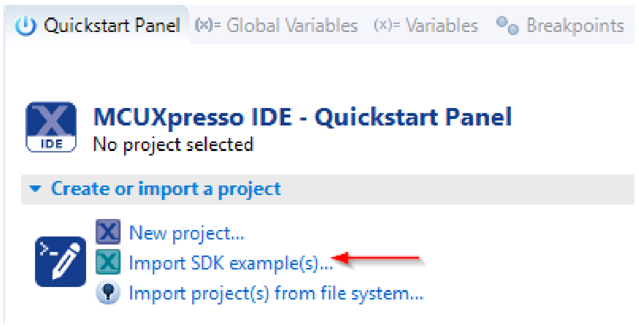
Lab 1 – Demo Applications

Demo application are SDK examples that provide more complex use case examples. Examples are still simple enough to provide basic understanding of how the include pieces work together.

Hello World

1. *Clone the "**hello_world**" example application*

   1.1. *From the **Quickstart Panel** (lower left dock of the IDE) select "**Import SDK example(s)…**"*



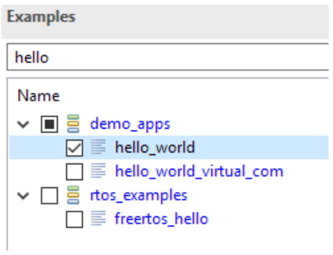
   1.2. *Select the FRDM-K64F board.*

      1.2.1. *If needed to help filter the selection of SDK, select "K6x" from the list of "SDK MCUs"*

      1.2.2. *Locate and select the picture of the board (the text below the picture is a hyperlink to learn more about the board on* http://nxp.com

      1.2.3. *Select the board selected (highlighted) select "**Next**"*

   1.3. *Locate and place a checkmark next to the "**hello_world**" example application*



      1.3.1. *Option 1) use the text filter to filter on **hello**, expand the results to find the correct example*

      1.3.2. *Option 2) browse the list of examples expanding the following path: **demo_apps -> hello_world***

   1.4. Review default selections

      1.4.1. For this example we will initially be using **Semihosting** as the Debug Console (*selected by default*)

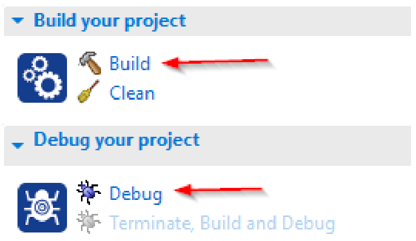
   1.5. Select **Finish**

2. Explore the project source code:

   2.1. The "user application" files are locates in the "sources" subfolder

   2.2. The main application is typically in the source file with the same name as the project.

3. Build and Debug



3.1. *Connect the **microUSB** cable between the laptop and the **OpenSDA** connector on the laptop (USB connector nearest the reset button)*

3.2. Build "***hello_world***" application using the "**Build**" option in the **Quickstart Panel**

3.3. *Launch the debug session using the "**Debug**" option in the **Quickstart Panel***

3.3.1. *Launch the debug session from the Quickstart Panel will initially open a probe discovery dialog.*

3.3.2. *Select the **CMSIS-DAP** probe from the list and select **OK***

4. *Runtime Controls*

4.1. *Refer to the table below for common runtime controls that are located along the top menu of the IDE*

Button	**Description**	**Keyboard Shortcut**
↺	Restart program execution (from reset)	
▶	Run/Resume the program	F8
⏸	Pause Execution of the running program	
🟥	Terminate the debug Session	Ctrl + F2
✖	Clean up debug	
▶ ⏸ 🟥	Run, Pause, Terminate all debug sessions	
↷	Step over a C/C++ line	F6
↘	Step into a function	F5
↗	Return from a function	F7
↘ ↷ ↗	Step in, over, out all debug sessions	
i→	Show disassembled instructions	

4.2. *Use the Run/Resume (play) button to start execution of the application*

4.2.1. *With semihosting used for debug, you should see "Hello World" display in the IDE Console*

4.2.2. *Typing into the console will echo the text after pressing enter. Text in green is the local echo, not the PUTCHAR implemented in the project.*

5. **End** the debug session using the **Terminate** (red stop) button
Note: If further in this lab you ever you encounter a "There is already a running debug session…" error message it is likely that you did not properly end / terminate the previous session. Use the red stop button to end the debug session and to try launch the new debug session again.

