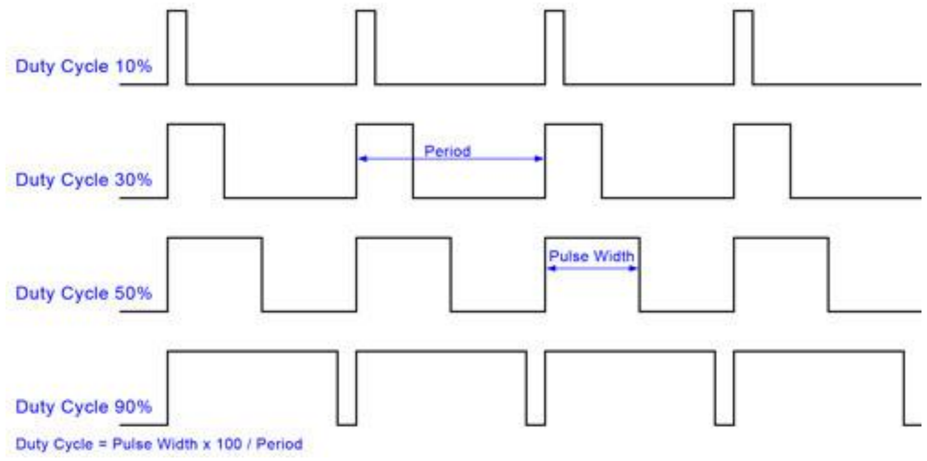
PWM and RC servos

Pulse width modulation is a name for a periodic digital signal where the duty cycle can be varied between 0 and 100%. Duty cycle is the ratio of the time that output is high to the length of the period. The inverse of the period is called frequency of the PWM signal. See the image below for examples of PWM signals.



Provided that PWM frequency is high enough for the application in question PWM can also be used for power control. With PWM one can think of the output voltage/current as the average voltage over the PWM period. For example when driving a led 40% duty cycle means that on the average the current is 40% of the maximum (100% duty cycle). PWM is also used to control RC servos, mostly for historical reasons.

In the following exercises you will use PWM both for power control and to control an RC servo. A modern microcontroller has integrated HW for generating PWM signal(s). PWM generation is based on programmable timers. Even though PWM signal generation principle is very similar though out different microcontrollers the details how to set up and operate PWM varies heavily from device to device. Mastering PWM (and timers) requires digging in to the details of the hardware. Typically processors user manuals and application notes are the best and most reliable source of information.

In the following exercises you need to consult UM10736.pdf (LPC1549 user manual), AN11538_SCTimer_PWM_Cookbook v5.0.pdf and header files in the chip library of our MCU. Copies of the pdf documents can be found in the workspace for your convenience. They can also be found on NXP website.

You will also learn about LPC1549 switch matrix which allows certain functions to be routed to any IO pin on the device. See chapter 8 in LPC1549 user manual and `swm_15xx.h` for more details. To route movable functions you need to call `Chip_SWM_MovablePortPinAssign` from the chip library.

Note: Your initializations must follow the board library initialization (SystemCoreClockUpdate() and BoardInit()). Board library configures some of the pins and functions and may override some of your settings if run after your own initializations.

LPC1549 has four PWM timers: two large ones and two small ones. Small ones have fewer input/output options but for simple PWM that we will do all of them have adequate features. To configure and operate PWM you need consults at least `sct_15xx.h` and `chip.h`.

Exercise 1

Write a program that sets up 1 kHz (period= 1 ms) PWM signal with initial duty cycle of 5%. Use Large SCT0 to generate the PWM signal. Connect the PWM output to the green led on LPCXpresso. Use on board switches to change the duty cycle. SW1 increases duty cycle (slowly), SW3 decreases (slowly) and when either of the switches is pressed simultaneously with SW2 the duty cycle changes rapidly. The program must print the duty cycle of the led (0 – 100%) on the ITM console.

Consult SCTimer cookbook chapter 6 and switch matrix chapter 8 in user manual plus the appropriate headers. Our processor runs at 72 MHz which you can see for example by printing the result of `Chip_Clock_GetSystemClockRate()`. Note that SCTimer cookbook examples assume a different system clock rate so some of the comments in the cookbook are in accurate when it comes to timing.

