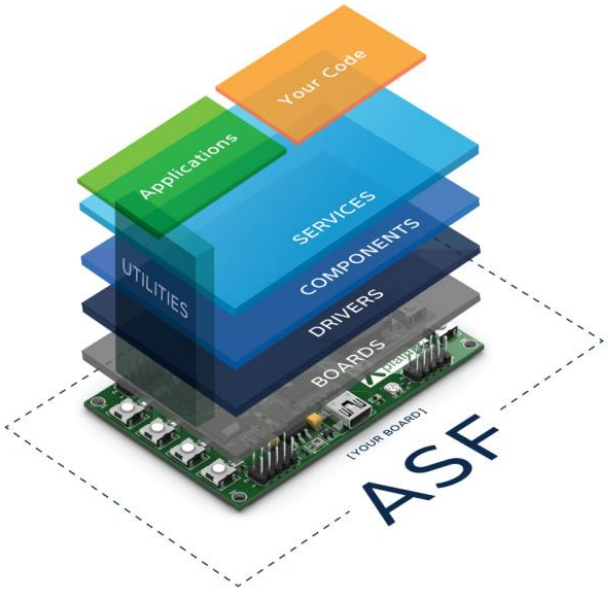
Introduction to Atmel Software Framework

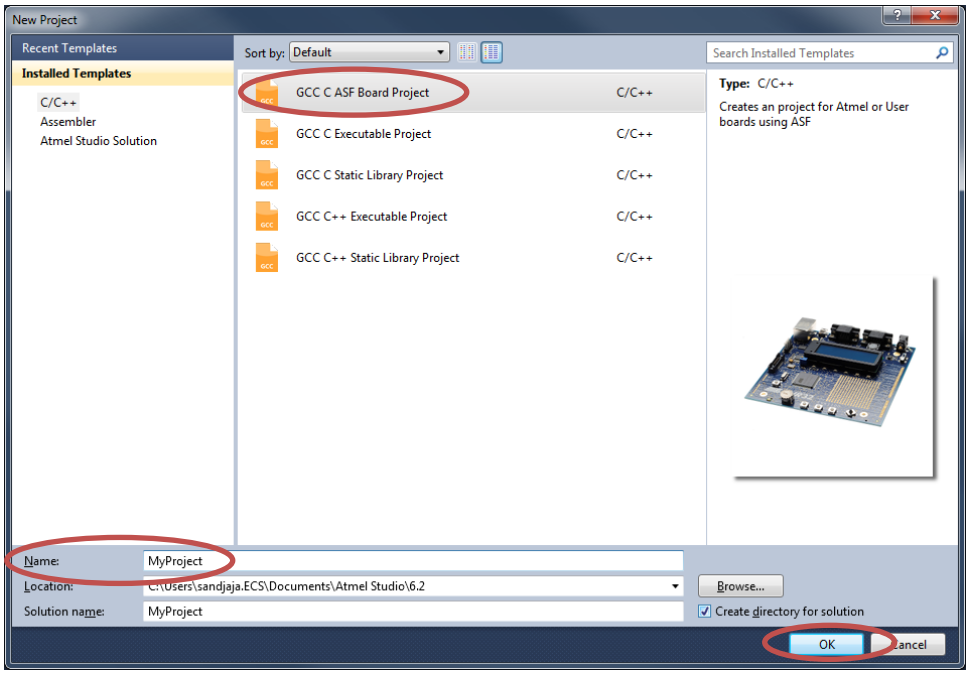
ELC 4438 Lab Manual

February 1st, 2016

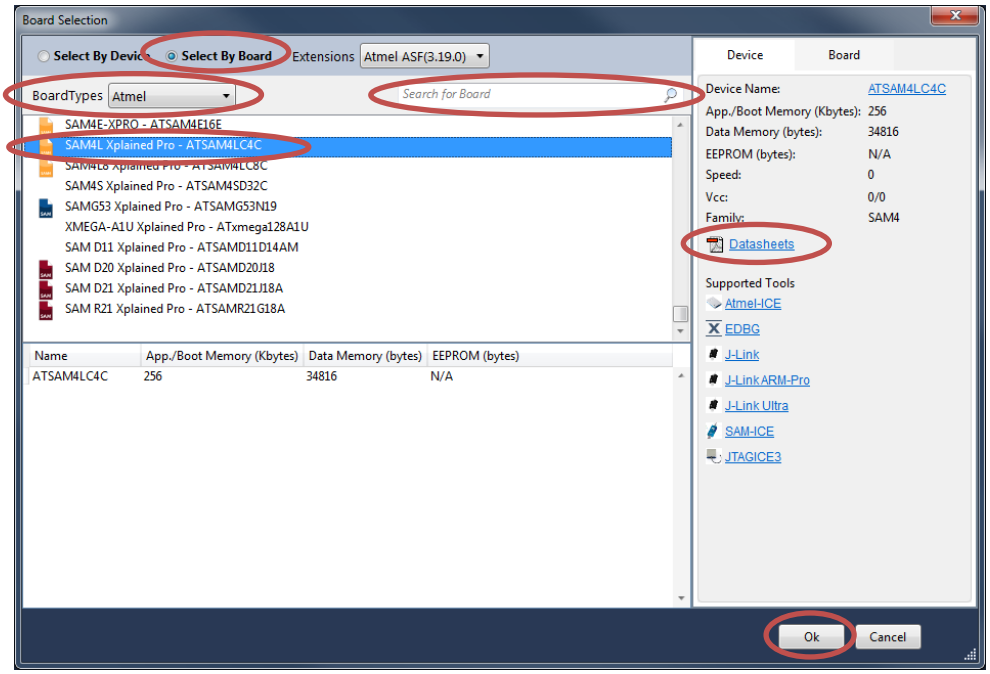


1. Please watch http://goo.gl/jeikPh and have a quick look at http://goo.gl/ILTQuB before you start.

   Select **File - New Project, GCC ASF Board Project**. Give a name to your project then click **OK**.



2. Choose **Select By Board** and BoardTypes **Atmel**.
   Type **SAM4L** in the *Search for Board* window.