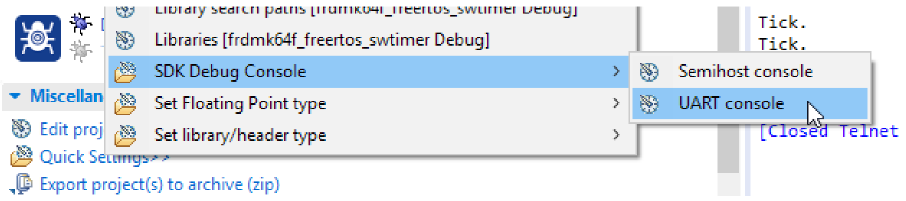
6. Switch the example application from Semihosting to UART Debug Console

6.1. Open Tera Term (shortcut in windows Task bar)

6.1.1. Select "Serial" and locate the "USB Serial Device" from the list (COM port may vary), Select **OK**

6.1.2. Select **Setup -> Serial Port** from the Tera Term menu and configure the Baud Rate to 115200
Note: Tick message now appears in Tera Term window

6.2. From the Quickstart Panel, under Miscellaneous select
Quick Settings -> SDK Debug Console -> UART console


Library search paths [frdmk64f_freertos_swtimer Debug]
Libraries [frdmk64f_freertos_swtimer Debug]
SDK Debug Console
Set Floating Point type
Set library/header type
Semihost console
UART console
Miscellan
Edit proj
Quick Settings>>
Export project(s) to archive (zip)

```
Tick.
Tick.
[Closed Telnet
```

6.3. From the Quickstart Panel, select **Debug** (which will cause a clean and build to occur)

6.4. Start the application with the **Run/Resume** button
Note: Now the output will appear on the Tera Term console. Note the difference in way the text input/output works. Local echo is not enabled by default in Tera Term, outputted text is from PUTCHAR.

7. End the debug session using the Terminate (red stop) button.

8. Explore **hello_world** project

- Items to note:
  - Minimal set of drivers included

Bubble Peripheral

9. Repeat the appropriate steps above to import the **demo_apps/bubble_peripheral** demo application

10. Explore **bubble_peripheral** project, explore Config Tools

10.1. Project reads accelerometer data within main loop, then periodically updates the state of the LEDs when the FTM Timer expires

- Items to note:
  - Use of Config Tool additional function group calls
    - **BOARD_Init_I2C_GPIO_pins**() – called from BOARD_I2C_ReleaseBus()
    - **BOARD_I2C_ConfigurePins**() – called from main()
  - Use of Peripheral Tool: **FTM Timer, I2C**

11. Build, Debug, and Run application

11.1. Tilting the board in various vertical orientation will cause corresponding LEDs to react

11.2. X and Y angle values based on the accelerometer readings

Command Line Shell Utility

12. Repeat the appropriate steps to import the **demo_apps/shell** demo application
13. Explore **shell** project
    13.1. Project initializes a Shell instance, registers a set of command (xLedCommand), then calls the Shell_Main function to execute the shell function.
    - Items to note:
        - **xLedCommand** create a set up shell command
14. Build, Debug, and Run application
    14.1. Typing “led 1 on” will turn on the Red LED
    14.2. Note: program is more responsive using Tera Term instead of semihosting

Lab 2 – CMSIS Driver Examples

CMSIS driver examples showcase basic driver usage of CMSIS drivers, such as SPI, I2C, and UART. CMSIS driver are an ARM standard API that provide increase portability.

