1 AVR32 Software Framework

Programming the AVR32 chip from scratch is time consuming and complex. To make it faster and easier to develop software there is a Software Framework which contain drivers for the EVK1100. The Framework provide good functionality at the expense of less flexibility of the more exotic options. A good enough trade-off in many cases.

When creating a AVR32 project in Atmel Studio 6 some parts of the Software Framework are already used, e.g. led.c/led.h in the folder:

{project}/src/ASF/AVR32/BOARDS/EVK1100/ that, for instance, provide the *LED_Set_Intensity()* function which uses Pulse Width Modulation (PWM) to control LEDs.

1.1 Analog to Digital Converter (ADC)

Here we will use the Analog to Digital Converter (ADC) module of the MCU to read the analog potentiometer on the EVK1100. The light sensor, temperature sensor and externally connected analog signals can also be read with the ADC. Your task is to play around with the code below.

- Create a new AVR32 project from template.
- To use the Software Framework, first select the current project in the Project Explorer, then right click on "ASF Wizard".
- Add the drivers you need:
  - Add ADC and INTC for this assignment.
    - ADC is needed when using the Analog to Digital Converter module of the MCU.
    - INTC is needed for the exception handling when using the Software Framework and is usually preselected by default.

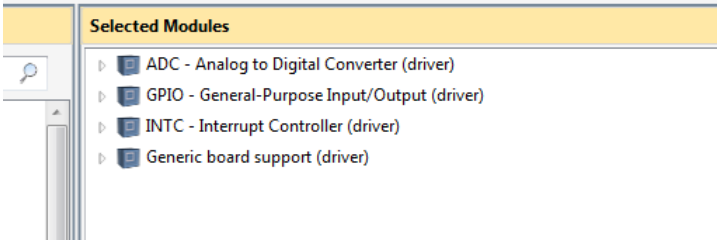

Figure 1: Software Framework - Add Driver

- Click "Apply", since we are not going to select any components etc. right now.
- In the *{project}/src/ASF/AVR32* directory a *DRIVERS/ADC* directory will appear which contain two files, *adc.c* and *adc.h*.
- Import the *main.c* from the lab source directory into your project *src/* directory.

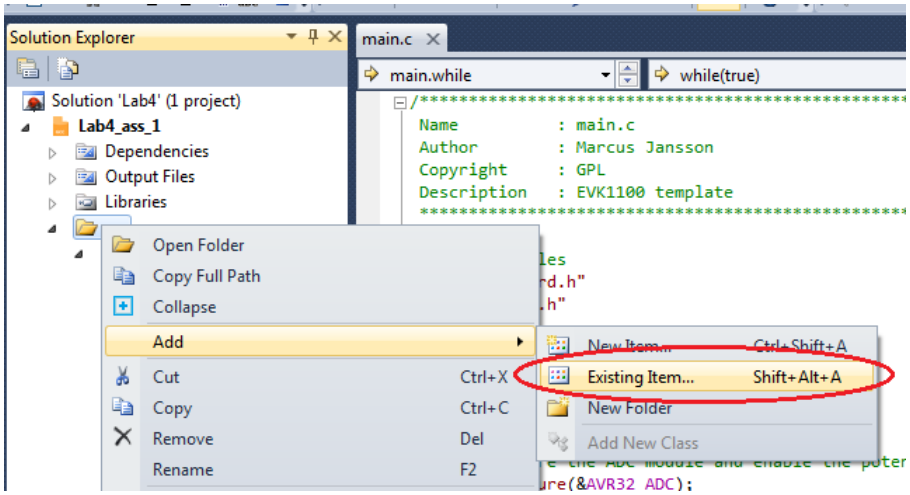

Figure 2: Import into

- Compile and program the MCU in AVR32 Studio.
- Play around with the source code, make changes and investigate until you feel comfortable.

```
/*****************************************************
Name          : main.c
Author        : Marcus Jansson
Copyright     : GPL
Description   : EVK1100 template
*****************************************************/
// Include Files
#include "board.h"
#include "adc.h"

int main(void)
{
    int pot_value = 0;
    int brightness_LED5 = 0;
```

```
    // Configure the ADC module and enable the
    // potentiometer channel
    adc_configure(&AVR32_ADC);
    adc_enable(&AVR32_ADC, ADC_POTENTIOMETER_CHANNEL);

    while(true) {
        // Start a ADC sampling of all active channels
        adc_start(&AVR32_ADC);

        // Get the potentiometer value
        pot_value = adc_get_value(&AVR32_ADC,
                                  ADC_POTENTIOMETER_CHANNEL);

        // Convert the potentiometer value to a value
        // between 0-255
        brightness_LED5 = pot_value * 255 / 1024;

        // Set the intensity of LED5
        LED_Set_Intensity(LED5, brightness_LED5);
    }

    // Never return
    while(true);
}
```

1.1.1 Some notes

ADC_LIGHT_CHANNEL and *ADC_TEMPERATURE_CHANNEL* are defined for reading the light and temperature sensor of the EVK1100.

