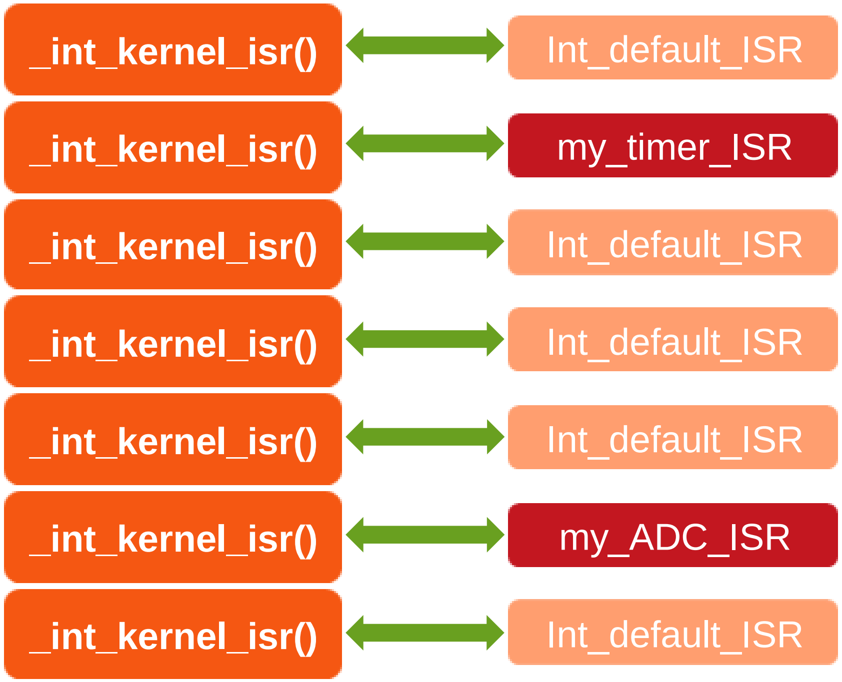
```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
/*  Task number, Entry point, Stack, Pri, String, Auto? */
   {MAIN_TASK,   Main_task,   512,   7,   "main", MQX_AUTO_START_TASK},
   {AFE_TASK,    Afe_task,    1024,  8,   “afe", MQX_AUTO_START_TASK},
   {SP_TASK,     Sp_task,     512,   10,  “sp", MQX_AUTO_START_TASK},
   {TPM_TASK,    Tpm_task,    512,   9,   “tpm", MQX_AUTO_START_TASK},
……

   {0,           0,           0,     0,   0,      0,                 }
};
```

MQX Interrupts

- Embedded systems are based on ISR
- Usually an ISR is used for signal an event
- The most common actions on an ISR are:
  - Post a semaphore
  - Send a message
  - Set an event
  - Clear an error condition
- **Important**: ISRs are not tasks
- **Remember:** ISR should be short and must not use blocking functions

MQX Interrupts

- **MQX kernel ISR manages all the user interruptions**
- **MQX kernel ISR performs:**
  - Context saving of the current task
  - Switches to interrupt stack
  - Calls the user ISR
  - Passes a parameter to the ISR
  - Restores the context of the next task to run (depends on the scheduler)



FXLC95000L Interrupt EVENT set up

2. Link the hardware interrupts to ISRs in initialization (use _init_install_isr)

```
// Install isr for Timer Overflow

 if (_int_install_isr(VectorNumber_Vtpm1ovf, (void
   (_CODE_PTR_)(pointer))tpm_isr, NULL) == NULL)
    return;
```

FXLC95000L Interrupt EVENT set up

3. Link the hardware interrupts to an EVENT variable

```
void tpm_isr(void *pDummy)
{
  TPMSC_TOF = 0;            // Clear interrupt source
  _lwevent_set(&gEvent_irq, EVENT_IRQ_TPM_OVERFLOW);
}
```

FXLC95000L Interrupt EVENT set up

4. Configure Event Tasks to be blocked until

- **Initialization is completed**
- **EVENT is true**

```
void Afe_task(uint32 initial_data)
{
    // Wait for system init to be done.
    _lwevent_wait_for(&gEvent_sys, EVENT_SYS_INIT_DONE, FALSE, 0);

    while(1)
    {
      // Block until AFE conversion is completed.
      _lwevent_wait_for(&gEvent_irq, EVENT_IRQ_AFE_CONV_COMPLETE, FALSE, 0);

      …….
    }
}
```

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000L Module: GPIO and TPM <Lab 2>

FTF-ENT-F0091

Agenda

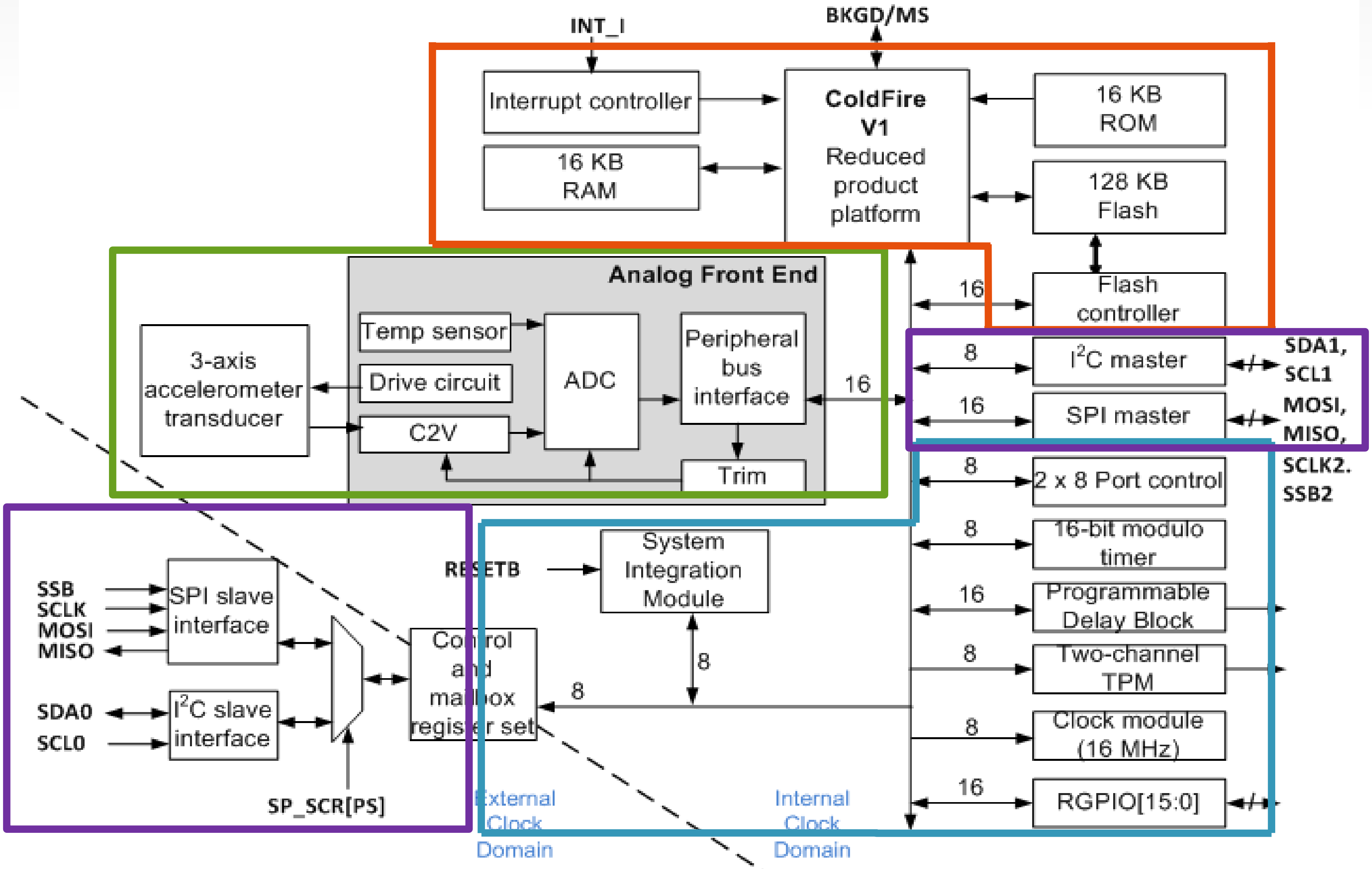
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

FXLC95000L Module: GPIO and TPM

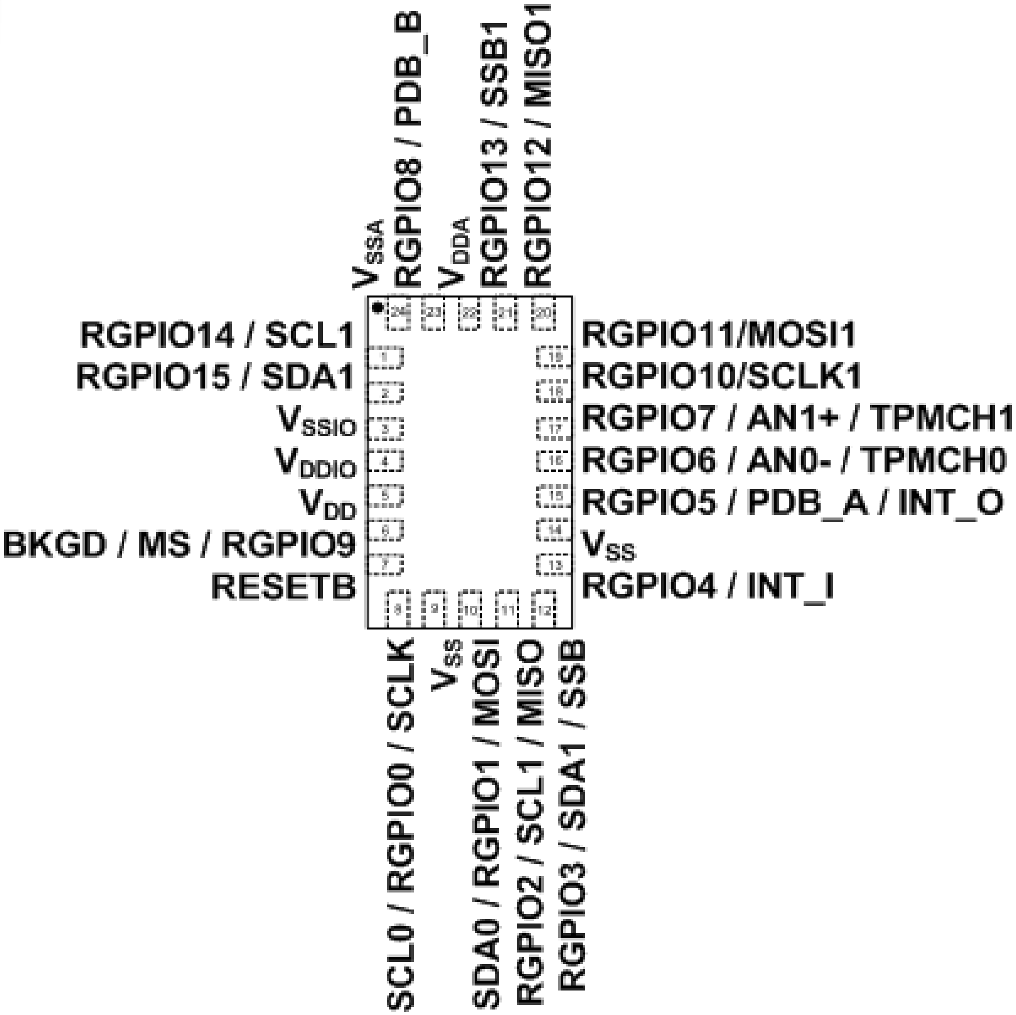
- **Object:**

1. Introduce the RGPIO peripheral functions
2. Configure the pins to be RGPIO inputs / outputs
3. Read / Write RGPIO pins
4. Lab:
   - Observe LED7-10 toggles when SW1 is ON
   - Toggle LED8 by Switches SW2 when SW1 and SW4 are OFF
   - Toggle LED9 by Switches SW3 when SW4 and SW4 are OFF

FXLC95000L Block Diagram



GPIO pin functionalities needs to selected



RGPIO	Default Functionality
0	SCL0
1	SDA0
2 – 9	RGPIO
10 -15	Auto Selected

Configure GPIO Pins

- **RGPIO 0-9**
  - Configure the pin functiona
    PMCR0 / PMCR1 / PMCR

PMCR0 SIM_BASE + 0xA

	7	6	5	4	3	2	1	0
R/W	A9	A8	A7		A6		RESERVED	A4
Reset	0	0	0	0	0	0	0	0

Figure 14-16 SIM Pin Mux Control Register 0

- **RGPIO 10-15**
  - Automatic pin selection be

PMCR1 SIM_BASE + 0xB

	7	6	5	4	3	2	1	0
R/W	A3		A2		A1		A0	
Reset	0	0	0	0	0	0	0	0

Figure 14-17 SIM Pin Mux Control Register 1

PMCR2 SIM_BASE + 0xC

	7	6	5	4	3	2	1	0
R/W	RESERVED						A5	
Reset	0	0	0	0	0	0	0	0

Figure 14-18 SIM Pin Mux Control Register 2

FXLC95000L RGPIO Configurations and Functions

- **The FXLC95000L RGPIOs can be configured as below**
  - RGPIO_DIR: *Configures the corresponding RGPIO pin as inputs or outputs*
  - RGPIO_ENB: *Enables the corresponding device pins as RGPIOs*

- **Write / Read with below**
  - RGPIO_CLR: *Clears the corresponding RGPIO pin*
  - RGPIO_SET: *Sets the corresponding RGPIO pin*
  - RGPIO_TOG: *Toggles the corresponding RGPIO pin*
  - RGPIO_DATA: *Specifies the write data for the corresponding RGPIO output pin or the read data value for the corresponding RGPIO input pin*

Configure the RGPIO and Use its Functions

- LED / Switch connected pins are mapped in header file

  demo_io.h

- Read from RGPIO to get switch settings using this code

  gpio_in = RGPIO_DATA;

- Write to RGPIO to change LED using this code

Toggle LED1:	RGPIO_TOG = (uint16) LED1;
Toggle LED1 and LED3:	RGPIO_TOG = (uint16) LED1 | LED2;
Turn on LED1:	RGPIO_CLR = (uint16) (~LED1);
Turn off LED1:	RGPIO_SET = (uint16) LED1;

TPM Timer

- FXLC95000L has internal TPM (Timer/PWM) module
  - 16 bit timer
  - TPMSC : External and Internal clock (~MQX timer)
  - TPMMOD : set the period by setting up
  - Hardware interrupts: including TPM Timer overflow

- Use Timer to schedule the switch read and LED writes
  - ¼ sec period
  - Internal clock

