

---

***SAM4S Xplained Pro Hands-On Tutorial and  
Tasks***

***Tutorial Document***

## Table of Contents

Introduction .....	3
1. TASK-1 Software Installation.....	3
1.1 Creating Example Project and exploring project architecture.....	3
1.2 Board connection and Example project execution .....	6
2. TASK :2 Creating OLED display application to display txt .....	7
2.1 Result of TASK-2 .....	10
3. TASK :3 Creating OLED application and check text buffer in watch window (debug mode).....	10
3.1 Result of TASK-3 .....	10
4. TASK :4 installing serial terminal application.....	11
5. TASK :5 installing serial terminal application.....	11
5.1 Result of TASK-5 .....	13
6. TASK :6 Code Exercises.....	14
6.1 Swap two variables without using intermediate variable .....	14
Result .....	14
6.2 Count number of bits set in a byte. ....	14
Result .....	15
6.2 Check n-th bit of 32 bit value is set (1) or not.....	15
Result .....	16

## Introduction

This document describes the basic setup and guidelines to get starting with Atmel studio and ASF based application development.

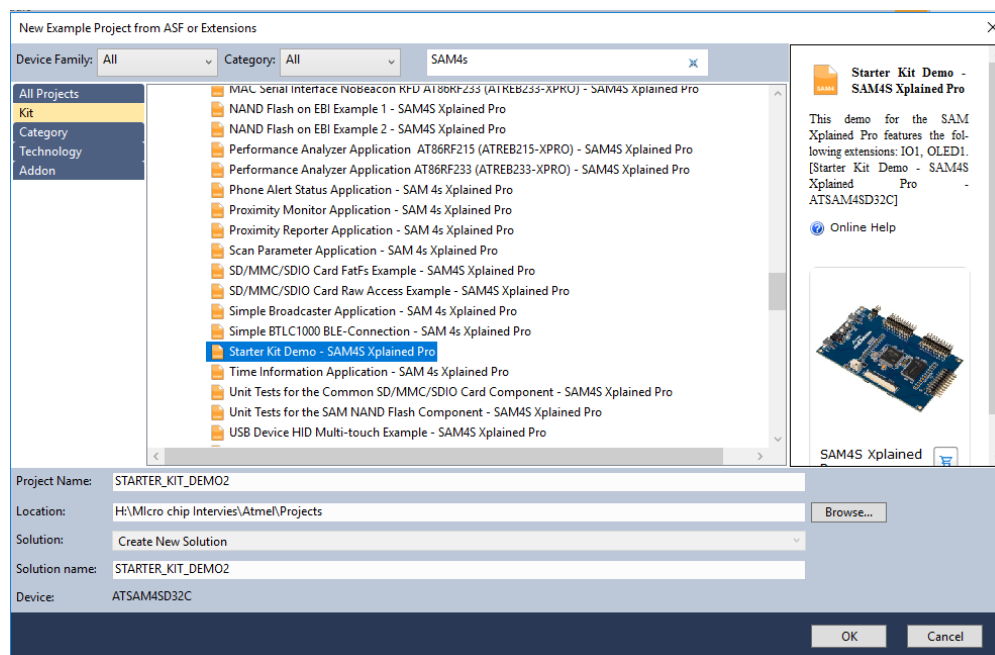
## 1. TASK-1 Software Installation

Download and install Atmel studio 7 and ASF from

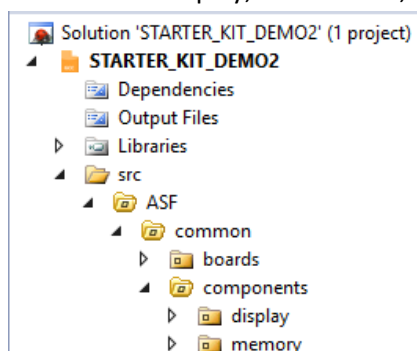
<http://www.atmel.com/tools/atmelstudio.aspx#download>

### 1.1 Creating Example Project and exploring project architecture

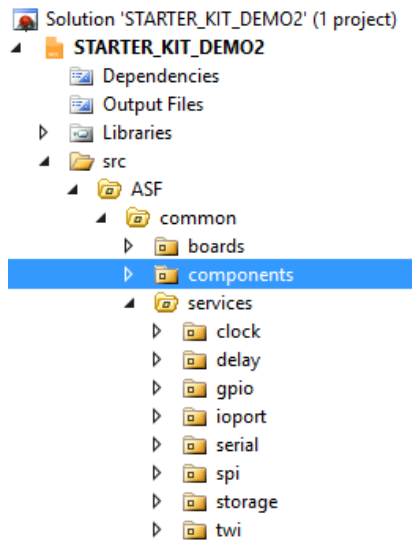
1. Launch Atmel studio and select “New Example Project..” in the dialogue select “Kit” and search for “SAM4s Xplained Pro” → “Starter Kit Demo” project and open.



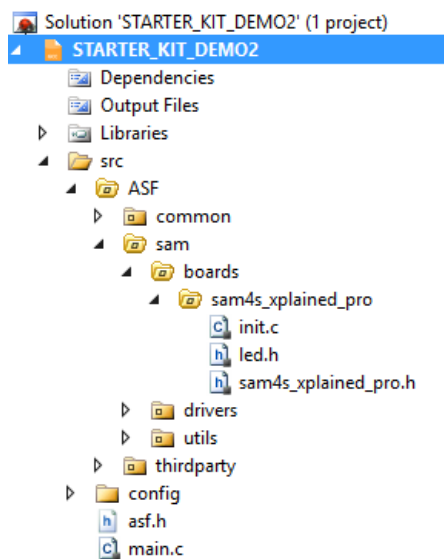
2. In solution Explorer,
  - The folder “Src→ASF→common→components” contains the external module related libraries like Display, NAND Flash, Image sensor and more.