The *LED_Set_Intensity()* function can only be used with LED4-LED7, which are the the red/green bi-color LEDs on the EVK1100.

1.2 LCD Display

We will now examine the LCD character display of the EVK1100. It is called DIP204 and have four rows with 20 character each.

1.2.1 DIP204 20*4 Character display

- Add from the Software Framework:
  - CPU - Cycle counter
  - GPIO - General Purpose I/O Controller
  - PM - Power Management
  - PWM - Pulse Width Modulation
  - SPI - Serial Peripheral Interface

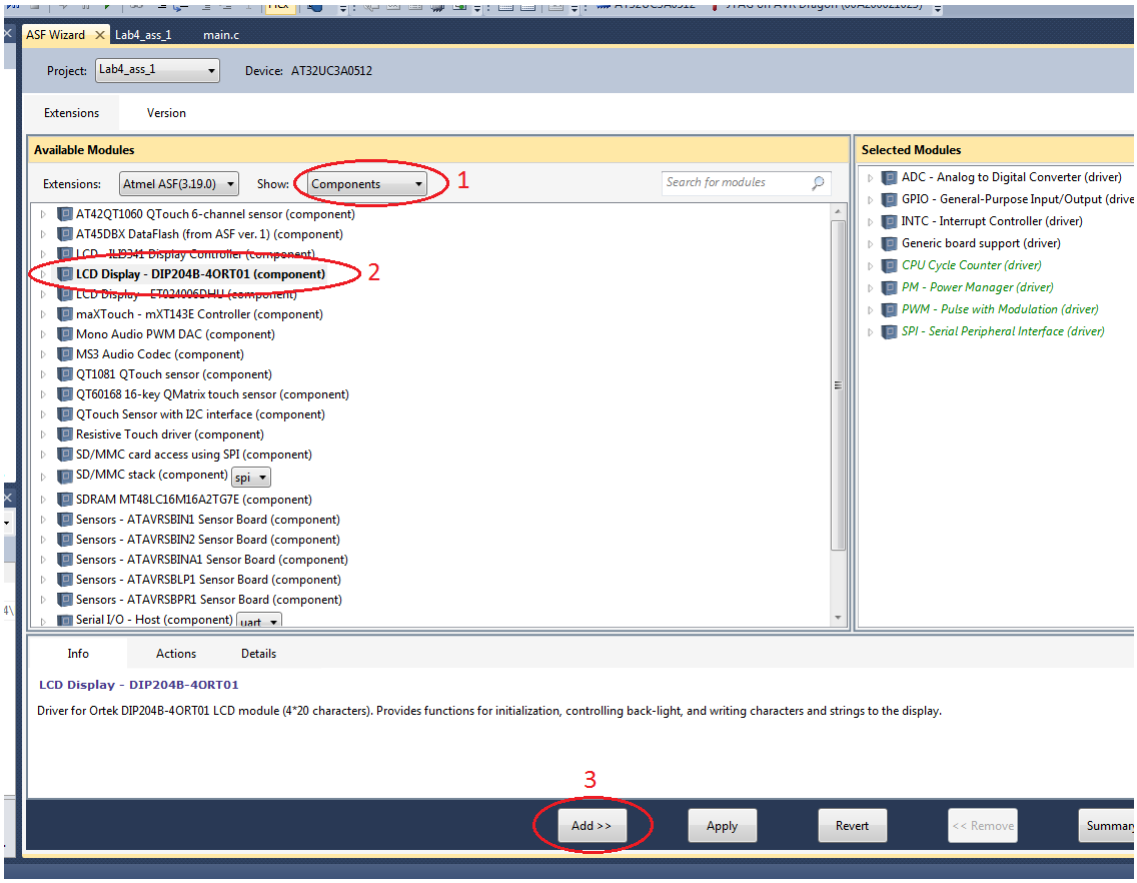

Figure 3: Add display component

- Switch to the "Components" filter (1).
- Select the Display (LCD Display - DIP204B-4ORT01) component (2).
- Click "Add" (3).
- Click "Apply".
- Import the *display_init.c* and *display_init.h* from the lab source directory.
  - This code uses the software framework to initialize the DIP204 character display on the EVK1100. It will set the CPU frequency to the 12 Mhz of oscillator 0 and calibrate the delay functions to this frequency. The DIP204 display will initialize on the SPI bus.
- Include the *display_init.h* into your main.c source file.

