1. Open Atmel Studio 6.2
2. Start a new project

Board: Atmel SAM4L

Run the program – make sure everything is working (LED is on when switch is pressed).

1. Add **TWI- Two-Wire Master Interface (driver),** **System Clock Control (service)**, and **Standard serial I/O (stdio) (driver)**
2. Add new header file in the config directory, name it **conf\_twi.h**. Use this code to configure Two Wire Interface.

#ifndef CONF\_TWI\_H\_INCLUDED

#define CONF\_TWI\_H\_INCLUDED

/\*\* TWIM Interrupt Number \*/

#define EXAMPLE\_TWIM\_IRQn TWIM3\_IRQn

/\*\* TWIM Module Used \*/

#define EXAMPLE\_TWIM TWIM3

/\*\* Target's TWI address \*/

#define TARGET\_ADDRESS 0x50

/\*\* Internal Address length \*/

#define TARGET\_ADDR\_LGT 2

/\*\* Internal Address \*/

#define VIRTUALMEM\_ADDR 0x0

/\*\* Speed of TWI \*/

#define TWIM\_MASTER\_SPEED TWI\_STD\_MODE\_SPEED

/\*\* TWIM Interrupt Handler \*/

#define EXAMPLE\_TWIM\_Handler TWIM3\_Handler

#endif /\* CONF\_EXAMPLE\_H\_INCLUDED \*/

1. Enable **TWIM3** pins and **COM PORT** at the **conf\_board.h**

/\*\* Enable Com Port. \*/

#define CONF\_BOARD\_COM\_PORT

/\*\* Enable TWIM3 pins. \*/

#define CONF\_BOARD\_TWIMS3

1. Use this code as **conf\_clock.h**

#ifndef CONF\_CLOCK\_H\_INCLUDED

#define CONF\_CLOCK\_H\_INCLUDED

#define CONFIG\_SYSCLK\_SOURCE SYSCLK\_SRC\_DFLL

#define CONFIG\_SYSCLK\_CPU\_DIV 0

#define CONFIG\_SYSCLK\_PBA\_DIV 0

#define CONFIG\_SYSCLK\_PBB\_DIV 0

#define CONFIG\_SYSCLK\_PBC\_DIV 2

#define CONFIG\_SYSCLK\_PBD\_DIV 2

#define CONFIG\_DFLL0\_SOURCE GENCLK\_SRC\_OSC32K

#define CONFIG\_DFLL0\_FREQ 48000000UL

#define CONFIG\_DFLL0\_MUL (CONFIG\_DFLL0\_FREQ / BOARD\_OSC32\_HZ)

#define CONFIG\_DFLL0\_DIV 1

#endif /\* CONF\_CLOCK\_H\_INCLUDED \*/

1. Configure the serial at conf\_uart\_serial.h

#ifndef CONF\_USART\_SERIAL\_H

#define CONF\_USART\_SERIAL\_H

/\*\* USART Interface \*/

#define CONF\_UART USART1

/\*\* Baudrate setting \*/

#define CONF\_UART\_BAUDRATE 115200

/\*\* Character length setting \*/

#define CONF\_UART\_CHAR\_LENGTH US\_MR\_CHRL\_8\_BIT

/\*\* Parity setting \*/

#define CONF\_UART\_PARITY US\_MR\_PAR\_NO

/\*\* Stop bits setting \*/

#define CONF\_UART\_STOP\_BITS US\_MR\_NBSTOP\_1\_BIT

#endif/\* CONF\_USART\_SERIAL\_H\_INCLUDED \*/

1. Add static void configure\_console(void) and before int main (void) in the main.c

#define BLINK\_PERIOD 1000

volatile uint32\_t g\_ul\_ms\_ticks = 0;

void SysTick\_Handler(void)

{

 g\_ul\_ms\_ticks++;

 //printf("%d\r",g\_ul\_ms\_ticks);

}

static void configure\_console(void)

{

 const usart\_serial\_options\_t uart\_serial\_options = {

 .baudrate = CONF\_UART\_BAUDRATE,

 #ifdef CONF\_UART\_CHAR\_LENGTH

 .charlength = CONF\_UART\_CHAR\_LENGTH,

 #endif

 .paritytype = CONF\_UART\_PARITY,

 #ifdef CONF\_UART\_STOP\_BITS

 .stopbits = CONF\_UART\_STOP\_BITS,

 #endif

 };

 /\* Configure console UART. \*/

 stdio\_serial\_init(CONF\_UART, &uart\_serial\_options);

}

1. Add this code initialize sysclk and configure console

sysclk\_init();

/\* Initialize the console USART \*/

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);

}

Try to print this and check it at the PuTTY:

/\* Output example information \*/

printf("-- Smart Signage --\r\n");

printf("-- %s\n\r", BOARD\_NAME);

printf("-- Compiled: %s %s --\n\r", \_\_DATE\_\_, \_\_TIME\_\_);

1. Include conf\_twi.h and sleepmgr.h

#include "conf\_twi.h"

#include "sleepmgr.h"