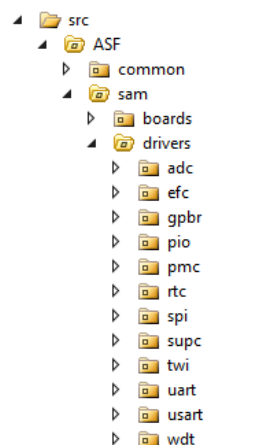
- The folder “Src→ASF→common→services” contains peripheral related API libraries like UART,USB, SPI and more.



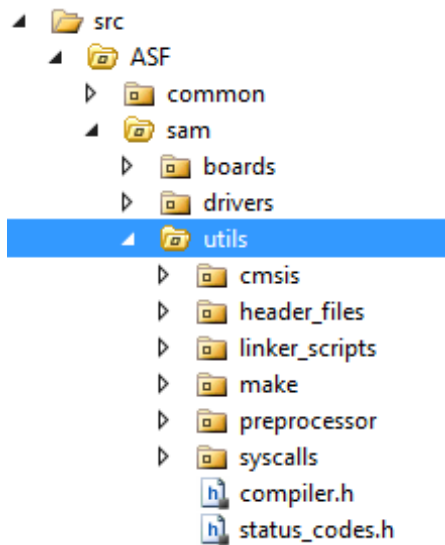
- The folder “Src→ASF→sam→boards” contains board specific configuraion and initialization files.



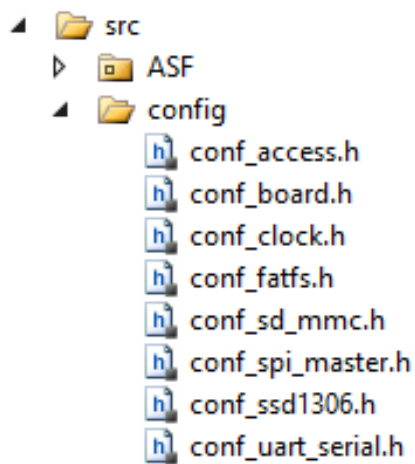
- The folder “Src→ASF→sam→drivers” contains board specific low level peripheral drivers.



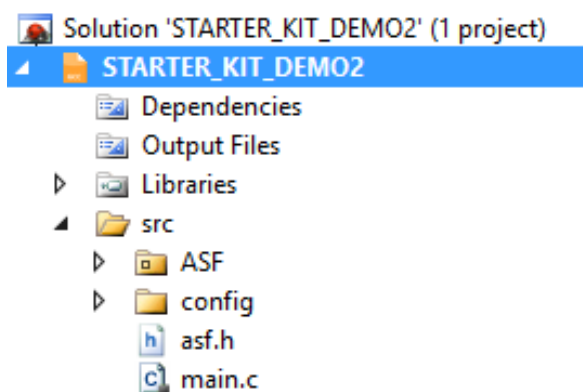
- The folder “Src→ASF→sam→utils” contains compiler related configuration files.



- The folder “Src→config” contains macro configuration files for board and peripherals.