Now you can call the *display_init()* function and start using the LCD display.

```
void display_init(void);
```

Useful functions from the DIP204 Software Framefork are found in your *{project}/src/ASF/avr32/components/display/dip204/dip204.h*.
Some are briefly show below:

- *void dip204_set_cursor_position(unsigned char x, unsigned char y);*
  - Move the cursor to a given position [x, y]. The positions go from [1,1] to [20,4]
- *void dip204_write_string(const char * string);*
  - Write the string at the current position.
- *void dip204_printf_string(const char *format, ...);*
  - Write a formated string at the current position.
- *void dip204_show_cursor(void);*
  - Show a blinking cursor.
- *void dip204_hide_cursor(void);*
  - Hide the blinking cursor.
- *void dip204_clear_display(void);*
  - Clear the entire display.

1.3 Timer Capture

Here are the main functions and data structures we need to use for this assignment.

```
// Options for capture mode.
tc_capture_opt_t capture_opt =
{
.channel  = channel,                  // Channel selection

.ldrb     = TC_SEL_NO_EDGE,           // RB loading selection
.ldra     = TC_SEL_FALLING_EDGE,      // RA loading selection

.cpctrg   = TC_NO_TRIGGER_COMPARE_RC,// RC compare trigger enable
.abetrg   = TC_EXT_TRIG_SEL_TIOA,     // TIOA or TIOB external trigger selection
.etrgedg  = TC_SEL_FALLING_EDGE,      // External trigger edge selection
```

```
.ldbdis    = false,                    // Counter clock disable with RB loading
.ldbstop   = false,                    // Counter clock stopped with RB loading

.burst     = TC_BURST_NOT_GATED,       // Burst signal selection
.clki      = TC_CLOCK_RISING_EDGE,     // Clock inversion
.tcclks    = TC_CLOCK_SOURCE_TC4       // Internal source clock 4 ( fPBA/32 )
};

tc_init_capture ();
```

Useful functions can be found in *tc.h*.
Some are briefly show below:

- *int tc_init_capture (volatile avr32_tc_t *tc, const tc_capture_opt_t *opt);*
  Sets options for timer/counter capture initialization.
  - Sets options for timer/counter capture initialization.
- *int tc_init_waveform (volatile avr32_tc_t *tc, const tc_waveform_opt_t *opt);*
  - Sets options for timer/counter waveform initialization..
- *int tc_read_ra (volatile avr32_tc_t *tc, unsigned int channel);*
  - Reads the channel's RA register and returns the value.
- *int tc_write_ra (volatile avr32_tc_t *tc, unsigned int channel, unsigned short value);*
  - Writes a value to the channel's RA register.
- *int tc_start (volatile avr32_tc_t *tc, unsigned int channel);*
  - Starts a timer/counter.
- *int tc_stop (volatile avr32_tc_t *tc, unsigned int channel);*
  - Stops a timer/counter.

2 Assignments

1. Write a program that display the potentiometer, the light sensor and temperature sensor values on the DIP204 display. If the readings are jittering, make average readings.

3 Report

Explain the mechanism of changing the LED intensity in the LED_Set_Intensity() function.

4 Optional Assignments

1. Write a simple game on the EVK1100. Do not feel limited by the small display area. Scroll the screen if you need more space.

   Examples:

   (a) Snake
   (b) Pong
   (c) Space invaders
   (d) Breakout
   (e) Pacman
   (f) Maze

   You are welcome to make any other game that is not included in the above list. However, the game should be of equal complexity as compared to the games in the above list. Therefore, you must discuss the new game with the lab assistant and get approval before implementing it.

2. Write a program that set the intensity of a LED at 30%, 60% and 90% by setting the duty cycle of a PWM signal. The timer module must be set in waveform mode.