   Choose **SAM4L Xplained Pro – ATSAM4LC4C**.
   Download and read the **datasheets**.
   Click **OK**. (Please be patient as it may take a few minutes.)



3. Open the source in **Solution Explorer** – **src** directory – **main.c**.

   You can use left-click on any function (ioport_get_pin_level/ ioport_set_pin_level) and choose **Goto Implementation** (ALT-G) to see its implementation.

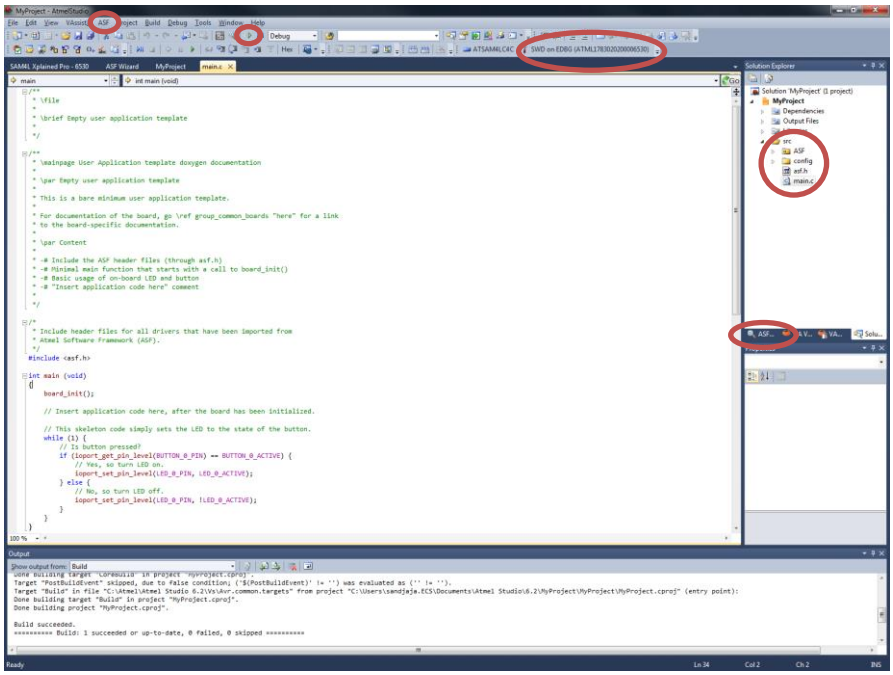
   Click "**No tool"** on the menu toolbar and select "**EDBG"** for debugger/ programmer in the Tool Tab.