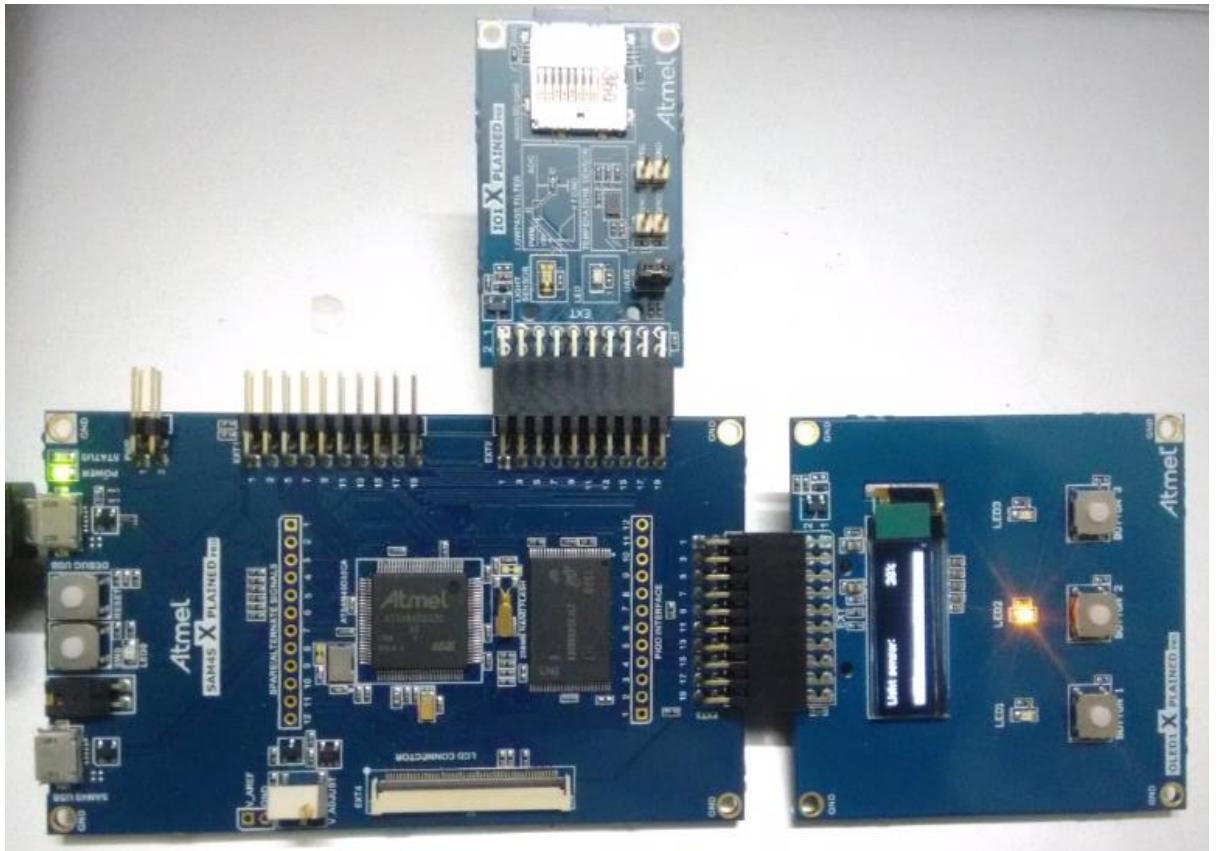


- Finally, “asf.h” has all related “.h” files and “main.c” for user coding.

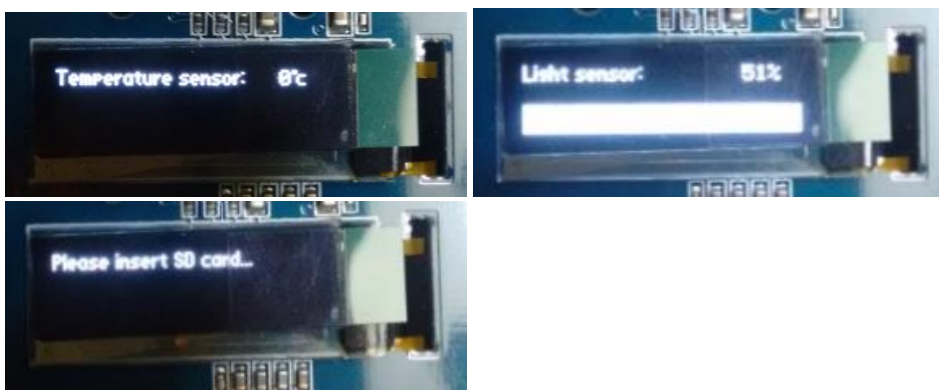


## 1.2 Board connection and Example project execution

1. Connect OLED board and sensor+SD card board with SAM4S xplained pro board as shown below.



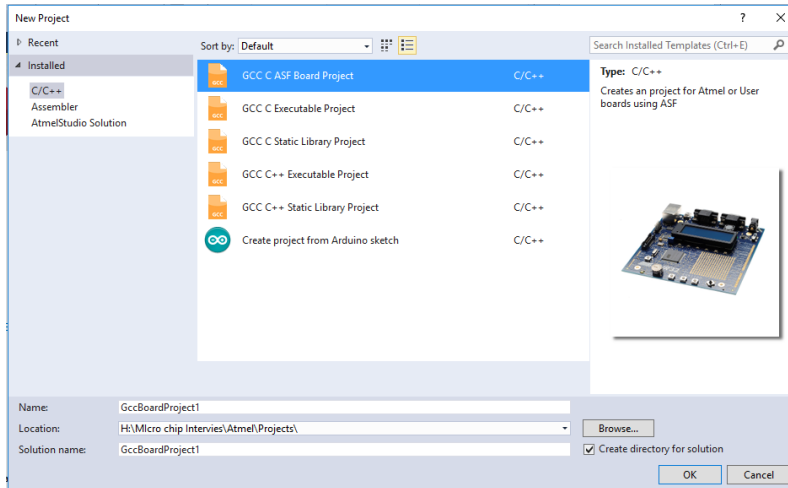
2. Now build and run the project on board. We can see the OLED Display changes based on Button1 press on OLED board as shown below.



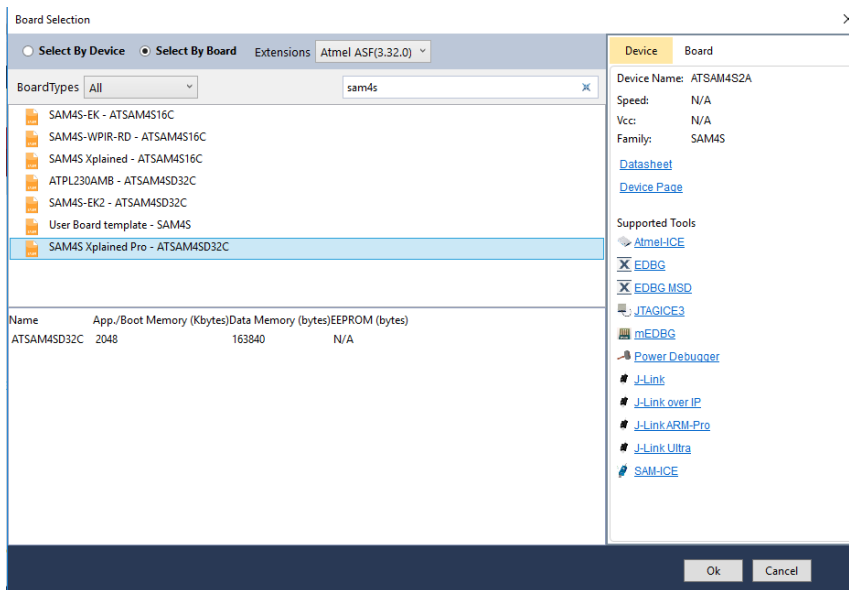
3. Now we are ready with working environment to develop our own applications in SAM4S Xplained pro using Atmel studio + ASF.