CMSIS I2C Read Accel Value Transfer

1. Repeat the appropriate steps to import the **cmsis_driver_examples/cmsis_i2c_read_accel_value_transfer** demo application
2. Explore **cmsis_i2c_read_accel_value_transfer** project
    2.1. Project reads 10 samples of the accelerometer and displays the result on the console
    - Items to note:
        - Addition of **CMSIS_driver** folder (Common and I2C)
        - Uses CMSIS standard to identify accelerometer
        - ACCEL_READ_TIMES (line 64) can be used to change the number of samples, consider increasing to 100
        - More details regarding CMSIS Driver APIs can be found at: www.keil.com/cmsis/driver
3. Build, Debug, and Run application
    3.1. Tilt the board to see resulting changes to the x,y,z printout

Lab 3 – SDK Driver Examples

SDK driver examples includes single-point driver examples the highlight basic functionality of the respective SDK driver. These examples provide an excellent resource in learning more about the drivers commonly used APIs and use cases. Some driver examples also include FreeRTOS examples.

I2C Read Accel Value Transfer (Optional)

1. Repeat the appropriate steps to import the **driver_examples/i2c_read_accel_value_transfer** demo application
2. Explore **i2c_read_accel_value_transfer** project
   - 2.1. Project reads 10 samples of the accelerometer and displays the result on the console
     - Items to note:
       - Compare to previous CMSIS I2C demo (Uses I2C_<function name> API)
       - ACCEL_READ_TIMES (line 65) can be used to change the number of samples, consider increasing to 100
3. Build, Debug, and Run application
   - 3.1. Tilt the board to see resulting changes to the x,y,z printout

Low Power Timer

4. Repeat the appropriate steps to import the **driver_examples/lptmr** demo application
5. Explore **lptmr** project
   - 5.1. Project creates a LPTMR with default settings and increments a counter inside the corresponding interrupt handler. Main while(1) loop monitors the counter and prints with it increments
     - Items to note:
       - Uses “GetDefaultConfig” and “Init” to configure and initialize the Low Power Timer to default conditions
       - Uses LPTMR_SetTimerPeriod and LPTMP_StartTimer to control timer
6. Build, Debug, and Run application
   - 6.1. View output on screen

EDMA Memory to Memory

7. Repeat the appropriate steps to import the **driver_examples/edma_memory_to_memory** demo application
8. Explore **edma_memory_to_memory** project
    8.1. Project transfers 4 bytes of data from a source address to destination address
    - Items to note:
        - Configures DMAMUX to use channel 63 (Always Enabled)
        - Configures EDMA with default configuration and callback function **EDMA_Callback**
9. Build, Debug, and Run application
    9.1. View print printout of destination address before and after transfer

Random Number Generator

10. Repeat the appropriate steps to import the **driver_examples/rnga_random** demo application
11. Explore **rnga_random** project
    11.1. Project generate a series of random numbers from a random hardware generator hardware accelerator
    - Items to note:
        - RNGA Driver does not require any configuration
        - Calls to **RNGA_Init** and **RNGA_GetRandomData**
        - Optional calls to set Seed and Mode are not used in this example
12. Build, Debug, and Run application
    12.1. View print printout of random numbers

Lab 4 – FreeRTOS Examples

FreeRTOS examples include FreeRTOS examples that showcase RTOS specific features, such as: events, mutexes, semaphores, software timers.

FreeRTOS Hello World

1. Repeat the appropriate steps to import the **rtos_examples/freertos_hello** demo application
2. Explore **freertos_hello** project
   - 2.1. Project creates a “Hello_task” that prints out “Hello world” the suspends the task
     - Items to note:
       - **xTaskCreate**() – creates and registers a task with specified stack size and priority
       - **vTaskStartScheduler**() – starts the RTOS tick processing
       - FreeRTOS configurations are set in **FreeRTOSConfig.h**
3. Build, Debug, and Run application
   - 3.1. View “Hello world.” on console