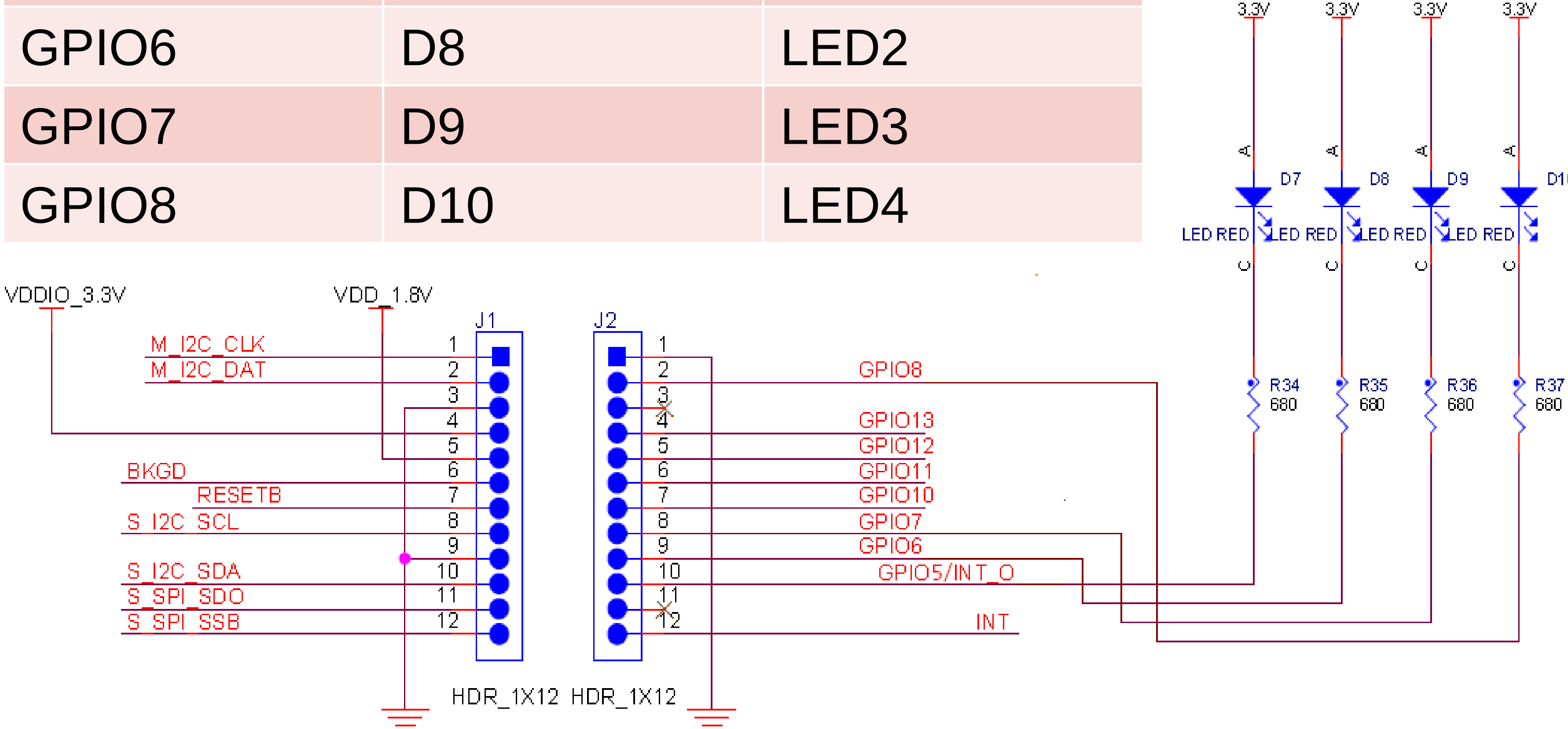
Write Code in Event Tasks

- Find the Event tasks to plug in code

```
void Tpm_task(uint32 initial_data)
{
    uint16 gpio_in = 0;

    // Wait for system init to be done.
    _lwevent_wait_for(&gEvent_sys, EVENT_SYS_INIT_DONE, FALSE, 0);

    while(1)
    {
      // Block until timer interrupt event occurs.
      _lwevent_wait_for(&gEvent_irq, EVENT_IRQ_TPM_OVERFLOW, FALSE, 0);

            < plug in your code >
    }
}
```

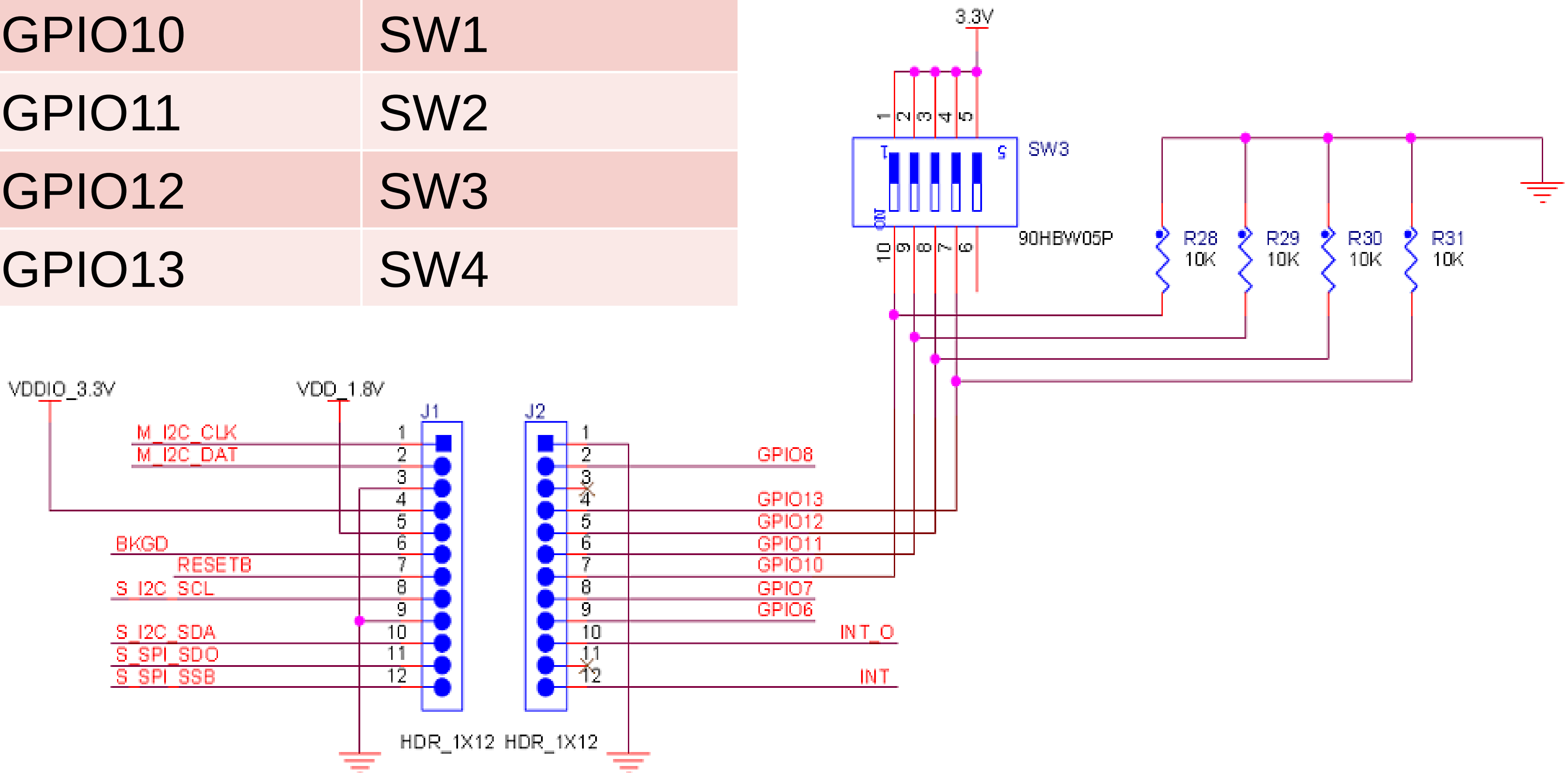
LEDs Connected to RGPIOs

RGPIO pin	LED on Board	LED in code
GPIO5	D7	LED1
GPIO6	D8	LED2
GPIO7	D9	LED3
GPIO8	D10	LED4



Switches Connected to RGPIOs

RGPIO pin	LEDs
GPIO10	SW1
GPIO11	SW2
GPIO12	SW3
GPIO13	SW4

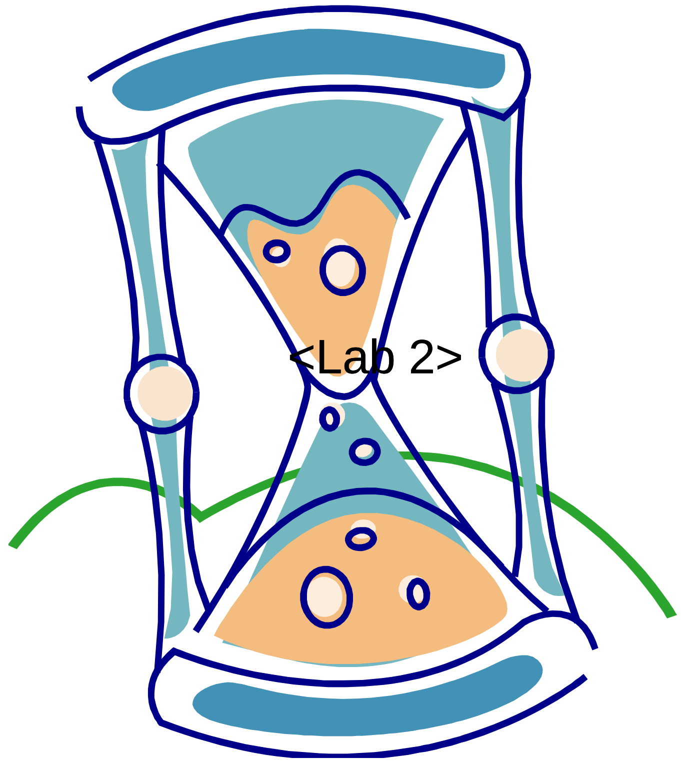


Lab 2: Tasks (15 minutes)

- **Find demo_io.h**
- **Find TPM_task()**
- **Find LED update section**
- **Program and Observe**
  - LED7-10 toggles when SW1 is ON

Refer to slides: 59- 70

10 minutes Class Break

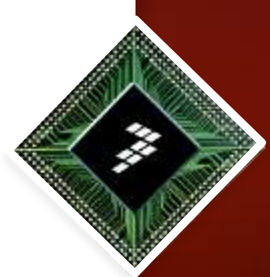


<Lab 2>

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000 Module: AFE

FTF-ENT-F0091



Agenda

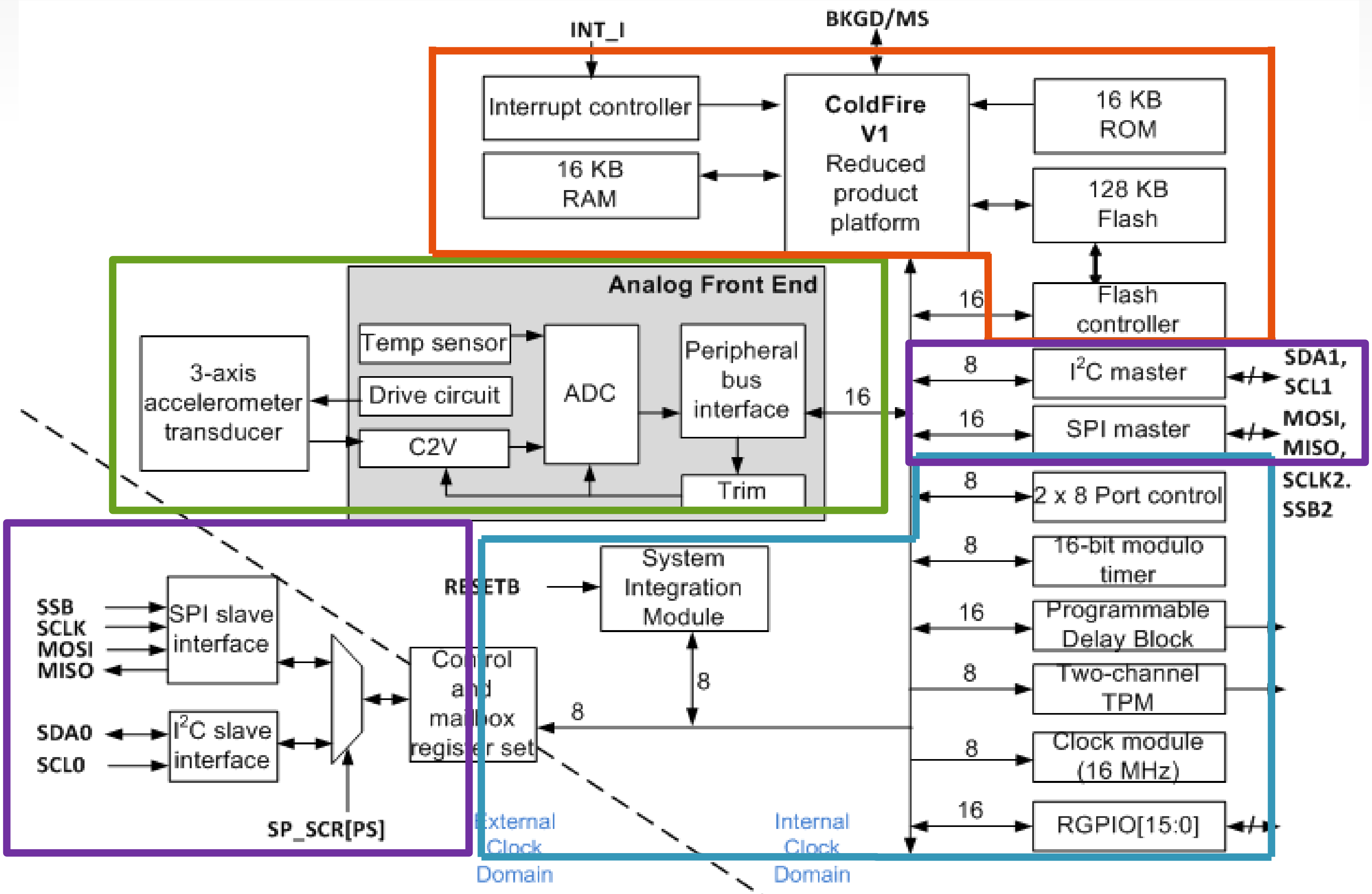
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

FXLC95000 Module: AFE

- **Object:**

  1. Introduce the Accelerometer’s general setting, reading and data usage
  2. Setup Accelerometer and acquire data
  3. Lab:
     - Set accelerometer sampling bits, rate, g-range
       - Sampling bit = 16 bits
       - Sampling range = 2g
       - Sampling rate = 1Hz
     - Toggle LED4 (D10) when a new accelerometer data is acquired