## 2. TASK :2 Creating OLED display application to display txt

1. Launch Atmel studio 7.0 and select “New Project”
2. Select “GCC C ASF Board Project”

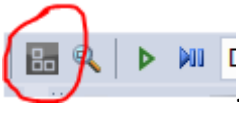


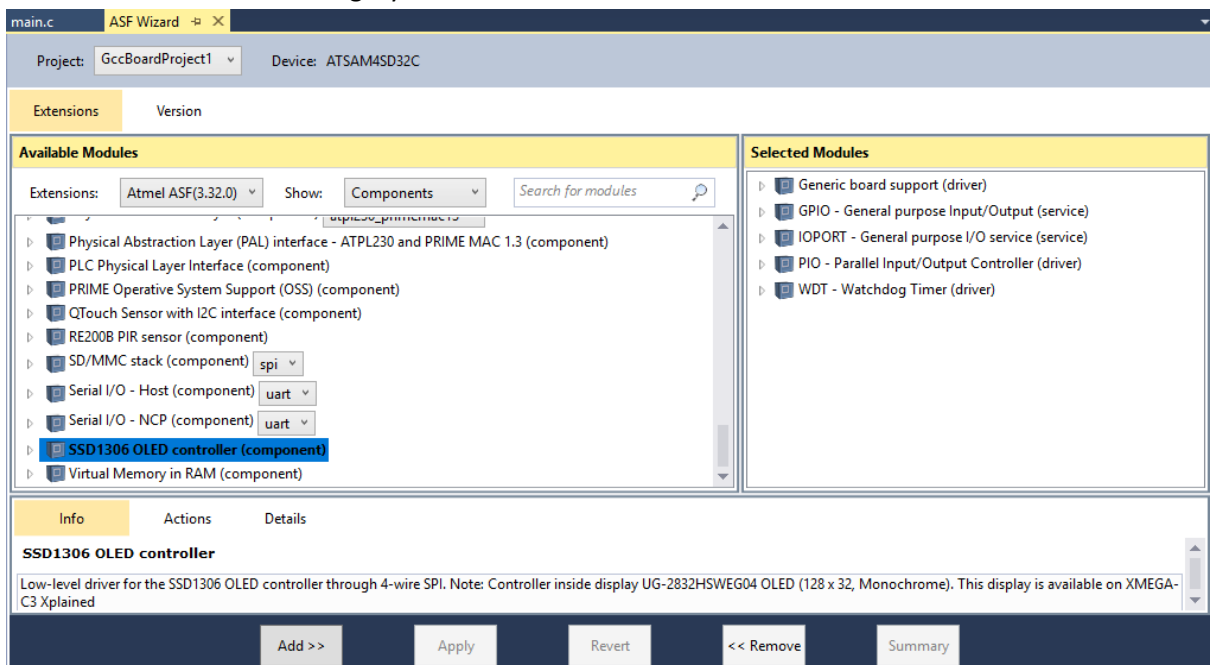
3. Select check box “Select By Board” and select “SAM4S Xplained Pro” from the list.



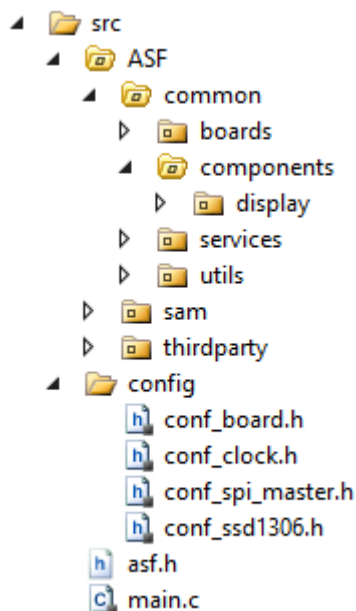
4. Now workspace is ready with simple Button press abed and Led blink application.



- Now select ASF Wizard from tool bar .
- On ASF wizard select "Components" under "Show" tab. Then select "SSD1306 OLED controller from list" and click "Add" button and click "Apply" button. And select "Delay routine " under services category.



- Now we can see related files added inside project folders.



- Add the OLED related macros inside "conf\_board.h" file.

```
// Enable the OLED screen & SD card
#define CONF_BOARD_SPI
#define CONF_BOARD_SPI_NPCS1
#define CONF_BOARD_SPI_NPCS2
#define CONF_BOARD_OLED_UG_2832HSWEG04
```



9. And edit "Conf\_ssd1306.h" with below configuration in place holder

```
#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
```

10. Now call initialization function of OLED display and write user application to print text.

```
#include <asf.h>

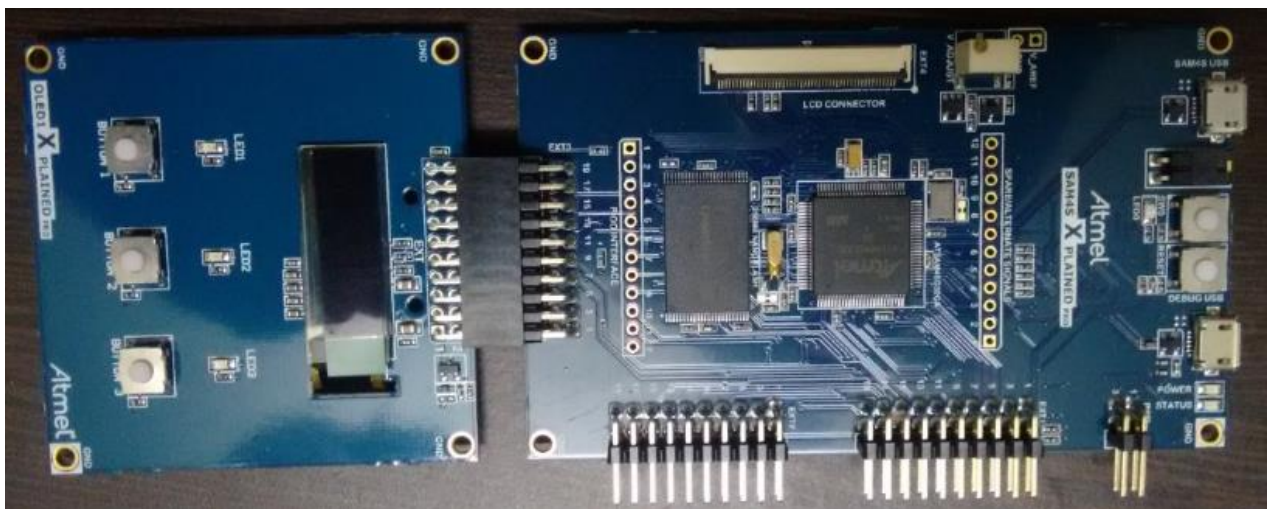
void Display_text(unsigned char * text,int page,int col)
{
    ssd1306_set_page_address(page);
    ssd1306_set_column_address(col);
    ssd1306_write_text(text);
}

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

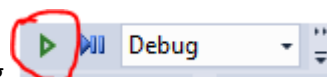
    // Initialize SPI and SSD1306 controller.
    ssd1306_init();
    ssd1306_clear();

    while (1)
    {
        ssd1306_clear();
        Display_text("Satheesh",0,0);
        Display_text("Microchip",1,0);
        delay_ms(50);
    }
}
```

