Week 4 Lab

CC2511

You will learn how to send and receive over a UART.

Your task

Build an application that allows each of the LEDs to be controlled individually using the serial port.

- When the "r" key is received, toggle the red LED.
- When the "b" key is received, toggle the blue LED.
- When the "g" key is received, toggle the red LED.

Identify the pins that drive each LED

Refer back to your work last week, or read the schematic, to identify the microprocessor pins that drive each colour of the LED[1]:

[1] You do not need to write the full name of the pin. Here, you can use the GPIO name which is of the format "PT*xx*", e.g. PTA1, PTB2, etc.

LED Colour	**Microprocessor pin**
Red	PTA1
Green	PTA2
Blue	PTD5

Identify the UART pins

One of the microprocessor UARTs is connected to the OpenSDA module. This is the UART that is available on your computer via USB. The other UARTs are available for custom hardware that you might build to plug into your development board.

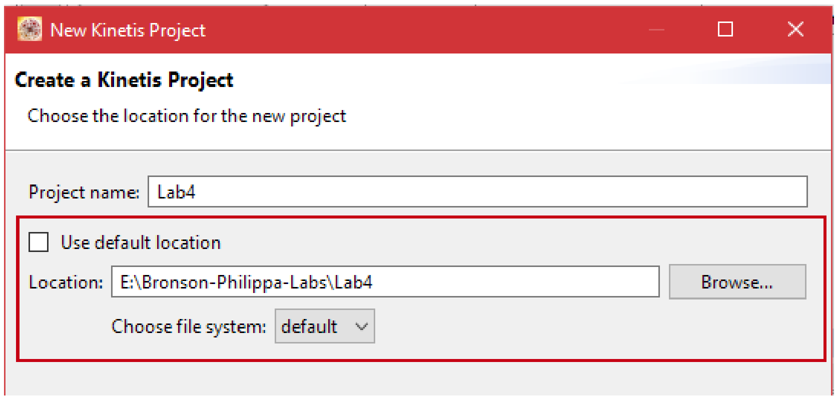
You need to read the schematic to identify which UART connects to OpenSDA and therefore to your computer. There will be two pins: RX and TX.

UART Function	**Microprocessor pin**
Receive (RX)	UART1_RX_TGTMCU_R
Transit (TX)	UART1_TX_TGTMCU_R

Writing the software

Creating a Kinetis Project

1. In Kinetis Design Studio, select **File -> New -> Processor Expert Project**.
2. Name your project "Lab4". **Ensure that the location is inside your Git repository.**


New Kinetis Project

Create a Kinetis Project

Choose the location for the new project

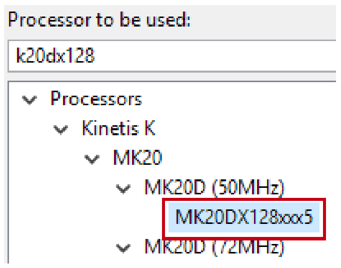
Project name: Lab4

Use default location

Location: E:\Bronson-Philippa-Labs\Lab4

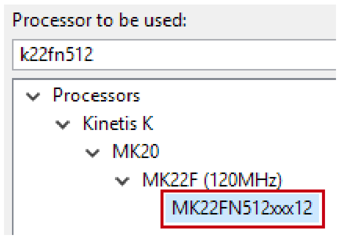
Choose file system: default

3. Choose the correct processor type for the dev board that you have.
   - **`K20D50M` board:** under `MK20D (50 MHz)`, select the **`MK20DX128xxx5`** processor type.


Processor to be used:

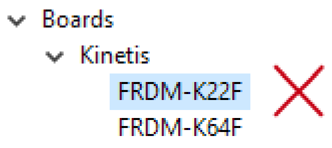
k20dx128

   - **`K22F` board:** under `MK22F (120 MHz)`, select the **`MK22FN512xxx12`** processor type.


Processor to be used:

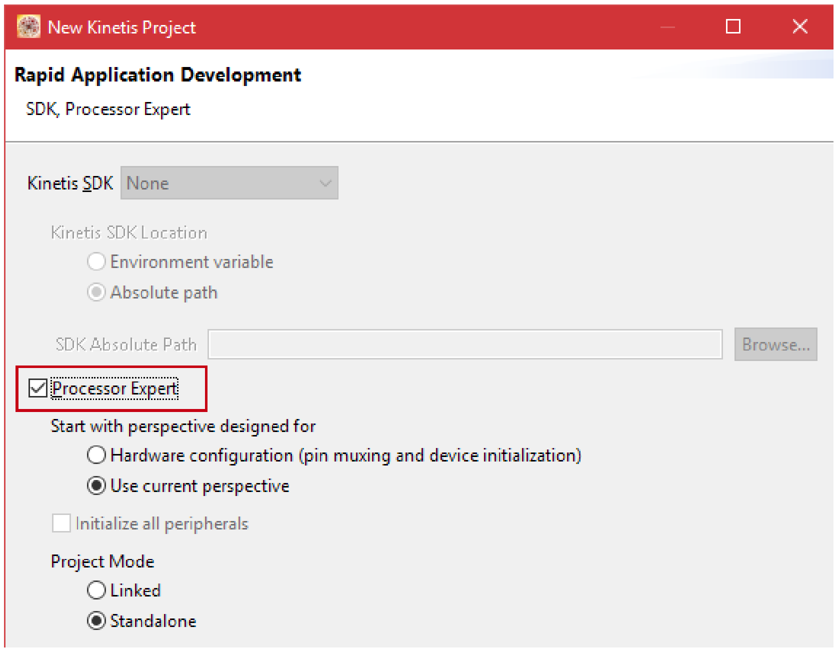
k22fn512

If you have the Kinetis SDK installed, you might have an option to choose the `K22F` board instead of the underlying processor. Do not do this!