FXLC95000L Block Diagram



AFE = Analog Front End Module

- Resolution: 16, 14, 12, 10 and 8-bit

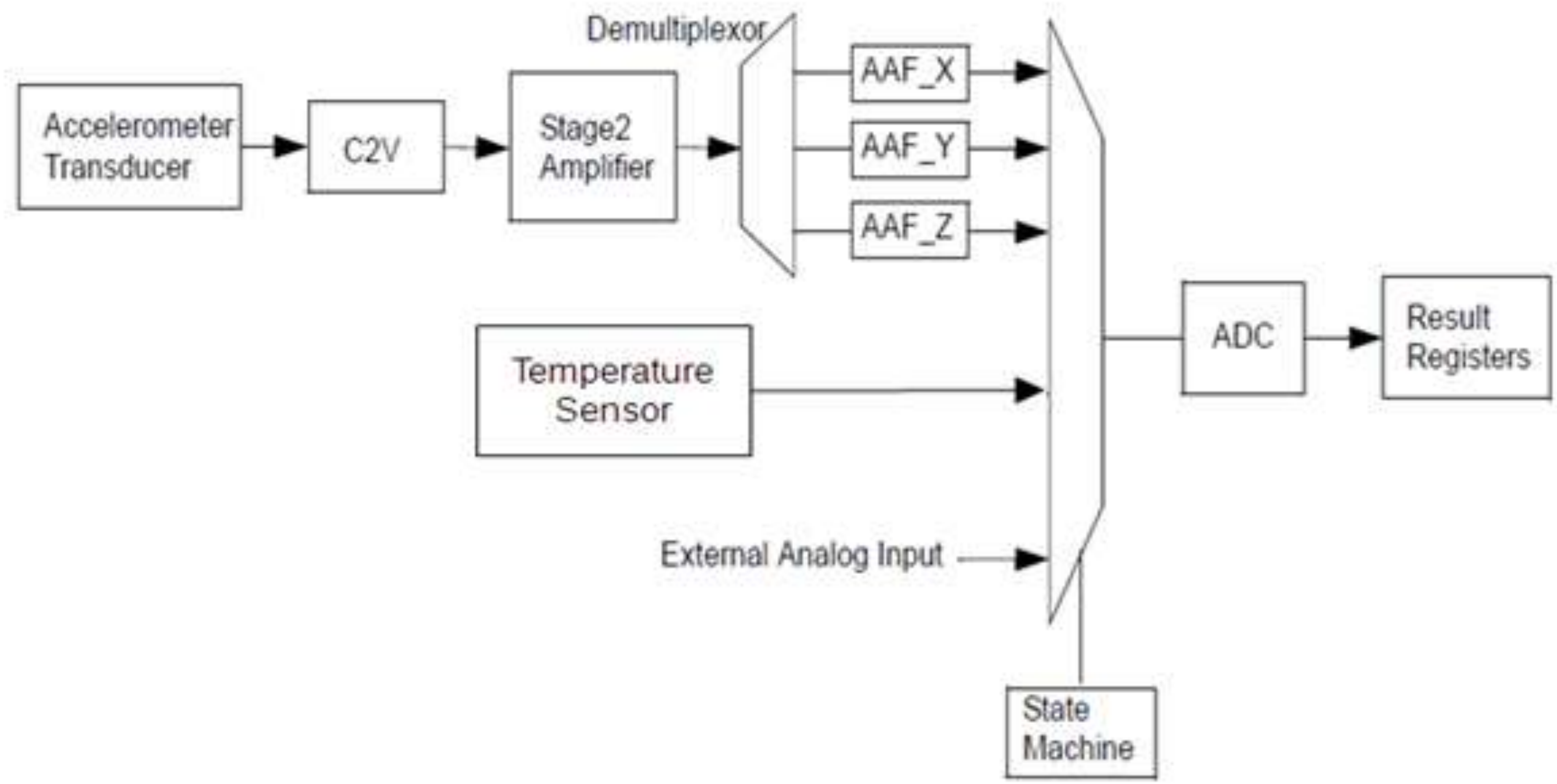
- g-range: +/-2g, +/-4g, and +/-8g

  $1g = 9.8m/s^2$

- Inputs:
  - Acceleration
  - Temperature
  - External analog input

- Sampling Rate: 0.24 – 7.8KHz in steps

Analog Frontend Block Diagram



AFE Setting

- **AFE_CSR (AFE Control & Status) register**
  - provides the control interface and status of the AFE subsystem
  - Set resolution
  - Set full scale g-range

- **Freescale provides a C-based function that abstracts away the intricacies of configuring AFE and converting the accelerometer readings to acceleration.**

```
fxlc9500x_afe_csr_set((afe_csr_options_t)
    (AFE_CSR_GRANGE_2G | AFE_CSR_C4MODE_NONE |
    AFE_CSR_CMODE_16BIT));

fxlc9500x_framerate_set(FRAMERATE_30HZ);
```

AFE Sampling Rate → "FIC"?

- FIC = Frame Interval Counter

- Automatically initiate a software triggered activity:
  - Toggling a GPIO
  - Performing an AFE acquisition
  - Etc.

- Can be configured to generate an interrupt
  - whenever the user programmed counter limit is reached

- Clocked by the FXLC95000L on-chip oscillator
  - 16 MHz / 62.5 KHz

Frame Interval Rates

- The FLE value is set in the **CK_OSCTRL – (Oscillator Control Register)**
  - Provide the system clock setting
  - Provide the AFE sampling rate setting

FLE	2^{FLE}	t_F (secs)	Max Frames per Second	Max Fast Clock Cycles per Frame
4	16	1.28E-4	7812.5	2,048
5	32	2.56E-04	3906.25	4,096
6	64	5.12E-04	1953.13	8,192
7	128	1.02E-03	976.56	16,384
8	256	2.05E-03	488.28	32,768
9	512	4.10E-03	244.14	65,536
10	1024	8.19E-03	122.07	131,072
11	2048	1.64E-02	61.04	262,144
12	4096	3.28E-02	30.52	524,288
13	8192	6.55E-02	15.26	1,048,576
14	16384	1.31E-01	7.63	2,097,152
15	32768	2.62E-01	3.81	4,194,304
16	65536	5.24E-01	1.91	8,388,608
17	131072	1.05E+00	0.95	16,777,216
18	262144	2.10E+00	0.48	33,554,432
19	524288	4.19E+00	0.24	67,108,864

AFE Acceleration Data Reading

- **The acceleration readings in the AFE registers are represented as raw ADC counts and they need to be processed**
  - Scaled to align with the user configured g range
  - Corrected to provide accurate reading

```
fxlc9500x_afe_trimmed_sensor_data_get(afe_data.data);
```

Understand the Acceleration Data

- **The post alignment acceleration readings can converted to g's (g)**

- **Example: Convert 16 bit acceleration data to g's**

  16 bit data counts = -32768 to 32767 (65535)

  g-acceleration = Acceleration Reading * (sensitivity)

Full scale	**Counts / g (16 bit data)**	**Sensitivity (ug/ count)**
2g	16384	61
4g	8192	122
8g	4096	244

Lab 3: Task (15 minutes)

- **Find afe_task ()**
  - This task is triggered when new AFE data is ready
- **Find tpm_task()**

- **Program**
  - **Uncomment the AFE data reading code**
  - **Program:**
    - **Read X, Y, Z axis data into local variables**
    - **Set flag = 0 at the end of reading to toggle LED for indication**

Refer to slides: 73- 82

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000 Module: AFE Motion Detection

FTF-ENT-F0091

Agenda

- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

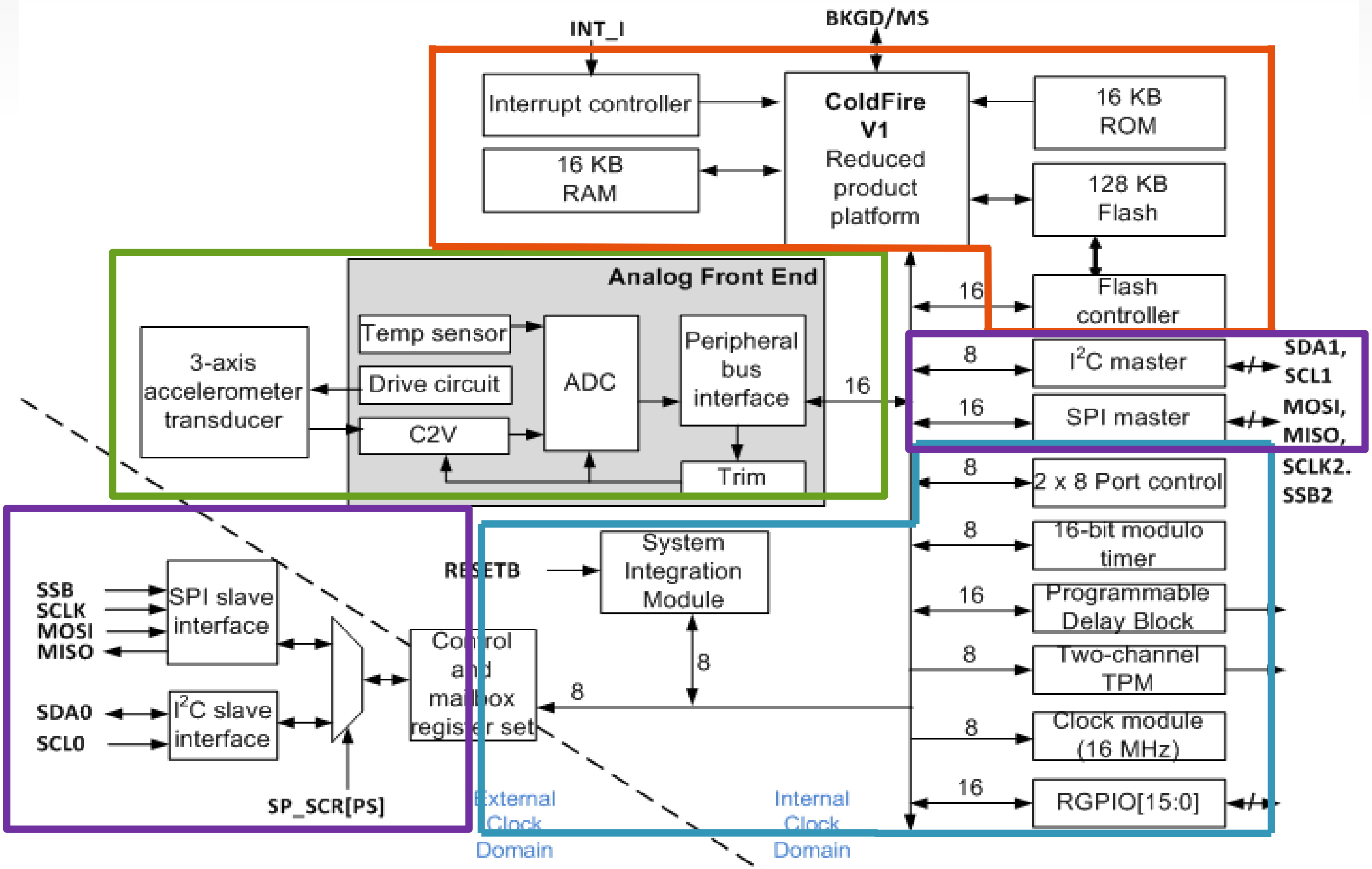
freescale™

AFE Motion Detection

- **Object:**

1. Introduce the understand of motions and use g-acceleration reading to identify them
2. Use the lab boards and set up the following functions
   - Freefall
   - Left / Right / Top / Bottom bump
   - Observe LEDs to toggle with movements

FXLC95000L Block Diagram

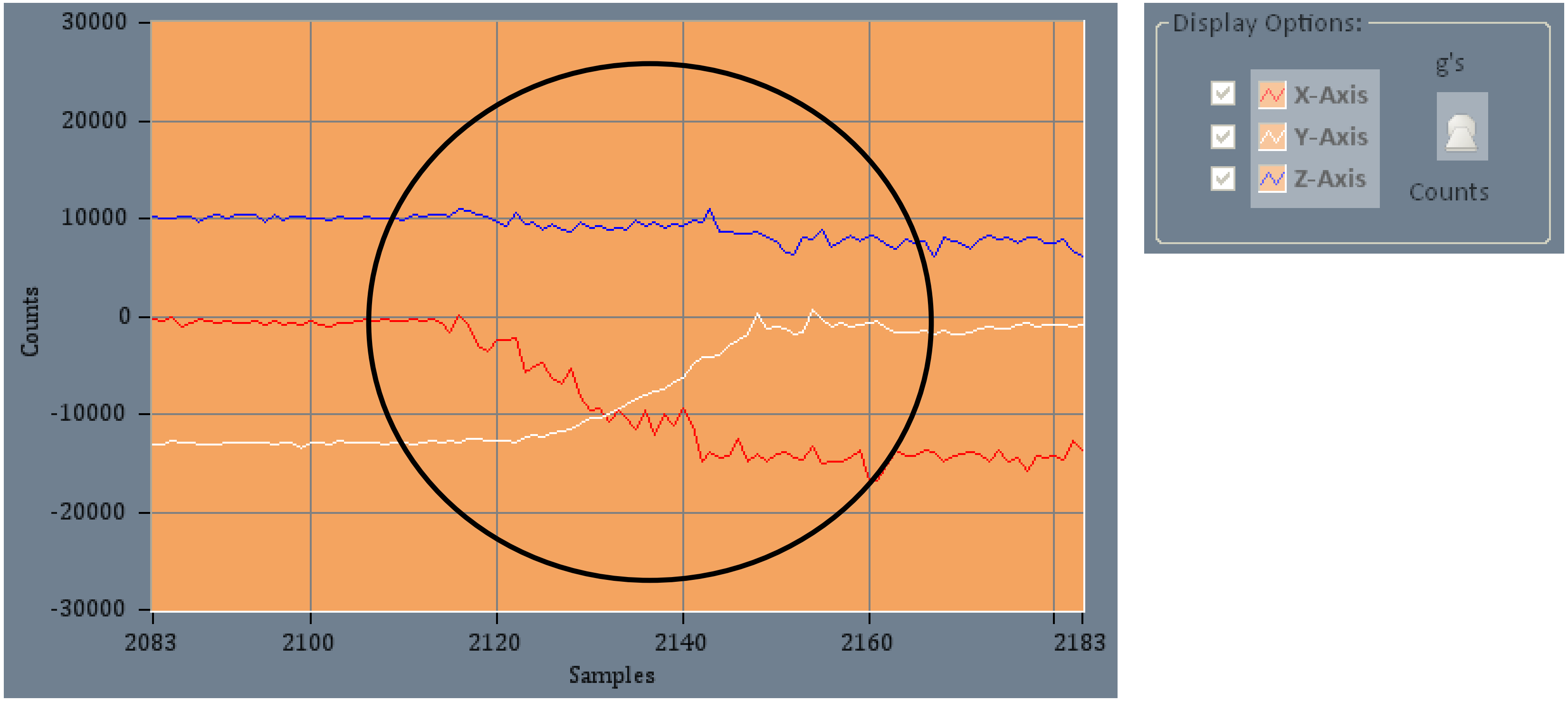


Motions & their Signatures

- **Motion movements can be identify with acceleration data**
  - Portrait / Landscape Orientation
  - Tilt
  - Freefall
  - Tap / Double tap
  - Shake
  - Etc.