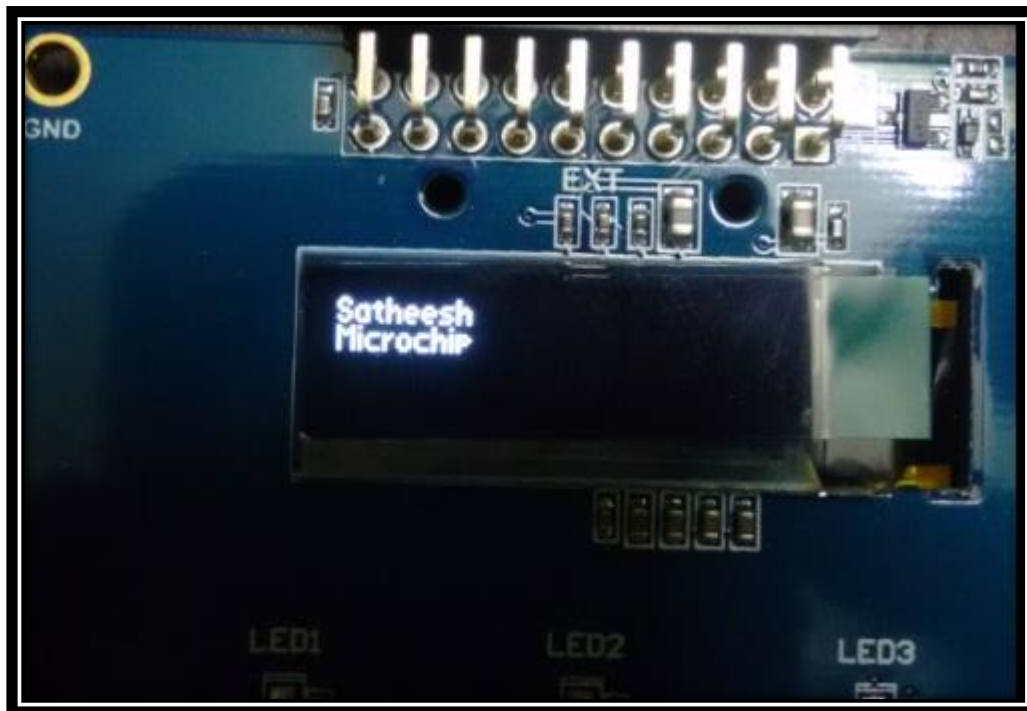
11. Now connect OLED display board with SAM4S Xplained Pro as shown below.




12. Now build and run the application on board by clicking



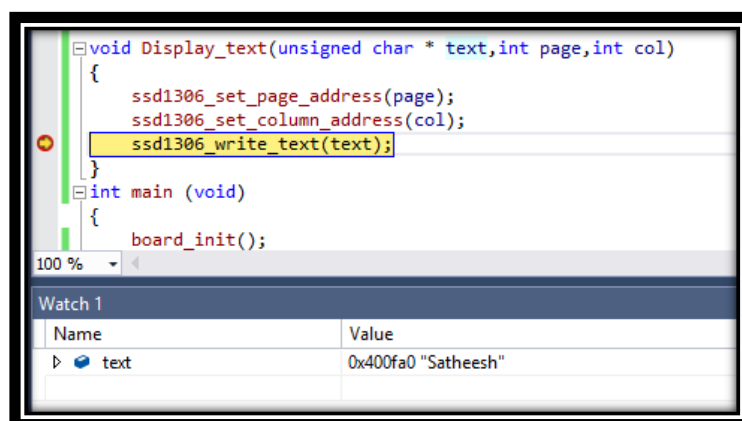
## 2.1 Result of TASK-2

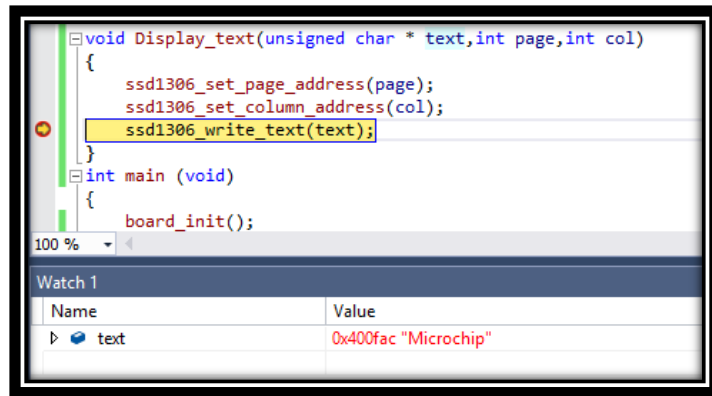


## 3. TASK :3 Creating OLED application and check text buffer in watch window (debug mode)

1. Put break point in display function call.
2. Run the application in Debug mode by clicking .
3. After hitting break point add the variable "text" to watch window.