To avoid problems, choose the processor. The "board" option will preconfigure some clock modes that conflict with the options you'll choose later. This lab is easier if you start with the plain processor instead of the preconfigured board.

4. Ensure that **Processor Expert** is ticked and click Finish.



New Kinetis Project

Rapid Application Development

SDK, Processor Expert

Kinetis SDK: None

Kinetis SDK Location
- ( ) Environment variable
- (●) Absolute path

SDK Absolute Path: Browse...

[✓] Processor Expert

Start with perspective designed for
- ( ) Hardware configuration (pin muxing and device initialization)
- (●) Use current perspective

[ ] Initialize all peripherals

Project Mode
- ( ) Linked
- (●) Standalone

Configuring the Processor Expert Components

Use the following Processor Expert components:

- **BitIO** for controlling the LED pins.
- **AsynchroSerial** for controlling the UART. You must configure AsynchroSerial to operate without interrupts[2]. To do this, set the property "**Interrupt service/event**" to "**Disabled**."

Refer to the Processor Expert documentation for each component to learn how to drive these components. In particular, the "Typical Usage" page of the AsynchroSerial documentation gives example code.

[2] Interrupts are a mechanism for event handling, whereby external events can "interrupt" the main program by causing the CPU to stop its current task and jump to a different part of the program to run an interrupt handler instead. We will consider interrupts later in the course. For now, disable interrupts to keep things simpler.

A common issue

The default CPU configuration in Processor Expert specifies certain pins for debug functionality. If you try to reuse those same pins for another purpose, such as BitIO, you will receive an error message that says "Peripheral is already used by the component" or "Selected value is in conflict with other configuration(s)." What is happening here is that two components are trying to select different MUX settings on the same Pin Control Register, i.e. two components are trying to use the same pin for different purposes.

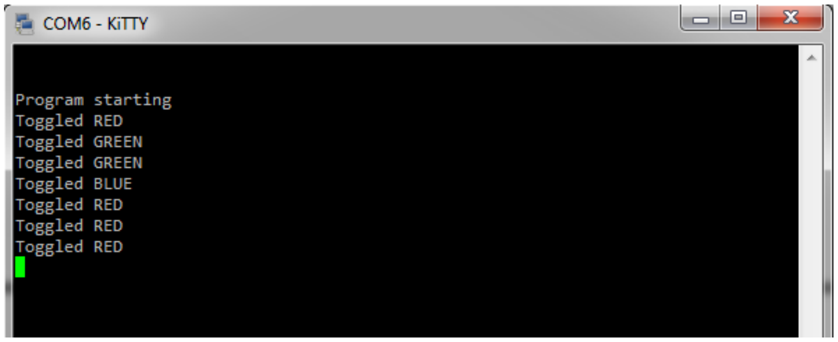
To solve this issue, open the CPU component and disable the resource that uses the pin you are trying to allocate. The location of the error is highlighted by red crosses in the Component Inspector. If the conflict is due to the JTAG debug interface, you could also change to the SWD mode[3].

[3] JTAG (Joint Test Action Group) and SWD (Single Wire Debug) are debug interfaces to the microcontroller that are used to upload code, set breakpoints, inspect memory, etc. Debug hardware (like OpenSDA) can select JTAG or SWD by generating the appropriate sequence of signals on the right pins. The selection of debug mode can also be pre-configured in software. SWD is generally preferred because it uses fewer pins. Our dev boards only provide the connections for SWD, and the extra JTAG pins are used for other purposes.

Write your code

Write the code to implement the task requirements. Some tips:

- Don't forget to click the "Generate Processor Expert Code" button after you modify any settings in Processor Expert.
- You should consider writing a function "send_string" that transmits a given string.
  - HINT—The strlen function returns the number of characters in a string. This function is declared in `<string.h>`.
- Use "`\r\n`" at the end of your messages to generate a Windows-style newline for display in your terminal emulator.
- There is a function "NegVal" in BitIO that toggles the output, but it is not generated by default. Right click on its entry in the Components view to enable it.
- The image below demonstrates how your user interface could operate. You do not need to replicate this exactly!



```
Program starting
Toggled RED
Toggled GREEN
Toggled GREEN
Toggled BLUE
Toggled RED
Toggled RED
Toggled RED
```

Assessment

To finish this lab, you must:

- Demonstrate to your prac tutor a working program that can use the serial communication to individually control each of the three LED colours.
- Show your prac tutor your GitHub website including your Lab 4 code.