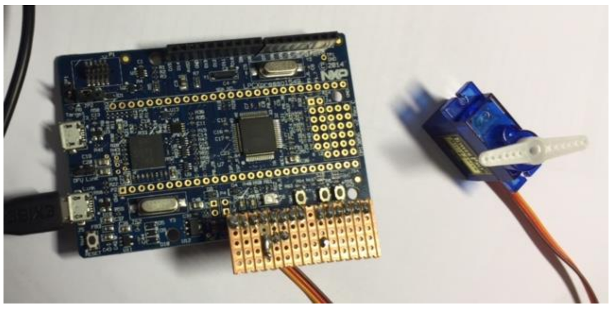
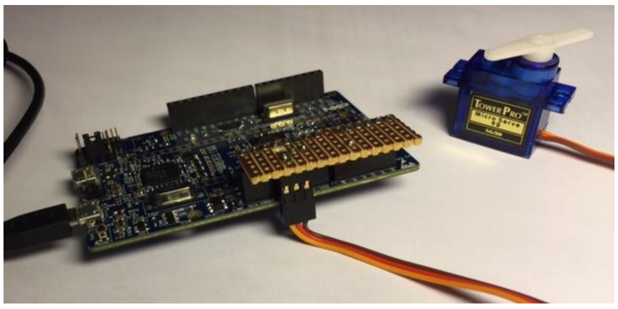
Remember that you must initialize SCT to power it on before you can configure or use it. See `Chip_SCT_Init()` for more details.

Hint: First test the cookbook example configuration as it is and only add IO pin routing with switch matrix. When you can control the led brightness with the example configuration then change PWM frequency to 1 kHz and make other required modifications.

Note: The on-board leds work with inverted logic. Setting the pin (=1) that controls the led switches the led off and clearing the pin (=0) switches the led on. This means that the longer SCT0_OUT0 is high the dimmer the led will be.

Exercise 2

Connect a servo adapter board (and the servo) to your LPCXpresso as shown below.

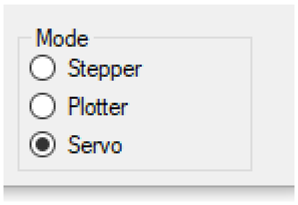
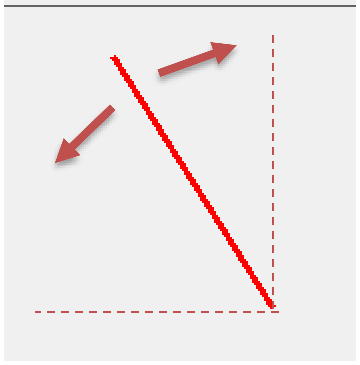
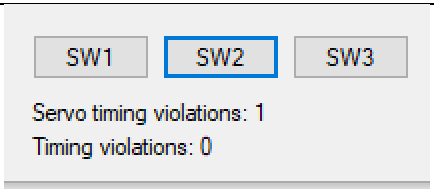



Write a program that drives the servo with PWM. The period must be 20 ms (50Hz) and the pulse width should be between 1 – 2 ms. A servo is at “center position” when the pulse width is 1.5 ms. Write a program that uses on board buttons to control the servo. SW1 and SW3 drive the servo in different direction and SW2 sets is back to center position. SW1 and SW3 must drive the servo slowly and allow the servo to be driven at any position between the edges.

The PWM input of the servo is connected to P0.8. If you are using simulator use pin P0.10

See https://learn.sparkfun.com/tutorials/hobby-servo-tutorial for a small tutorial and some warnings about driving a servo with pulses that exceed the range above.

Simulator:

Set simulator to Servo mode.  Mode ○ Stepper ○ Plotter [✓ Servo]	Simulated servo ARM has 90 degree movement range. 	 Servo timing violations: 1 Timing violations: 0 Servo timing violations is incremented if your PWM frequency is incorrect or your pulse width is out of allowed range. You can reset timing violation counters by pressing Reset on the simulator. Timing violations must stay at zero after your software is started. There may be couple of violations when the LPC board is reset but after that the number may not increase.