### 3.1 Result of TASK-3



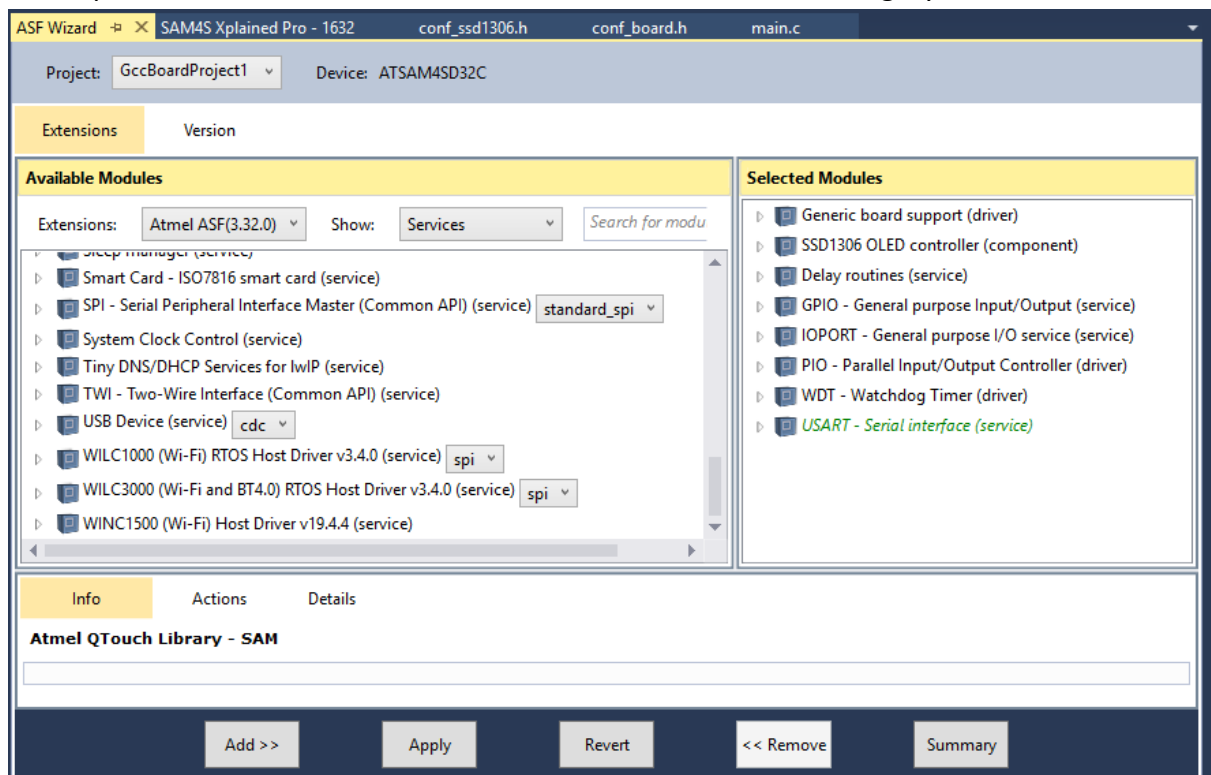


#### 4. TASK :4 installing serial terminal application

1. Installed "Docklight" serial terminal application.

#### 5. TASK :5 installing serial terminal application

1. Use the project created during Task-2 for OLED Display.
2. Now open ASF wizard and add "USART" service under services category.



3. Edit "conf\_board.h" file and add macro to enable UART module

```

//Enable UART
#define CONF_BOARD_UART_CONSOLE

```

4. Also, configure "Conf\_uart\_serial.h" with

```

/* A reference setting for USART */
/** 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