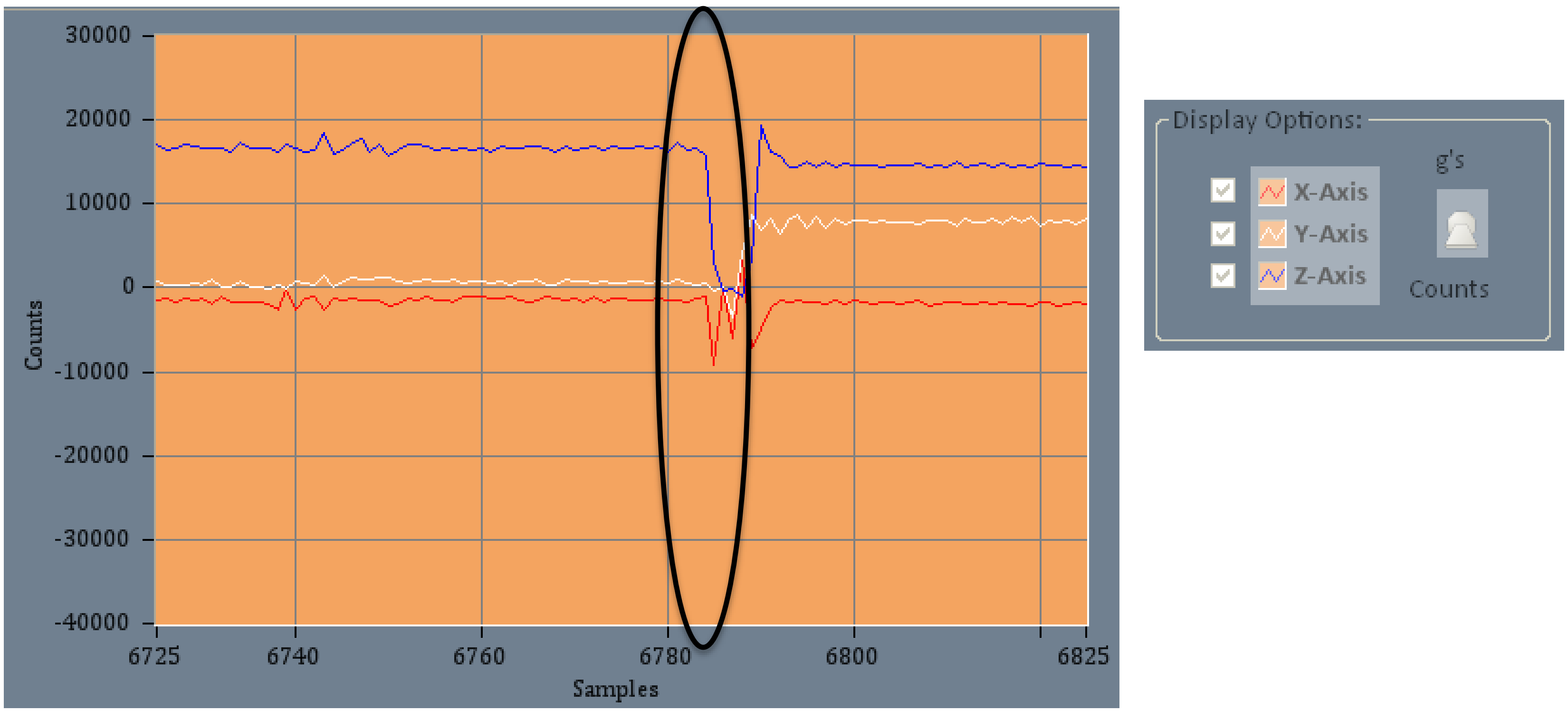
Example 1: Orientation Change Graphics

- **A Orientation Change (on X/Y) creates**
  - An alternative g-acceleration on X and Y axis



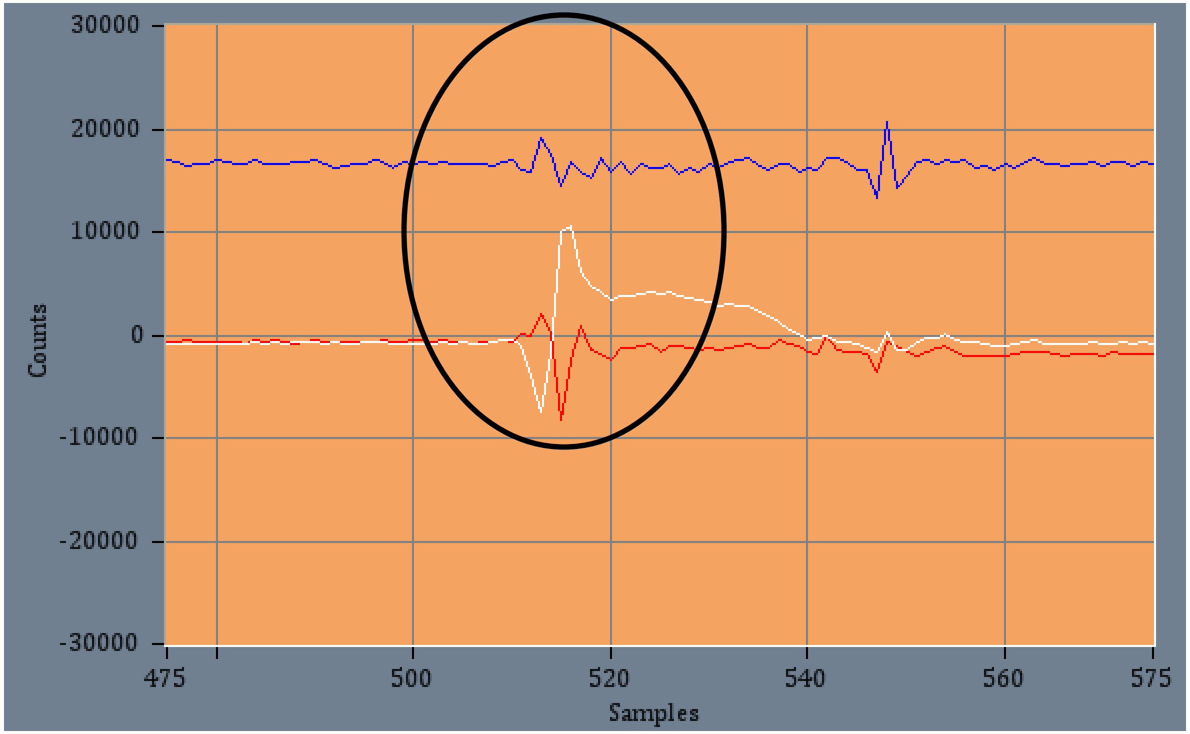
Example 2: Freefall

- **A Freefall creates**
  - a very low g on all X, Y, Z axis

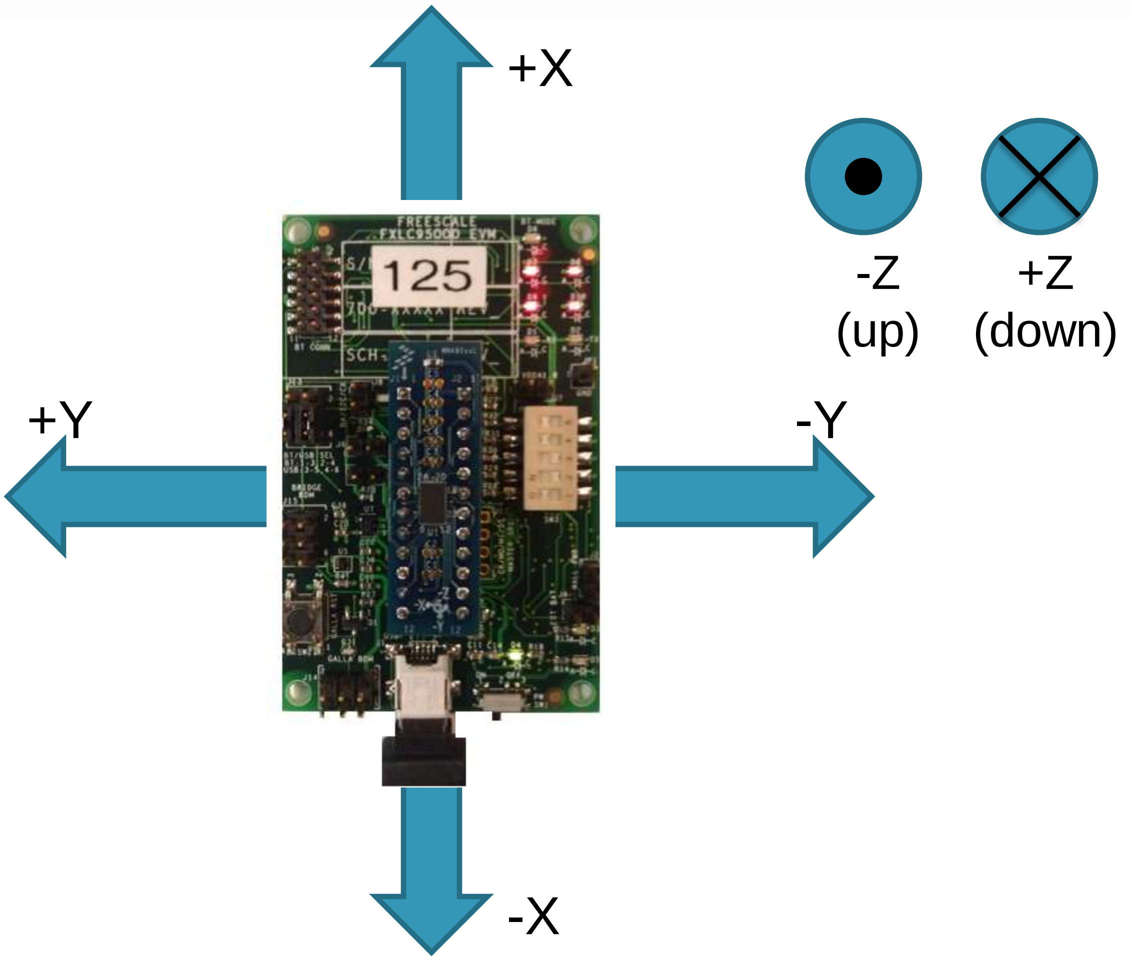


Example3: Jolt on Y axis

- **A light jolt on Y axis creates**
  - a medium g on Y axis
  - X, Z g-acceleration remains the same



XYZ Axes Direction on Programming Board



Lab 3: Tasks

- **Find afe_task ()**
  - This task is triggered when new AFE data is ready
- **Find tpm_task()**

- **Use below variables:**
  - AFE data : X axis – afe_data.data[0]
    Y axis – afe_data.data[1]
    Z axis – afe_data.data[2]
  - Flag: int flag

- **Program**
  - **Convert the threshold g-values to accelerometer counts**
  - **Compare the acceleration readings with the thresholds to Identify the motions**
  - **Output results on LEDs**

Lab3: Identify these Motions and Output w/ LEDs (15 minutes)

- **Identify motions, refer to code example**
- **Set flag**
- **Toggle LED as table below using the existing code**

Refer to slides: 85- 94

Gesture	LED display				Event flag	Detection Hold time
	D7	D8	D9	D10		
Board slide to left	ON	OFF	ON	OFF	1	1 sec
Board slide to right	OFF	ON	OFF	ON	2	1 sec
Board slide to upper	ON	ON	OFF	OFF	3	1 sec
Board slide to lower	OFF	OFF	ON	ON	4	1 sec
Board freefall	OFF	OFF	OFF	OFF	5	1 sec
Board flat on board	ON	ON	ON	ON	0	0
Other	ON	OFF	ON	OFF	10	0

FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000 Module: I2C Slave

FTF-ENT-F0091

Agenda

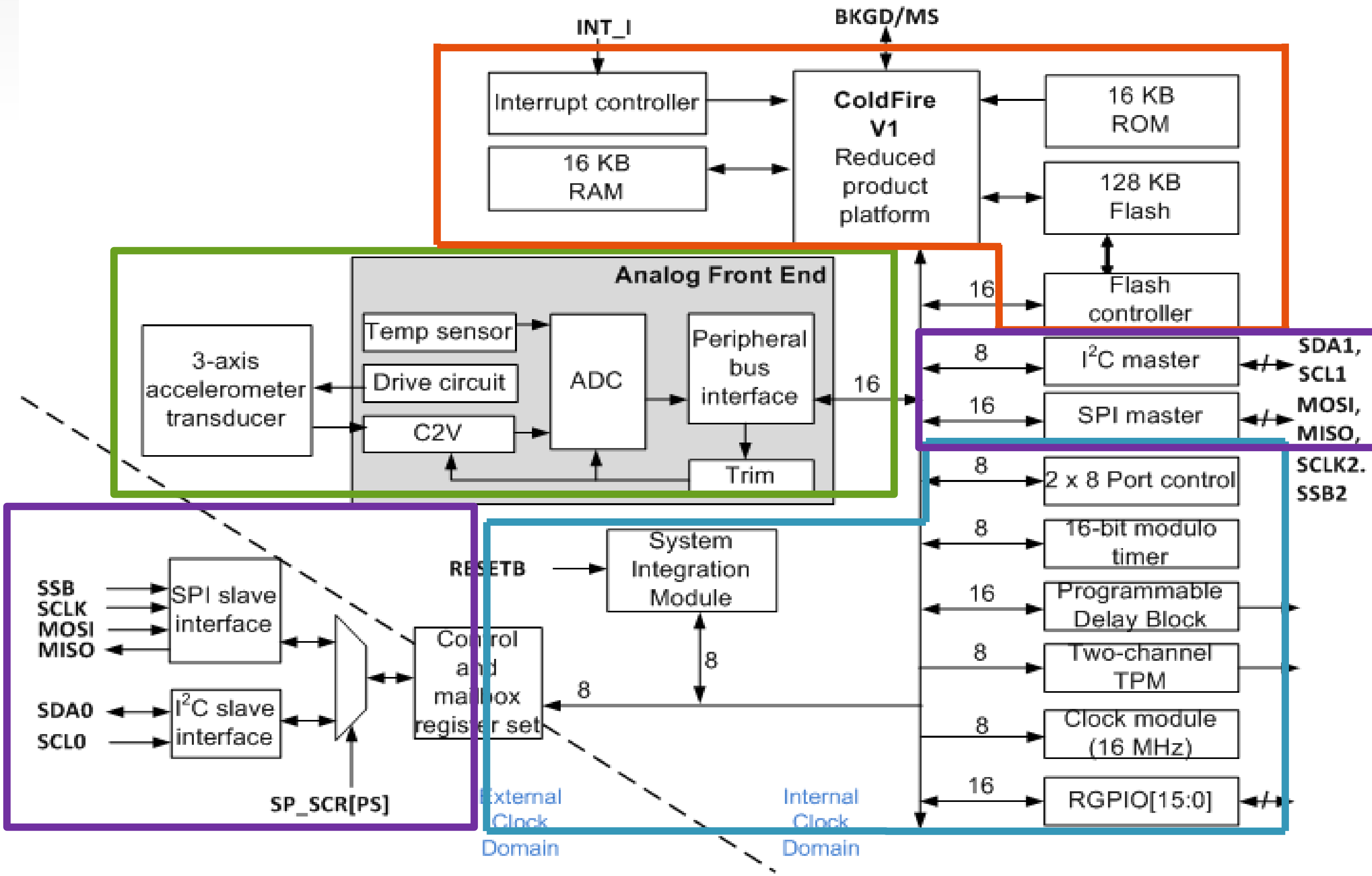
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