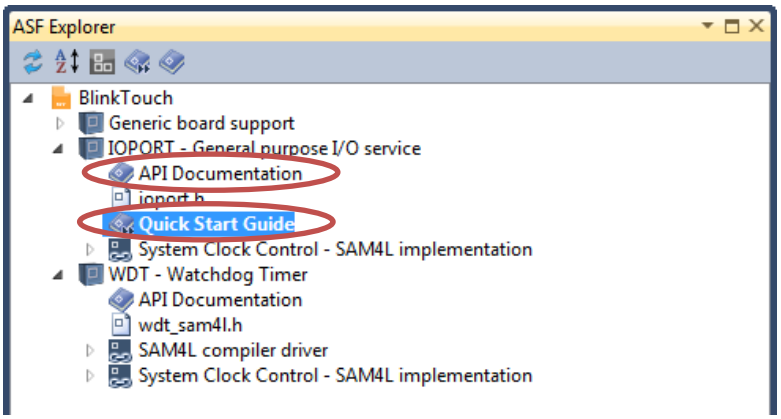
   Click "**Start without debugging"** (Green triangle) in the menu toolbar.

   Verify that the code is running on the board by using SW0 to turn on LED0.

4. Click **ASF Explorer** and the white triangle before the module name to see **API documentation** for each module and **Quick Start Guide** for several modules.

   Read **IOPORT** API documentation and Quick Start Guide.



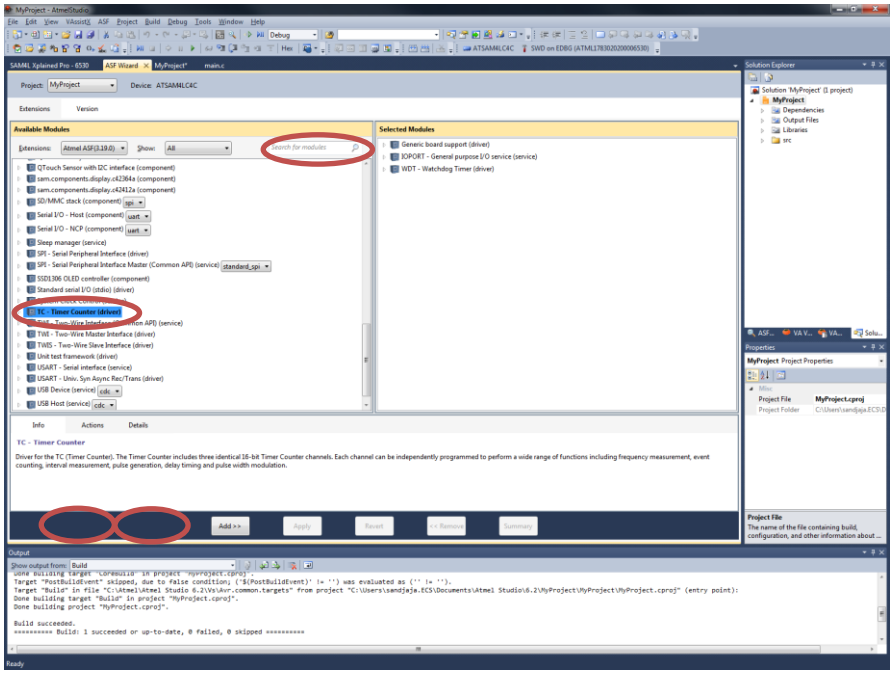


5. You can add modules (driver/component/service) through menu bar **ASF - ASF Wizard**.

   Add **Standard serial I/O (stdio) (driver)** and **System Clock Control (service)**. Click **Apply** then **OK**.