FreeRTOS Queues

4. Repeat the appropriate steps to import the **rtos_examples/freertos_queue** demo application
5. Explore **freertos_queue** project
   - 5.1. Project creates three tasks: two tasks at the same priority that log a message into a common queue and a lower priority task that retrieves messages from the queue and prints them to the screen
     - Items to note (in addition to the items in the **freertos_hello** project above):
       - **xQueueCreate**() – creates the queue with length and size of items
       - **xQueueSend**() – adds messages to the queue
       - **xQueueReceive**() – waits for item to be added to the queue retrieves it
       - **FreeRTOS TAD** will not show this queue by default, it needs to be registered first using: `vQueueAddToRegistry(log_queue,"Log Queue");`
       - Optional: This project can be explored further by adjusting task priorities
6. Build, Debug, and Run application
   - 6.1. View “Hello world.” on console

Lab 5 – Middleware Examples

Middleware examples will vary depending on the specific middleware, but are a valuable resource in understanding the basic use cases and API used.

Lightweight IP HTTP Web Server

1. Repeat the appropriate steps to import the **lwip_examples/lwip_httpsrv_freertos** demo application
2. Explore **lwip_httpsrv_rtos** project
   2.1. Create a webserver running on the target board with Webserver, CGI Post/Get and Polling, SSI update request, http authentication, and web sockets
   - Items to note:
     - Webpage is stored as C arrays, example include perl script to convert webpage content to source code, see readme.txt for details
     - **cgi_lnk_tbl** and **ssi_lnk_tbl** include registered callbacks for CGI and SSI functions
     - Web sockets register: Connect, Disconnect, Message, and Error callbacks (**ws_tbl**)
     - There are two main function calls:
       - **stack_init**() – Initializes the networking interface, configures IP address
       - **http_server_socket_init**() – Initializes various services (CGI, SSI, WebSocket), configures the starting index.html page
3. Build, Debug, and Run application
   3.1. Configure IPv4 address of the local wired to **192.168.0.100**
      3.1.1.Open “Network and Sharing Center” and click **Change adapter settings**
      3.1.2.Right-click **Ethernet** and select **Properties**
      3.1.3.Double-click Internet Protocol Version 4 (TCP/IPv4)
      3.1.4.Select “Use the following IP address” and type in 192 168   000 100
   3.2. Open web browser and enter IP address **192.168.0.102**

USB Device – MSC in RAM

4. Repeat the appropriate steps to import the **usb_examples/dev_msc_ramdisk_bm** demo application

5. Explore **dev_msc_ramdisk_bm** project

   5.1. Creates a Mass Storage Device using a section of target RAM that can be formatted, written to, and read from by the connect host computer

   - Items to note:
     - There are two main USB Class Configuration structures
       - Class Config: **msc_config**, which registers the class specific **USB_DeviceMscCallback** function
       - Class Config List: **mcs_config_list**, which registers the class config and device specific **USB_DeviceCallback** function

6. Build, Debug, and Run application

   6.1. Connect second USB cable between Laptop and FRDM-K64F board using the other microUSB connector

   6.2. Windows will enumerate a connected USB Mass Storage Device as "USB Drive"

   6.2.1. Driver can be formatted and small text file created

   6.2.2. USB can be disconnected and reconnected and file will remain, assuming FRDM board remained powered

USB Host – HID Mouse Support

7. Repeat the appropriate steps to import the **usb_examples/host_hid_mouse_bm** demo application

8. Explore **host_hid_mouse_bm** project

   8.1. Implements a HID controller that supports connecting a mouse via a microUSB to USB Type A adapter

   - Items to note:
     - Host is initialized with **USB_HostEvent** as the ISR function which calls **USB_HostHidMouseEvent** to process connections
     - Application calls the **USB_HostHidMouseTask** function in a while(1) to poll for mouse event

9. Build, Debug, and Run application

   9.1. Connect the microUSB to Type-A adapter to the FRDM board, disconnect the mouse from the laptop and connect to the adapter

   9.2. Mouse events (clicks and moves) will cause a corresponding PRINTF on the screen.