FXLC95000 Module: I2C Slave

- **Object:**

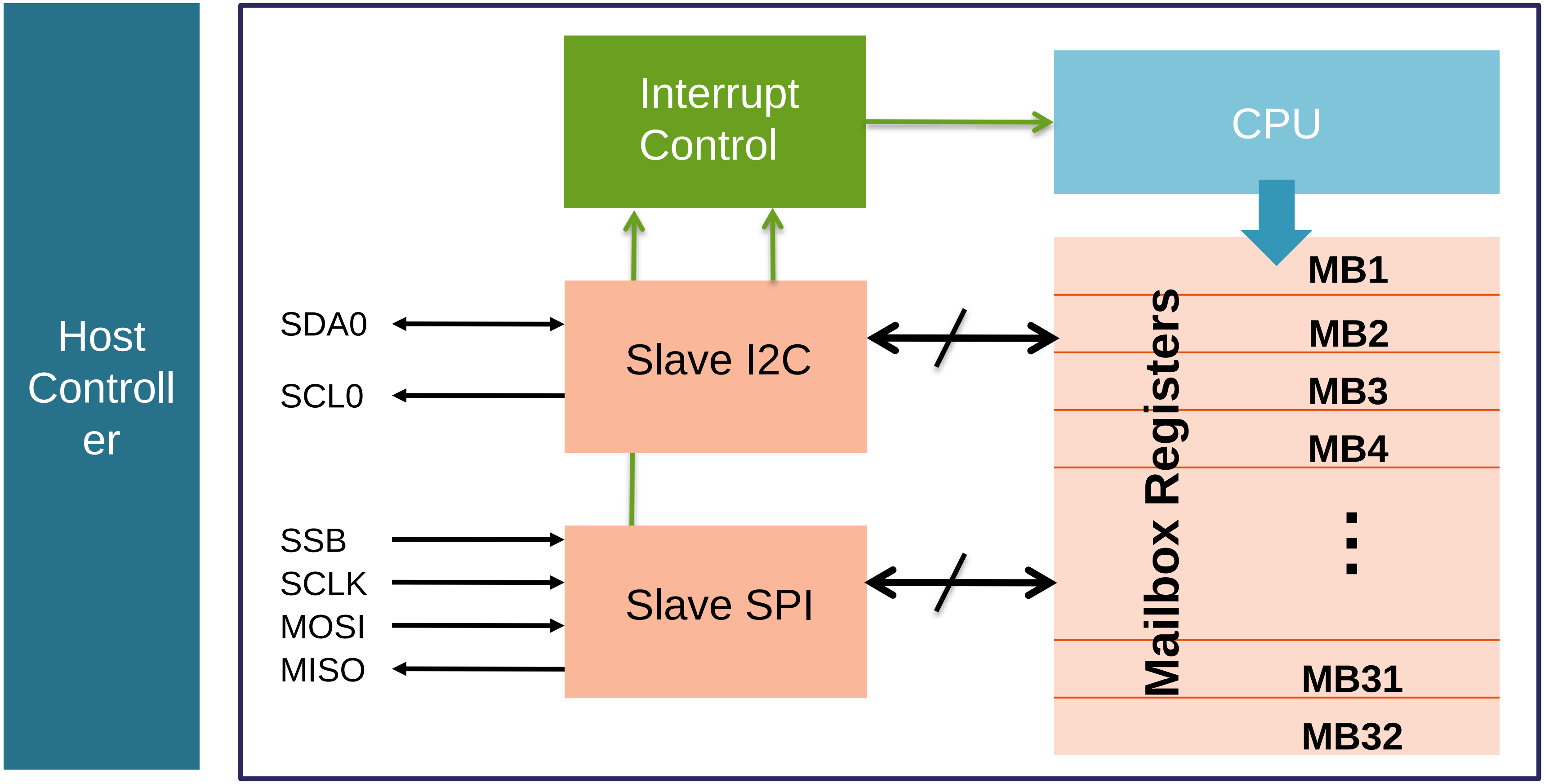
  1. Introduce serial communication port (I2C / SPI), and the Mailbox registers associated to the slave mode ports
  2. Introduce the concept of the Demo GUI
  3. Configure the slave I2C mode
  4. Read / Write mailbox
  5. Use a Demo GUI to communicate with FXLC95000L
     - Observe the AFE data using a demo GUI on slave I2C bus
     - Establish the command that asks for Device ID

FXLC95000L Block Diagram



Lab 5: Mailbox and Slave Communication Ports (I2C/SPI)

- Bidirectional, read/write access by both CPU and Sensor master



Read / Write Mailbox Registers w/ CPU

- **To Write mailbox, simply use code formatted below**

```
SP_MB5 = (uint8)frame_ctr; // MB4
mb_ptr = &SP_MB3_word;
*(mb_ptr)++ = (uint16) data_x; // MB6
*(mb_ptr)++ = (uint16) data_y; // MB8
*(mb_ptr)++ = (uint16) data_z; // MB10
```

- **To Read mailbox, simply use code**

```
flag = SP_MB12;
```

Setup Slave Port

- **FXLC95000L enables interrupt tracking mailbox read / write via slave port**

- **Set up in SP_SCR register (Slave Port Status & Control Register)**

```
// Configure slave port interrupts at the end of writes
SP_SCR = (SP_SCR & (SP_SCR_EN_MASK | SP_SCR_PS_MASK)) |
  SP_SCR_ACTIVE_CSR_MASK |
  SP_SCR_STOP_EN_MASK |
  SP_SCR_WIE_MASK | SP_SCR_RIE_MASK;
```

SP_SCR **SP_BASE + 0x23**

	7	6	5	4	3	2	1	0
R	EN	PS	ACTIVE	STOP_EN	RIE	WIE	WUP	
W			CSR					
Reset	0	0	0	0	0	0	0	0

Mailbox has Read/Write Status Register

- **Write Status Registers SP_WST0,1,2,3**
- **Read Status Registers SP_RST0,1,2,3**

SP_WSTS3 **SP_BASE + 0x27**

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
Reset	0	0	0	0	0	0	0	0

SP_RSTS3 **SP_BASE + 0x2B**

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
Reset	0	0	0	0	0	0	0	0

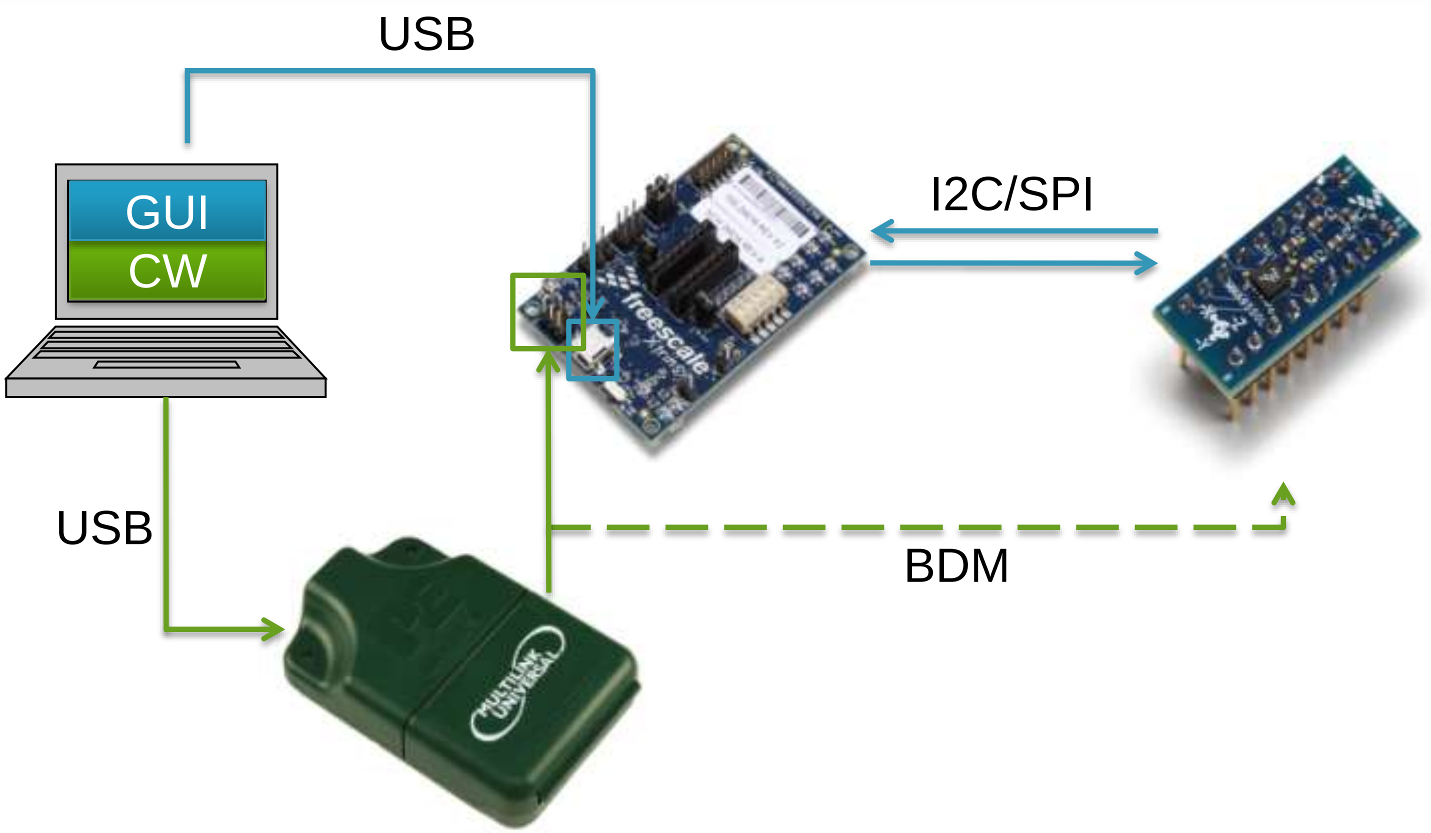
Detect a Mailbox is Written/Read

- **To detect a mailbox is written, below need to be set**
  - SP_SCR_WIE_MASK set
  - SP_SCR_RIE_MASK set

- **To detect a mailbox is written, check**
  - A Slave port interrupt is set
  - ITS corresponding SP_WST0,1,2,3 register bit is set

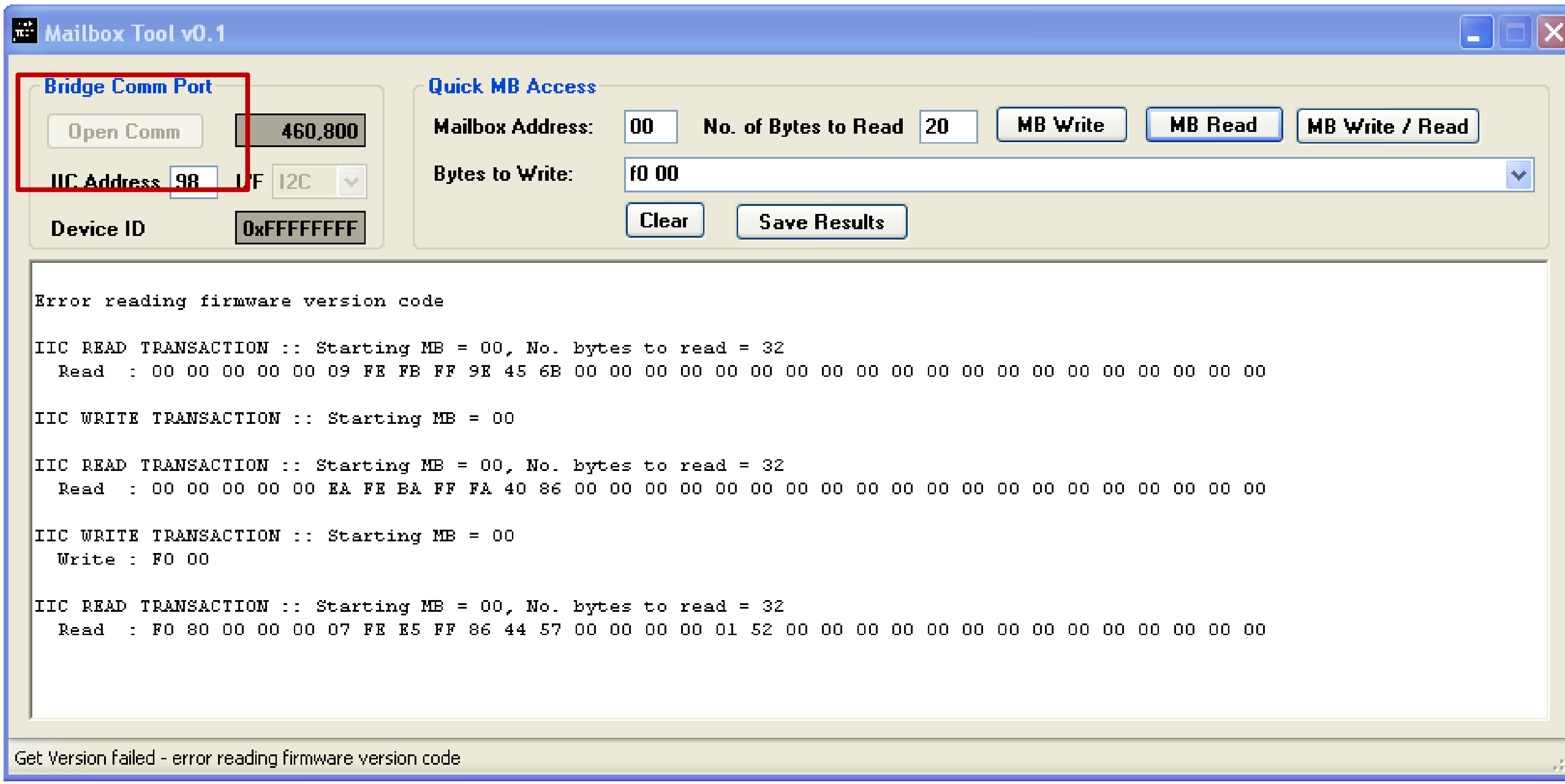
```
if (sp_writes & 0x03) {
   // ID Code = 0xF0 0x00
   if (SP_MB0_word == 0xF000) {
    mb8_ptr = &SP_MB14;
    for (ints=0; ints<4; ints++)
     *(mb8_ptr)++ = ((uint8*)(&device_info))[ints];
    SP_MB1 = 0x80;
   }
  }
```