   Please read their API documentation and Quick Start Guide. Do not forget to adjust the configuration files in **config** directory within **Solution Explorer**.

   Compare the configuration files with the ones of the previous lab (Introduction to Atmel SAM4L Xplained Pro).



6. Add procedure `configure_console(void)` from the previous lab (Introduction to Atmel SAM4L Xplained Pro).

   Call it from `main(void)` after `board_init()` and `sysclk_init()`.

   Print something in PC terminal such as “Hello World” after the initialization. When SW0 is pressed, change the output and print “ON”.

7. Modify the program to use **`SysTick_Config`** and **`SysTick_Handler`**.

   Refer to the code on the following page.

```c
#include <asf.h>

/** Global g_ul_ms_ticks in milliseconds since start of application */
volatile uint32_t g_ul_ms_ticks = 0;
volatile uint32_t time_delay = 500;

#define BLINK_PERIOD        1000

static void configure_console(void)
{
    const usart_serial_options_t uart_serial_options = {
           .baudrate = (115200UL),
           .charlength = US_MR_CHRL_8_BIT,
           .paritytype = US_MR_PAR_NO,
           .stopbits = US_MR_NBSTOP_1_BIT,
    };

    /* Configure console UART. */
    stdio_serial_init(COM_PORT_USART, &uart_serial_options);
}
void SysTick_Handler(void)
{
    g_ul_ms_ticks++;
    //printf("Ticks");
}

int main (void)
{
    board_init();
    sysclk_init();
    configure_console();

    /* Configure systick for 1 ms */
    puts("Configure system tick to get 1ms tick period.\r");
    if (SysTick_Config(sysclk_get_cpu_hz() / BLINK_PERIOD)) {
           puts("-F- Systick configuration error\r");
           while (1);
    }

    while (1) {
           // Is button pressed?
           if (ioport_get_pin_level(BUTTON_0_PIN) == BUTTON_0_ACTIVE) {
                  // Yes, so turn LED on.
                  //ioport_set_pin_level(LED_0_PIN, LED_0_ACTIVE);
                  time_delay = 500;
           } else {
                  // No, so turn LED off.
                  //ioport_set_pin_level(LED_0_PIN, !LED_0_ACTIVE);
                  time_delay = 2000;
           }

           if (g_ul_ms_ticks>=time_delay)
           {
                  g_ul_ms_ticks = 0;
                  LED_Toggle(LED0);
           }
    }
}
```

8. Load new example project: **SAM4L QTouch Example – SAM4L Xplained Pro**

   Read the code main.c in Atmel Studio to understand how it works.

   Refer to the **SAM4L Datasheet** (Chapter 19) to understand the Asynchronous Timer (AST).
   http://www.atmel.com/Images/Atmel-42023-ARM-Microcontroller-ATSAM4L-Low-Power-LCD_Datasheet.pdf

   Using the same procedure, run the program and use the onboard Qtouch button.

9. Referring to both of the sample programs above, modify and write a program that blinks a LED at different frequencies.

   You have two buttons (SW0 and Qtouch) to control the blinking frequency. If no button is pressed, the LED blinks at 0.5 Hz. When the SW0 button is pressed, the LED blinks at 1 Hz. When the QTouch button is pressed, the LED blinks faster at 2 Hz. If both buttons are pressed, the LED blinks at 4 Hz.

   Save your code and show your experiment result to your TA.

10. Extra: Create the same program using timer counter (TC) instead of system clock tick.

    Refer to the SAM4L Datasheet (Chapter 30) to understand the Timer Counter.
    http://www.atmel.com/Images/Atmel-42023-ARM-Microcontroller-ATSAM4L-Low-Power-LCD_Datasheet.pdf