```

5. Now write code to initialize UART write a code to transmit data.

```

#include <asf.h>
#include <string.h>

void configure_usart(uint32_t baudrate);
void usart_stream_reset(void);
void usart_stream_write(uint8_t data);

void configure_usart(uint32_t baudrate)
{
    usart_serial_options_t uart_serial_options = {
        .baudrate = baudrate,
        .paritytype = CONF_UART_PARITY,
        .charlength = US_MR_CHRL_8_BIT,
        .stopbits = 0,
    };

    /* Configure the UART console. */
    usart_serial_init((usart_if)CONF_UART, &uart_serial_options);
    usart_stream_reset();
}

void usart_stream_reset(void)
{
    uart_reset((Uart *)CONF_UART);
    uart_enable((Uart *)CONF_UART);
}

void usart_stream_write(uint8_t data)
{
    usart_serial_putchar((usart_if)CONF_UART, data);

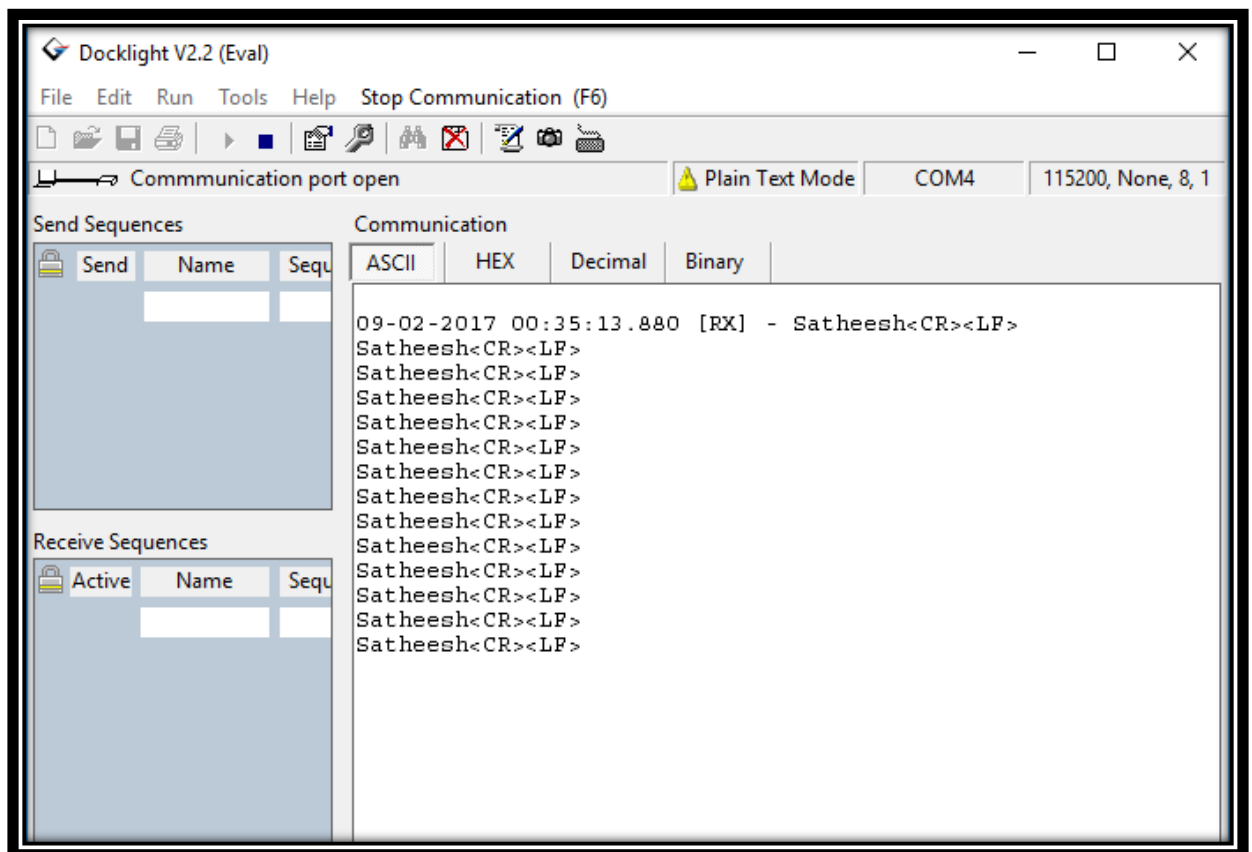
    while (!uart_is_tx_buf_empty((Uart *)CONF_UART)) {
    }
}

void Send_text_UART(unsigned char * text)
{
    int i;
    for (i = 0; i < strlen(text); i++){
        usart_stream_write(text[i]);
    }
}

void Display_text(unsigned char * text,int page,int col)
{
    ssd1306_set_page_address(page);
    ssd1306_set_column_address(col);
    ssd1306_write_text(text);
}

```

- Now run the application and select the proper COM port of SAM42 board in serial terminal application and make the settings with baud rate 115200.

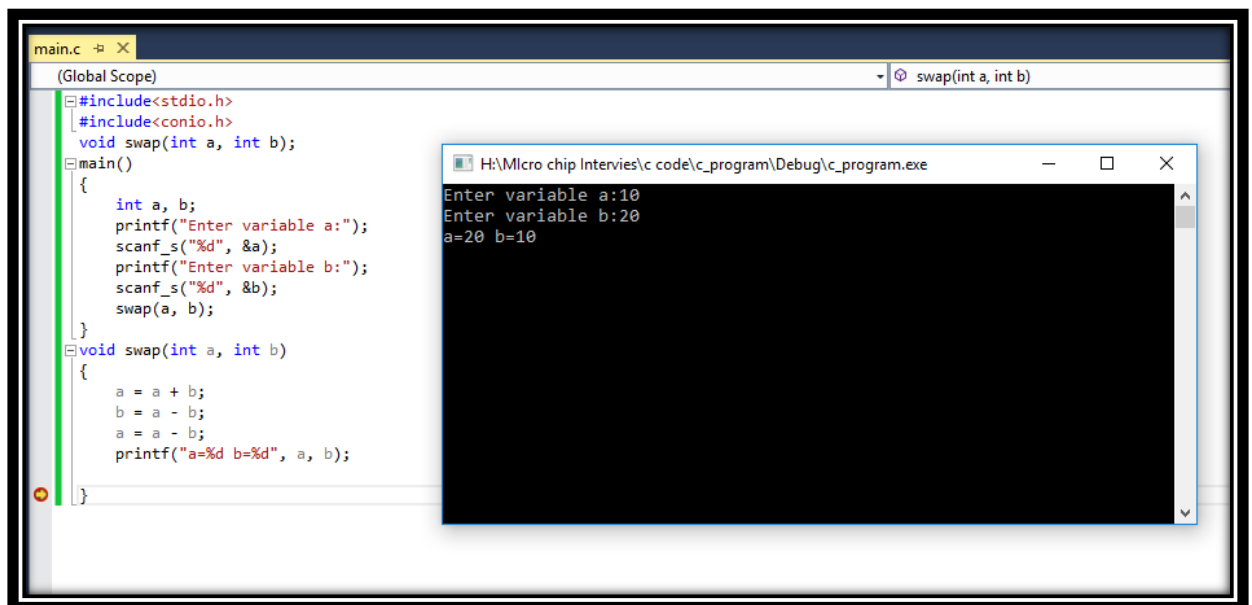


## 6. TASK :6 Code Exercises

### 6.1 Swap two variables without using intermediate variable

```
#include<stdio.h>
#include<conio.h>
void swap(int a, int b);
main()
{
    int a, b;
    printf("Enter variable a:");
    scanf_s("%d", &a);
    printf("Enter variable b:");
    scanf_s("%d", &b);
    swap(a, b);
}
void swap(int a, int b)
{
    a = a + b;
    b = a - b;
    a = a - b;
    printf("a=%d b=%d", a, b);
}
```