Hardware Connections



USB-I2C GUI

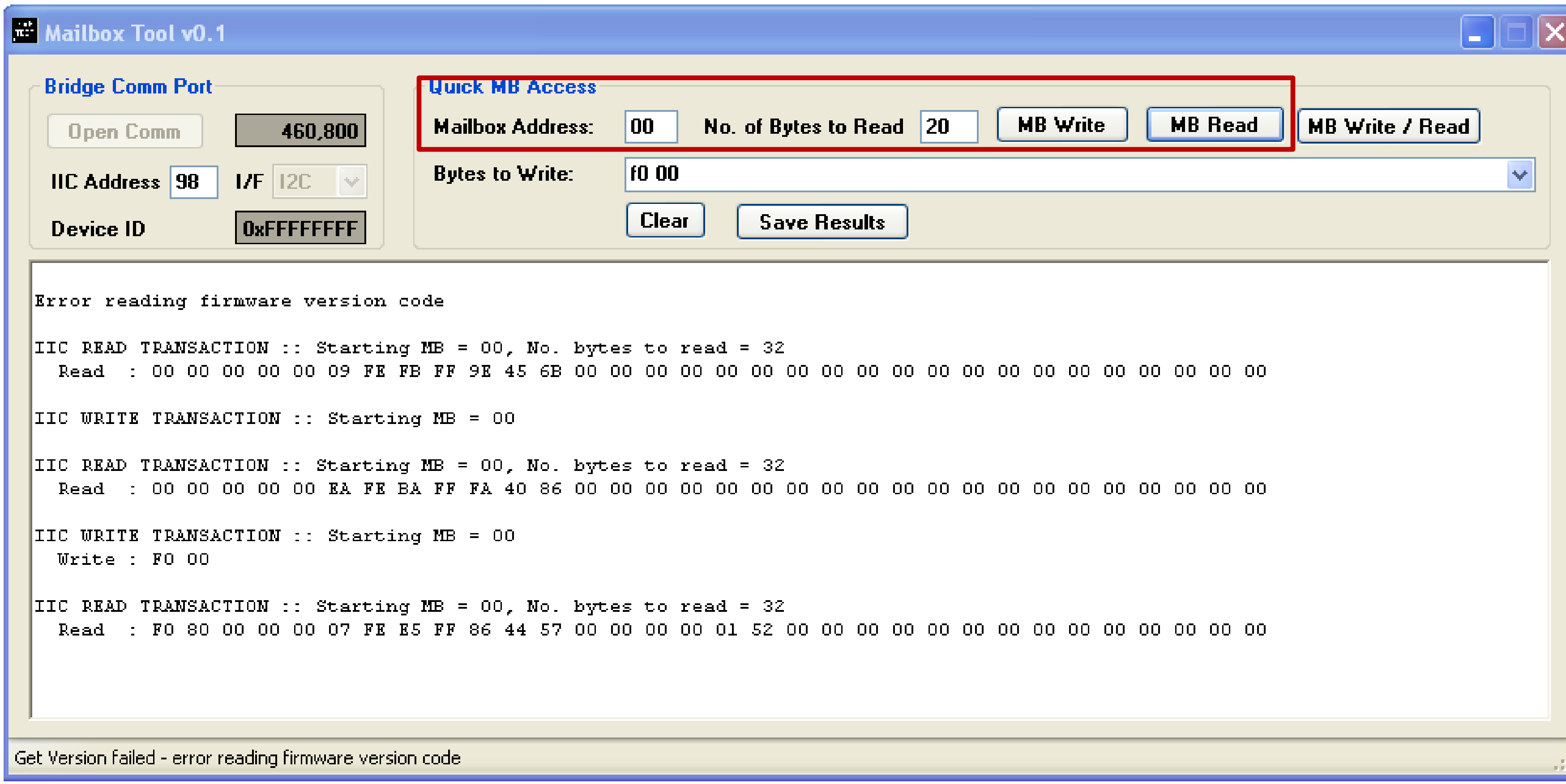
- **Step 1: establish communication**



freescale™

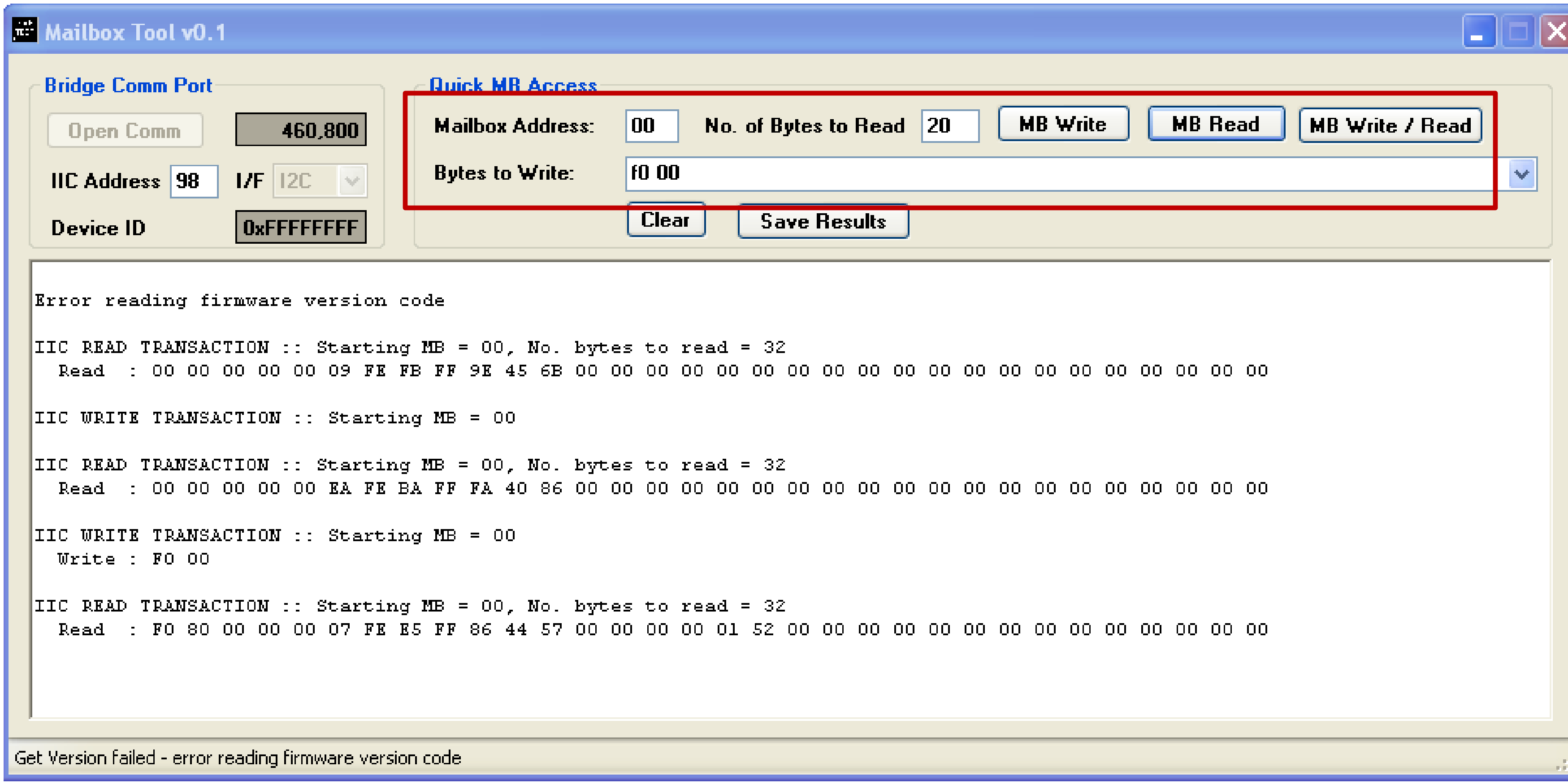
USB-I2C GUI

- **Step 2: Read Mailbox Registers**
  - Starting address, No. of Bytes to Read, MB Read



USB-I2C GUI

- **Step 3: Write Mailbox Registers and Read**
  - Bytes to Write, MB Write + MB Read
  - Bytes to Write, MB Write / Read



Lab 5: Tasks

- **Find the sp_task()**
- **Program to write mailboxes with content on**

Mailbox	Read/Write update	Content
5	Read/write	Frame counter
6,7	Read/write	X axis data
8,9	Read/write	Y axis data
10,11	Read/write	Z axis data
12	Read/write	Flag
13,14,15,16	Write	4 byte Device ID
1	Write, after Dev ID is loaded	0x80

- Use Demo GUI to communicate via slave I2C port and observe

Refer to slides: 96- 108

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

FXLC95000 Module: I2C Master

FTF-ENT-F0091

Agenda

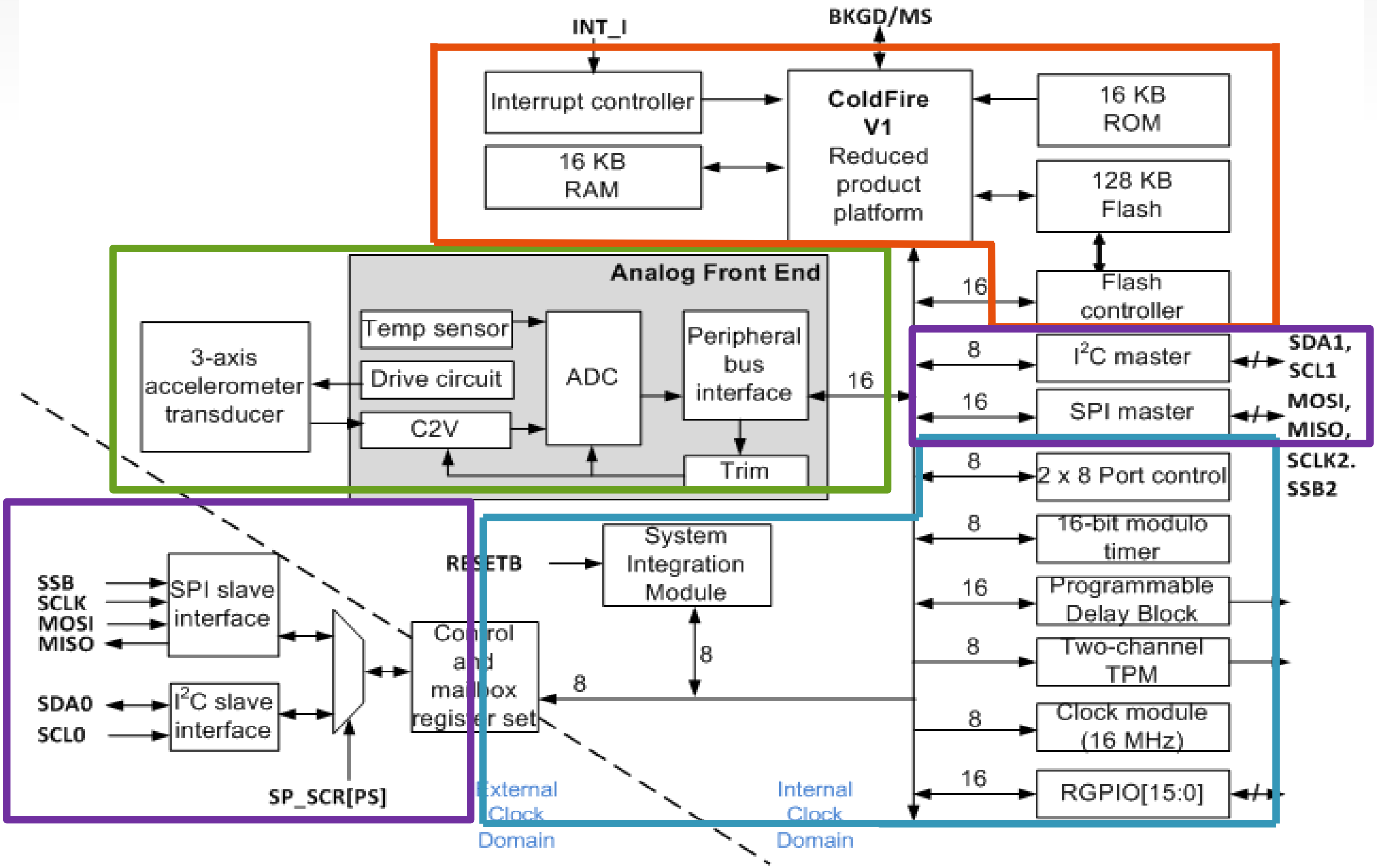
- **FXLC95000L Preview**
- **Designing with the FXLC95000L Platform**
  - Development Hardware and Software Overview <Lab 1>
  - MQX Based Coding Structure < N/A >
  - FXLC95000 Module: GPIO and TPM <Lab 2>
  - FXLC95000 Module: AFE <Lab 3>
  - FXLC95000 Module: AFE Motion Detection <Lab 4>
  - FXLC95000 Module: I2C Slave <Lab 5>
  - FXLC95000 Module: I2C Master <Demo>
- **Summary**

FXLC95000 Module: I2C / SPI Master

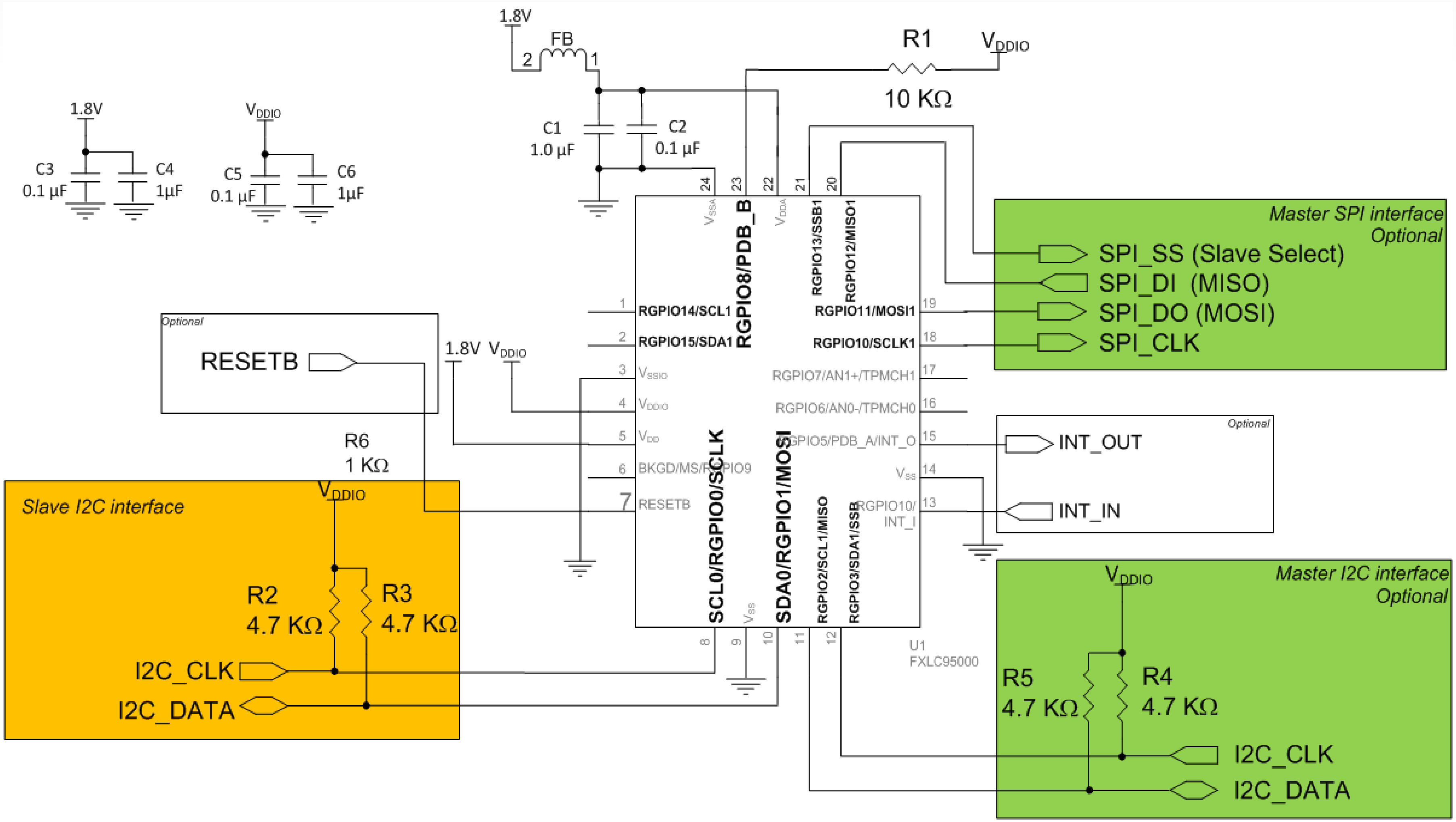
- **FXLC95000L has I2C / SPI Slave + Master**