Exercise 3

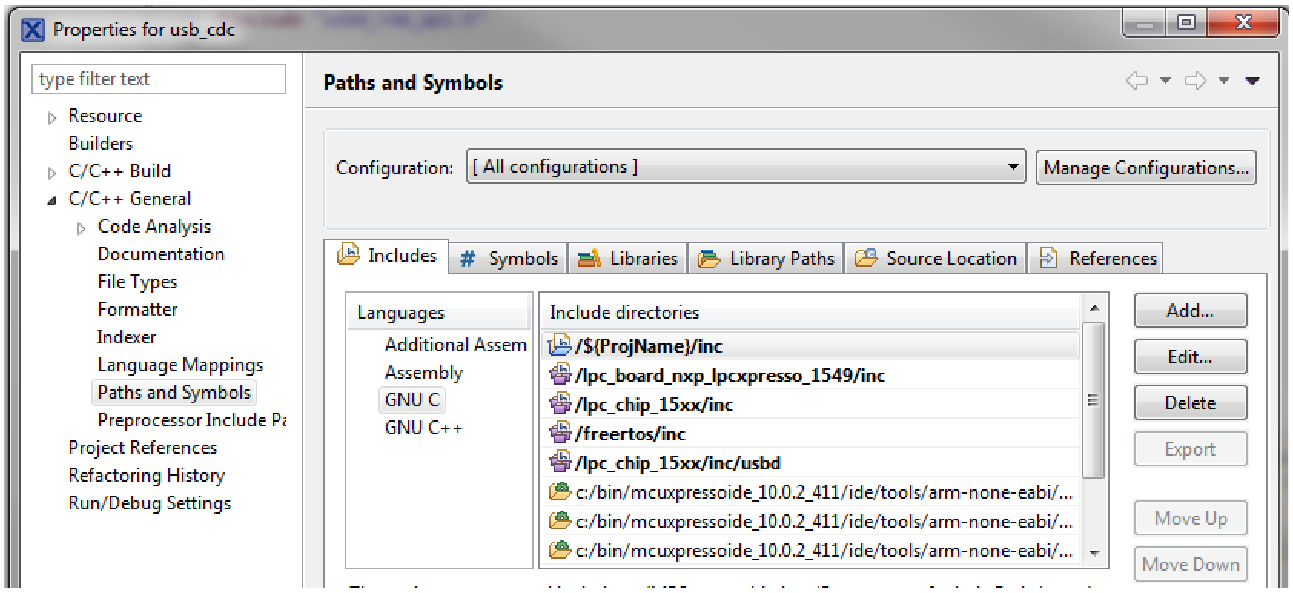
Use Large SCT0 as two 16-bit timers and Large SCT1 as two 16-bit timers. Use three of the timers to control the brightness of the on board leds. Each channel drives a led of one color.

Write a program that reads commands from the USB serial port (vcom_cdc) and sets color of the leds. The commands to set the color must follow HTML color notation. See: http://www.w3schools.com/colors/colors_picker.asp for details.

The command must be called: rgb which is followed by the color code (# + 6 hexadecimal digits).

Note: Using USB requires that you add /lpc_chip_15xx/inc/usbd directory to your project paths.

Switch to **Includes** tab and select **All configurations**. The IDE allows us to set both Debug and Release configurations on one go.



Properties for usb_cdc

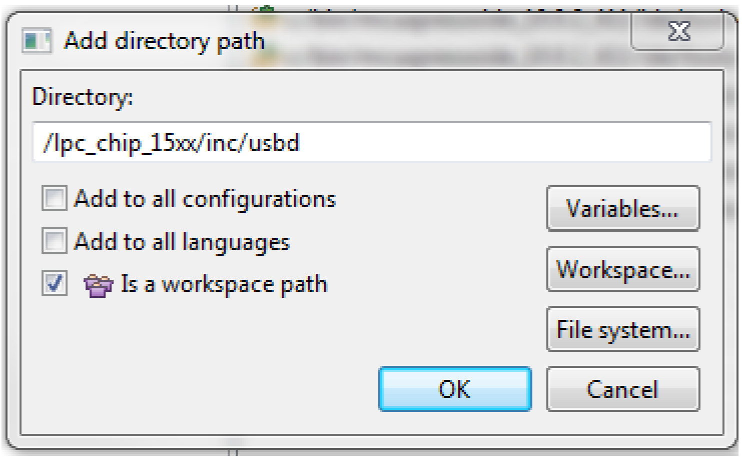
Paths and Symbols

Configuration: [All configurations]

Languages: Additional Assem, Assembly, GNU C, GNU C++

Include directories:
- /${ProjName}/inc
- /lpc_board_nxp_lpcxpresso_1549/inc
- /lpc_chip_15xx/inc
- /freertos/inc
- /lpc_chip_15xx/inc/usbd
- c:/bin/mcuxpressoide_10.0.2_411/ide/tools/arm-none-eabi/...
- c:/bin/mcuxpressoide_10.0.2_411/ide/tools/arm-none-eabi/...
- c:/bin/mcuxpressoide_10.0.2_411/ide/tools/arm-none-eabi/...

Select one language at time and add the directoriy to each language (there are 4 languages listed). Remember to check **Is a workspace path**.



Add directory path

Directory: /lpc_chip_15xx/inc/usbd

[ ] Add to all configurations
[ ] Add to all languages
[✓] Is a workspace path