#### Result



### 6.2 Count number of bits set in a byte.

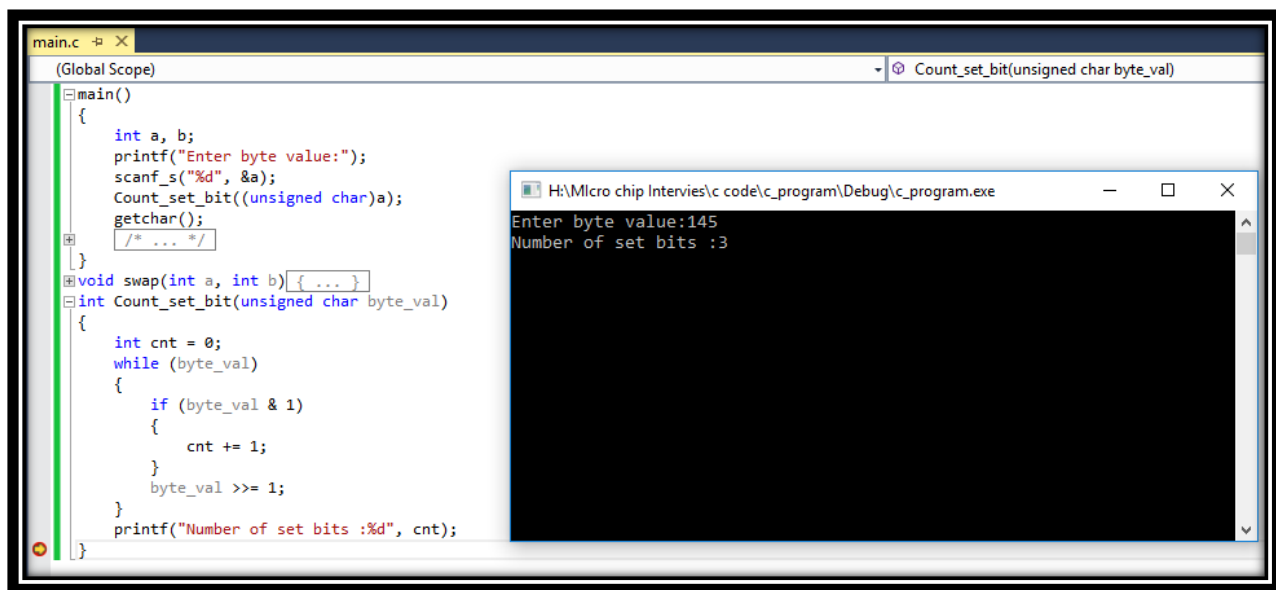
```
#include<stdio.h>
#include<conio.h>
void swap(int a, int b);
main()
{
    int a, b;
    printf("Enter byte value:");
```

```

scanf_s("%d", &a);
Count_set_bit((unsigned char)a);
}
int Count_set_bit(unsigned char byte_val)
{
    int cnt = 0;
    while (byte_val)
    {
        if (byte_val & 1)
        {
            cnt += 1;
        }
        byte_val >>= 1;
    }
    printf("Number of set bits :%d", cnt);
}

```

## Result



## 6.2 Check n-th bit of 32 bit value is set (1) or not

```

#include<stdio.h>
#include<conio.h>
main()
{
    int a, b, idx;
    long int val;

    printf("Enter 32bit Hex value:");
    scanf_s("%x", &val);

    printf("Enter index to be checked:");
    scanf_s("%d", &idx);

    if (Check_set_bit(val, idx))
    {
        printf("%d-th Bit of value is set(1)", idx);
    }
    else
    {

```

```

        printf("%d-th Bit of value is clear(0)", idx);
    }
}

int Check_set_bit(long int val, unsigned char index)
{
    int cnt = 0;
    val >>= index;
    if (val & 1){
        return 1;}
    else{
        return 0;}
}

```

## Result

The screenshot displays a C program in a code editor and its execution output in a separate window.

**Source Code (main.c):**

```

main()
{
    int a, b, idx;
    long int val;
    printf("Enter 32bit Hex value:");
    scanf_s("%x", &val);
    printf("Enter index to be checked:");
    scanf_s("%d", &idx);
    if (Check_set_bit(val, idx))
    {
        printf("%d-th Bit of value is set(1)", idx);
    }
    else
    {
        printf("%d-th Bit of value is clear(0)", idx);
    }
}

void swap(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}

int Count_set_bit(unsigned char byte_val)
{
    int cnt = 0;
    while (byte_val)
    {
        cnt++;
        byte_val = byte_val & (byte_val - 1);
    }
    return cnt;
}

int Check_set_bit(long int val, unsigned char index)
{
    int cnt = 0;
    val >>= index;
    if (val & 1){
        return 1;}
    else{
        return 0;}
}

```

**Execution Output (H:\Micro chip Intervies\c code\c\_program\Debug\c\_program.exe):**

```

Enter 32bit Hex value:0xAB142
Enter index to be checked:
6
6-th Bit of value is set(1)

```