  → set as a sensor hub

FXLC95000L Block Diagram



Platform as a Sensor Hub (I2C)



Demo: Use FXLC95000 Module as a Sensor Master

- **Demo: FXLC95000 communicate with MAG3110 magnetometer**

- **Observe**
  - More LEDs lit up with stronger magnetic field
  - Less LEDs lit up with weaker magnetic field

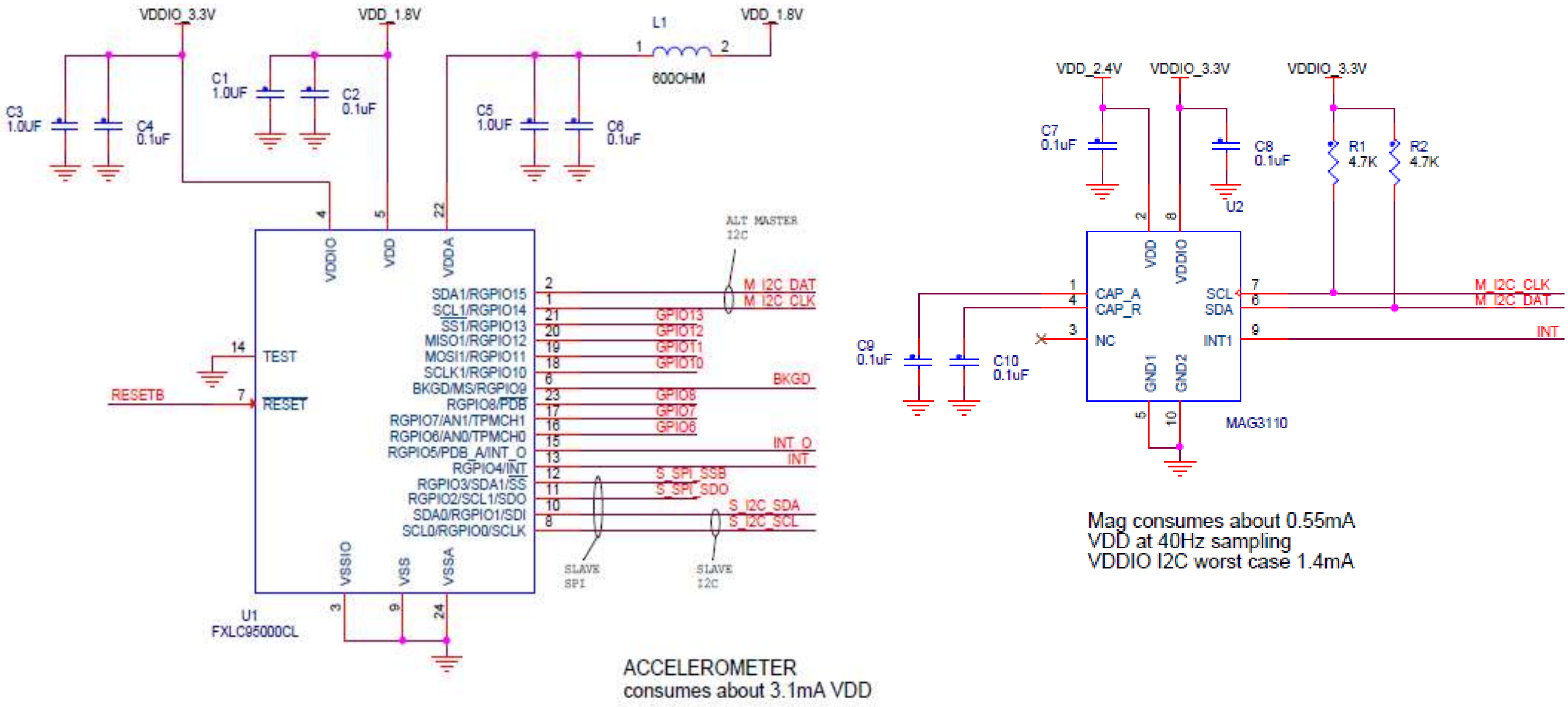
- **Hardware**
  - FXLC95000L Evaluation Kit
  - FXLC95000L with MAG3110 Magnetometer DIP board

Available Resources

- **Setups <find in solution file>:**
  - Configure the FXLC95000L master port
  - Configure the MAG3110 on board as the slave device.
  - See task pi_task() to read Magnetometer data

FXLC95000L Sensor Hub Schematic

- I2C lines: SDA1, SCL1



Session Summary

- **Understand the FXLC95000L device on the high level**
  - The FXLC95000L platform consists of a family of feature rich embedded solutions.
  - Gain an overview of the FXLC95000L AFE, System Clock, GPIO, and I2C modules

- **Developing a simple application using the FXLC95000L platform**
  - Learn the basics on how to create custom FXLC95000L smart sensing solutions project in CodeWarrior 10.1
  - Create a custom application and program the FXLC95000L smart sensor platform to toggle an LED based on a various gesture recognition
  - Communicate with the device via I2C ports

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

For More Information

FTF-ENT-F0091

For Further Information

- **Contact information**
  - Local FAE / Sales
  - Jeannette Wilson (Prod. Line Manager)
  - jeannette.wilson@freescale.com

- **Supporting Docs, References, Hardware kit etc.**
  - www.freescale.com/xtrinsic
  - www.freescale.com/sensingplatform

Facebook.com/Freescale
Tag yourself in photos
and upload your own!

Tweeting?
Please use hashtag
#FTF2012

Session materials will be posted @ www.freescale.com/FTF
Look for announcements in the FTF Group on LinkedIn or follow Freescale on Twitter

Xtrinsic Sensor Classes at FTF

Tuesday

12:45 pm - FTF-ENT-F0137: Lunch and Learn: Xtrinsic Sensor Fusion, Part 1: Terms, Trends, Challenges and Advantages

2:00 pm - FTF-ENT-F0226: Xtrinsic Sensor Fusion, Part 2: Alignment to Sensor Framework for Windows 8, Android and Other Operating Systems

5:15 pm - FTF-CSD-F0225: Xtrinsic Sensing: Introduction, Part 1

Wednesday

10:30 am - FTF-CSD-F0224: Xtrinsic Sensing, Part 4: Applications for Pressure Sensors in Home Appliances and Health Care Solutions

12:45 pm - FTF-ENT-F0123: Lunch and Learn: Augmented Reality: New Applications and the Role of Hardware

12:45 pm - FTF-ENT-F0138: Lunch and Learn: Xtrinsic Sensor Fusion, Part 4: Designing eCompass Solutions

2:00 pm - FTF-IND-F0223: Xtrinsic Sensing, Part 2: Applications for Industrial Automation

4:15 pm - FTF-SEG-F0221: Xtrinsic Sensing, Part 3: Application of Tamper Detection for Smart Metering

5:15 pm - FTF-CSD-F0222: Xtrinsic Sensing, Part 5: Application of Location Based Services: GPS Enhancement and Indoor Navigation

Thursday

9:30 am - FTF-ENT-F0067: Hands-on Workshop: Implementing Xtrinsic Touch Sensing (Reserved Seating Required)

9:30 am - FTF-ENT-F0091: Hands-on Workshop: Xtrinsic Sensor Fusion, Part 3: Start Programming with the Xtrinsic Sensor Platform (Reserved Seat Required)

<compiled by Paul Gan>

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Summary

FTF-ENT-F0091

Summary AND Q&A

- **By now, you should be able to:**
  - Describe, at a high level, the FXLC95000L platform and how it is used in customers' designs.
  - Discuss the key components used when designing products leveraging the FXLC95000L solution
  - Apply the knowledge gained in this presentation to begin or refine your design efforts.

Facebook.com/Freescale
Tag yourself in photos
and upload your own!

Tweeting?
Please use hashtag
#FTF2012

Session materials will be posted @ www.freescale.com/FTF
Look for announcements in the FTF Group on LinkedIn or follow Freescale on Twitter

Thank You!

Facebook.com/Freescale
Tag yourself in photos
and upload your own!

Tweeting?
Please use hashtag
#FTF2012

Session materials will be posted @ www.freescale.com/FTF
Look for announcements in the FTF Group on LinkedIn or follow Freescale on Twitter

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

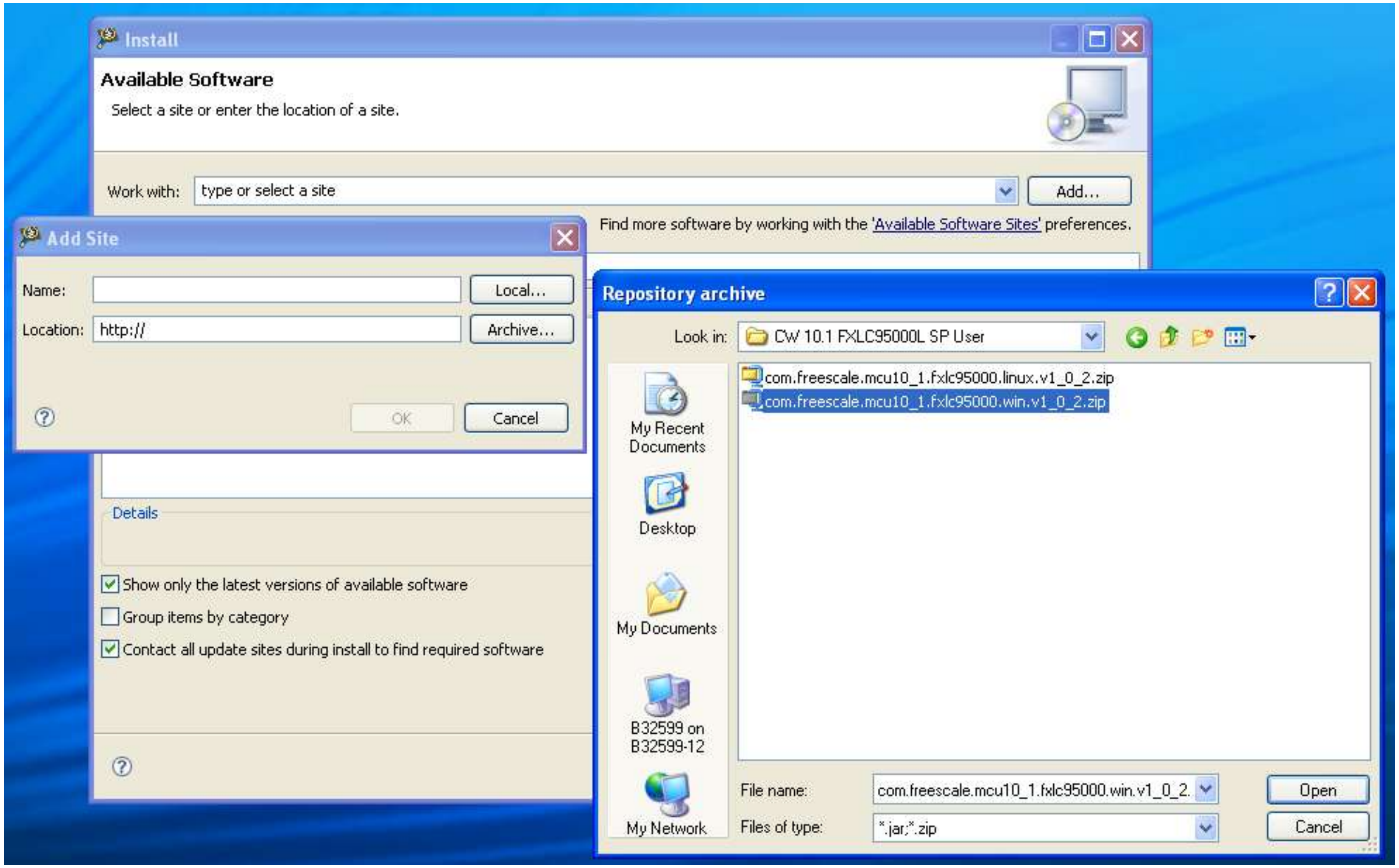
Install Service Pack

Install FXLC95000 CW Service Pack

- 1. Open CW MCU 10.1
- 2. Download all service pack
  - CW 10.1 compiler update
  - CW 10.1 update
  - CWFXLC95000 service pack com.freescale.mcu10_1.FXLC95000.win.v1_0_0.zip 10.1
- 3. Go to Menu →Help→ Install New software
- 4. When the install dialog is display, choose “Add“ next to “--All Available Sites—” menu.

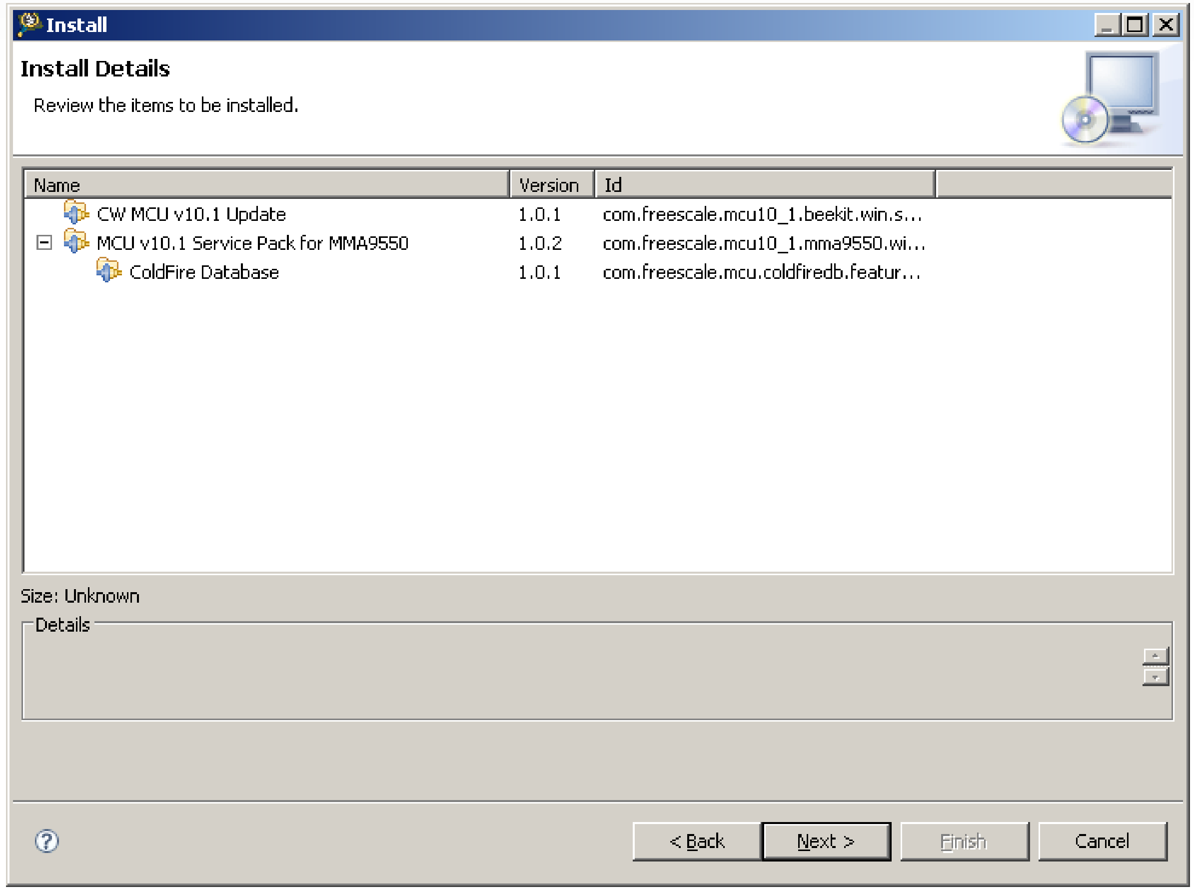
Install FXLC95000 CW Service Pack

- 5. In the add site, choose “Archieve”.
- 6. Find the service pack that needs to be installed.