1. Add the static status\_code\_t init\_test(void) and its data structure and see it initialized successfully
2. Add static status\_code\_t write\_test(void), static status\_code\_t read\_test(void), static status\_code\_t read\_temperature(void), and static void print\_data(void)
3. Try to perform Write Test and Read Test. After try to pool the temperature data.
4. Congratulation you have been built the Temperature app from scratch. Now we will add the OLED display
5. Add **SSD1306** from ASF Wizard
6. Set the config\_ssd1306.h

#ifndef CONF\_SSD1306\_H\_INCLUDED

#define CONF\_SSD1306\_H\_INCLUDED

// Interface configuration for SAM4L Xplained Pro

#define SSD1306\_SPI\_INTERFACE

#define SSD1306\_SPI SPI

#define SSD1306\_DC\_PIN UG\_2832HSWEG04\_DATA\_CMD\_GPIO

#define SSD1306\_RES\_PIN UG\_2832HSWEG04\_RESET\_GPIO

#define SSD1306\_CS\_PIN UG\_2832HSWEG04\_SS

// Minimum clock period is 50ns@3.3V -> max frequency is 20MHz

#define SSD1306\_CLOCK\_SPEED UG\_2832HSWEG04\_BAUDRATE

#define SSD1306\_DISPLAY\_CONTRAST\_MAX 40

#define SSD1306\_DISPLAY\_CONTRAST\_MIN 30

#endif /\* CONF\_SSD1306\_H\_INCLUDED \*/

1. Add this code to sam4l\_xplained\_pro.h before the AT86RFX transceiver in ASF/sam/boards/sam4l\_xplained\_pro.h

//! \name OLED

//@{

/\*\* OLED command/data select pin \*/

#define UG\_2832HSWEG04\_DATA\_CMD\_GPIO (EXT2\_PIN\_5)

/\*\* OLED reset pin \*/

#define UG\_2832HSWEG04\_RESET\_GPIO (EXT2\_PIN\_10)

/\*\* OLED SPI configuration \*/

#define UG\_2832HSWEG04\_SS 2

#define UG\_2832HSWEG04\_BAUDRATE 5000000

//! \name OLED dimensions

//@{

#define LCD\_WIDTH\_PIXELS (128)

#define LCD\_HEIGHT\_PIXELS (32)

//@}

//@}

1. Enable the OLED screen at conf\_board.h

/\*\* Enable the OLED screen. \*/

#define CONF\_BOARD\_SPI

#define CONF\_BOARD\_SPI\_NPCS2

Add this code to init.c in ASF/sam/boards/init.h

#ifdef CONF\_BOARD\_OLED\_UG\_2832HSWEG04

 ioport\_set\_pin\_dir(UG\_2832HSWEG04\_DATA\_CMD\_GPIO, IOPORT\_DIR\_OUTPUT);

 ioport\_set\_pin\_mode(UG\_2832HSWEG04\_DATA\_CMD\_GPIO, IOPORT\_MODE\_PULLUP);

 ioport\_set\_pin\_dir(UG\_2832HSWEG04\_RESET\_GPIO, IOPORT\_DIR\_OUTPUT);

 ioport\_set\_pin\_mode(UG\_2832HSWEG04\_RESET\_GPIO, IOPORT\_MODE\_PULLUP);

#endif

1. You are ready to use the OLED display. Now we will add the Touch Sensor
2. Add **AST-Asynchronous Timer (driver)** and **QTouch Library for SAM4L (service)** from ASF Wizard
3. Create a new file at the config directory named conf\_qtouch.h and type this code:

#ifndef CONF\_QTOUCH\_H\_

#define CONF\_QTOUCH\_H\_

#define EXAMPLE\_UPDATE\_PERIOD (5u)

#define EXAMPLE\_KEY\_ID (0u)

#endif /\* CONF\_QTOUCH\_H\_ \*/

1. Add **qtouch** folder to **src** and add two files from qtouch example (**QDebugSettings.h** and **touch.c**)
2. Configure ast.h to activate the PER

#ifndef CONF\_AST\_H\_INCLUDED

#define CONF\_AST\_H\_INCLUDED

#define AST\_PER\_ENABLE

#endif /\* CONF\_AST\_H\_INCLUDED \*/

1. Make sure in the conf\_clock.h

#define CONFIG\_SYSCLK\_PBC\_DIV (2)

#define CONFIG\_SYSCLK\_PBD\_DIV (2)

1. Include conf\_qtouch.h in the main.c

#include "conf\_qtouch.h"

1. Add static void ast\_per\_callback(void) and static void example\_qtouch\_init(void) before the int main (void) in the main.c
2. Now, you can initialize and use qtouch in the main.c

/\* Initialize Qtouch \*/

example\_qtouch\_init();

while (1) {

 touch\_sensors\_measure();

 if (GET\_QT\_SENSOR\_STATE(EXAMPLE\_KEY\_ID)) {

 LED\_On(LED0);

 }

 else {

 LED\_Off(LED0);

 }

 }