Install FXLC95000 CW Service Pack

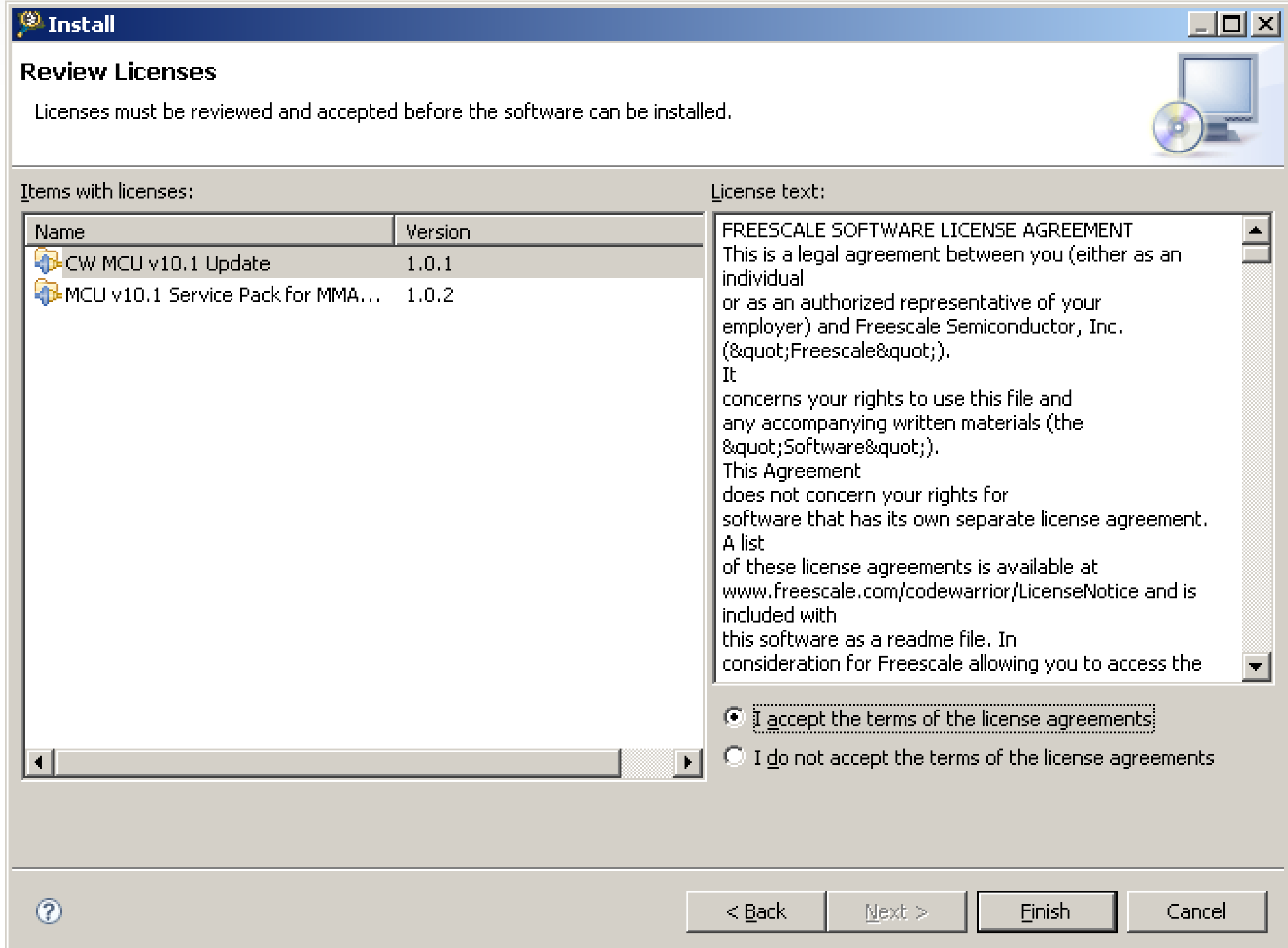
- 7. CodeWarrior identifies and lists the related data packages that need to be installed, including the MMA955xL service pack and display the details. Click on the patch to view more details. Click “Next”.



freescale™

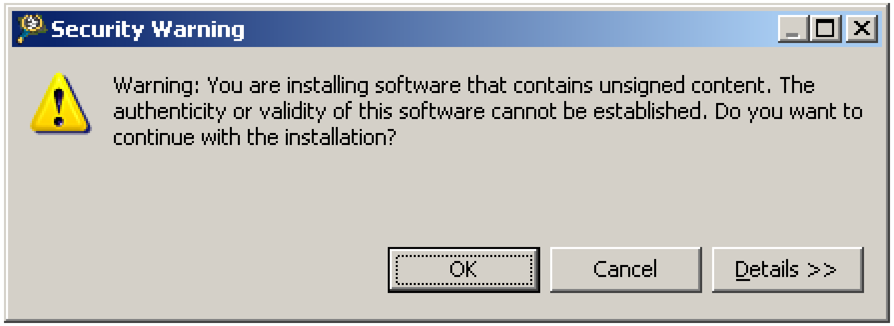
Install FXLC95000 CW Service Pack

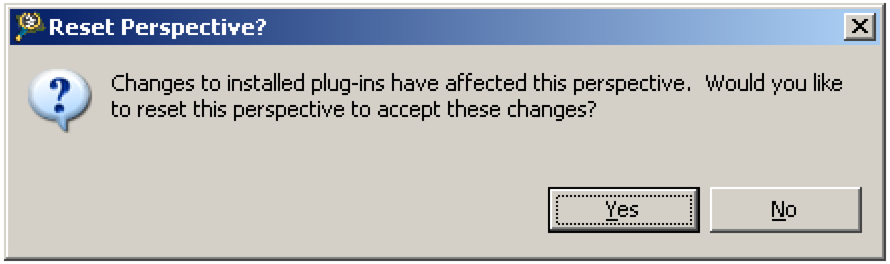
- 8. In the Review Licenses dialog, read through the Freescale licensing agreement and click “I agree the terms of the license agreement” option. Make sure internet connection is valid. Click “Finish” to begin the service pack installation.



Install FXLC95000 CW Service Pack

- 9. During the installation, Security warning messages below might pop up. Click “OK” to continue installation.





Install FXLC95000 CW Service Pack

- 10. At the end of installation, a Software Updates Message Window pops up. Click “No”
  11. Repeat the process to finish install all service packs.
  12. At the end of installation, a Software Updates Message Window pops up. Click “Yes” to restart CW for service packs to take effect.

freescale™

FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

25 April 2012

MQX Fxlc95000 Patch Installation

Hung Tran
Software Development

Margaret Nadworny
Software Manager

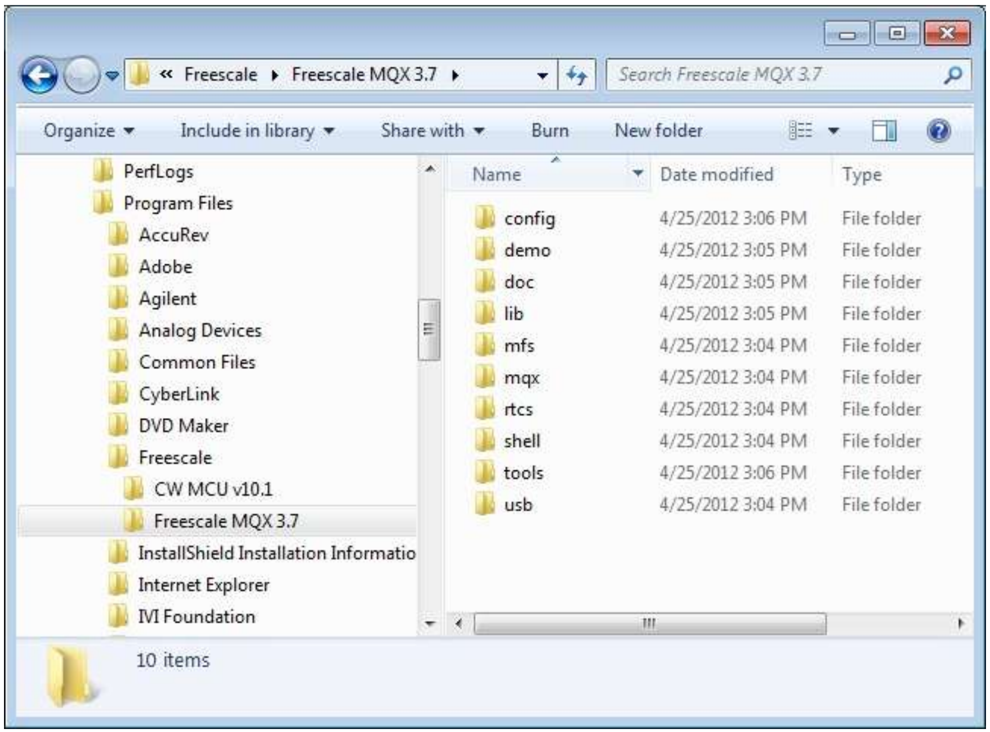
Created 25 April 2012

MQX Fxlc95000 Patch Installation Instruction

- Instruction for installing "Freescale MQX 3.7 Fxlc95000 patch.zip"
  - Provides MQX RTOS support for Fxlc95000 device.
  - Installs Fxlc95000 board & processor support packages.

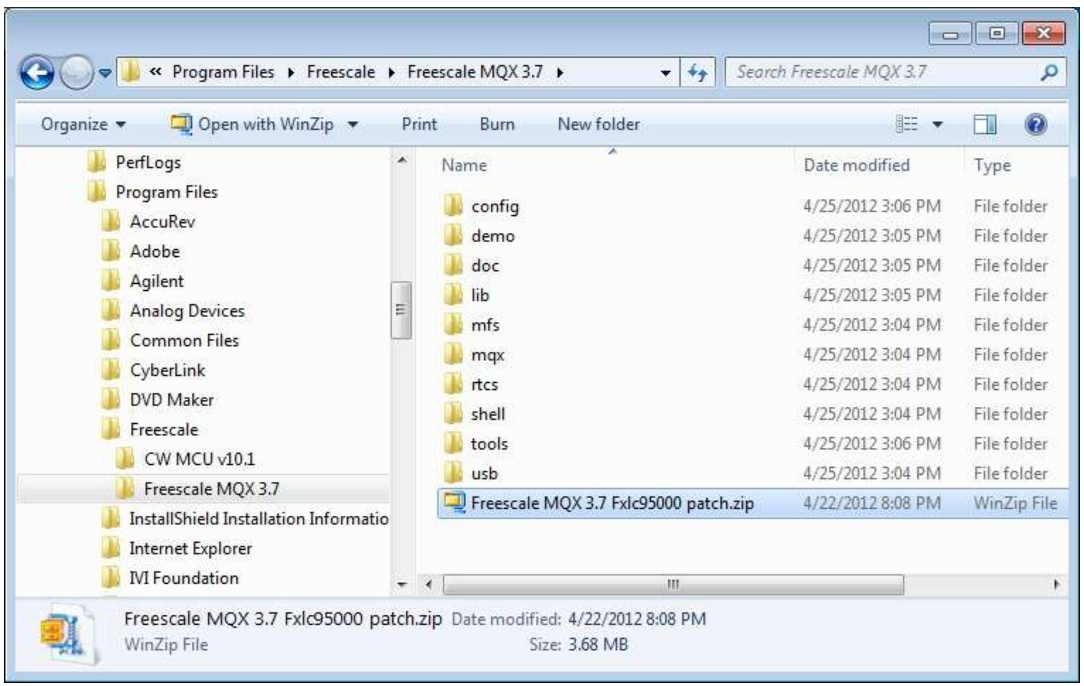
MQX Fxlc95000 Patch Installation Instruction

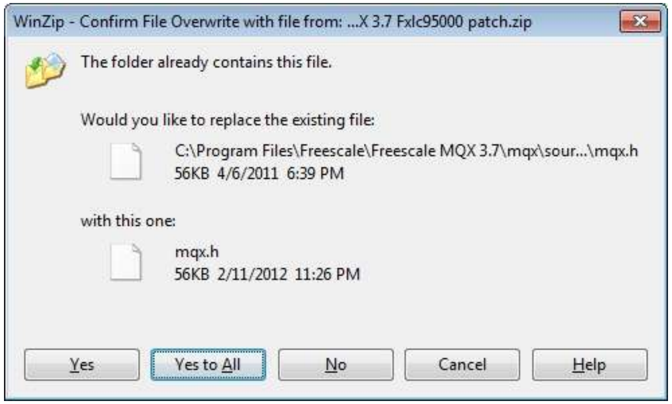
- Step 1: Install Freescale MQX v3.7
  - Download MQX file “FSLMQXOS_3_7_0.exe” from Freescale website and run.
  - MQX should be installed in default directory.
    - Win XP and Win7: C:\Program Files\Freescale\Freescale MQX 3.7.
    - For Win7, DO NOT install in the C:\Program Files (x86)\ directory.



MQX Fxlc95000 Patch Installation Instruction

- Step 2: Install patch file “Freescale MQX 3.7 Fxlc95000 patch.zip”
  - Copy the patch file to the MQX installed directory: C:\Program Files\Freescale\Freescale MQX 3.7.
  - Unzip the patch file.
  - The zip program will ask whether to overwrite existing files, select YES to overwrite existing files.
  - The MQX Fxlc95000 patch installation is now complete.





freescale™

Agenda (combined)

- **FXLC95000L Overview**
- **Designing with the FXLC95000L Platform**
  - Development hardware and software overview <Lab 1>
    - Lab 1: Import, build and download demo project
  - MQX Coding Structure
  - FXLC95000 Module: GPIO and TPM
    - Lab 2: Toggle LED using the RGPIOs
  - FXLC95000 Module: AFE
    - Lab 3: Acquire AFE data
  - FXLC95000 Module: AFE – Motion Detection
    - Lab 4: Toggle LED based on Gesture Recognition
  - FXLC95000 Module: I2C Slave
    - Lab 5: Communicate with FXLC95000L via slave I2C ports
  - FXLC95000 Module: I2C Master
    - Demo: Use FXLC95000L as a sensor hub
- **Summary**

freescale™