

Lab 5: SPI

Quang Hai Nguyen

Revision 0.99, 07.01.2020

SAMD21 workshop

©2017 by ARROW

All rights reserved. No part of this manual shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, desktop publishing, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this document, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein. All terms mentioned in this manual that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks or service marks have been appropriately capitalized. ARROW cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Revision History

Revision, Date	Editor	Subject (major changes)
Revision 0.99, 07.01.2020	Quang Hai Nguyen	Preliminary

Table of Contents





Revision History	3
Table of Contents	4
List of Figures.....	5
List of Icon Identifiers.....	6
Prerequisite	7
Hardware	7
Software	7
Labs description	7
Description.....	7
Assignment.....	7
Lab procedure	8
Setup SPI bus with Atmel Start	8
Writing the code to read manufacture and device ID.....	11
Contact information	15

List of Figures

No table of figures entries found.

List of Icon Identifiers

Table 1: Icon Identifiers List

-  Extra information about a topic
-  Task needs to be done
-  Important information or a warning
-  Result expected to see

Prerequisite

Hardware

- Laptop or PC with Windows 7 or Windows 10
- SAMD21-Xplained-Pro evaluation board (provided during the workshop)
- Extension boards, LEDs, Potential meter, buttons and jumper wires (provided during the workshop)
- A micro USB cable (provided during the workshop)
- Breadboard (provided during the workshop)
- **Internet access without proxy or firewall**

Software

- Terminal program (e.g. TeraTerm)
- Atmel Studio 7 ([link](#))

Labs description

Description

This lab will show you how to configure the SPI bus by using Atmel Start and how to interface with Serial flash via SPI bus.

Assignment

1. Setup SPI bus with Atmel Start
2. Writing the code to read out Serial Flash's manufacturer and device ID



Some of the basic configuration steps are skipped in this lab so make sure you have already started lab 1 to learn the basic steps before continuing with this lab.

Lab procedure

Setup SPI bus with Atmel Start

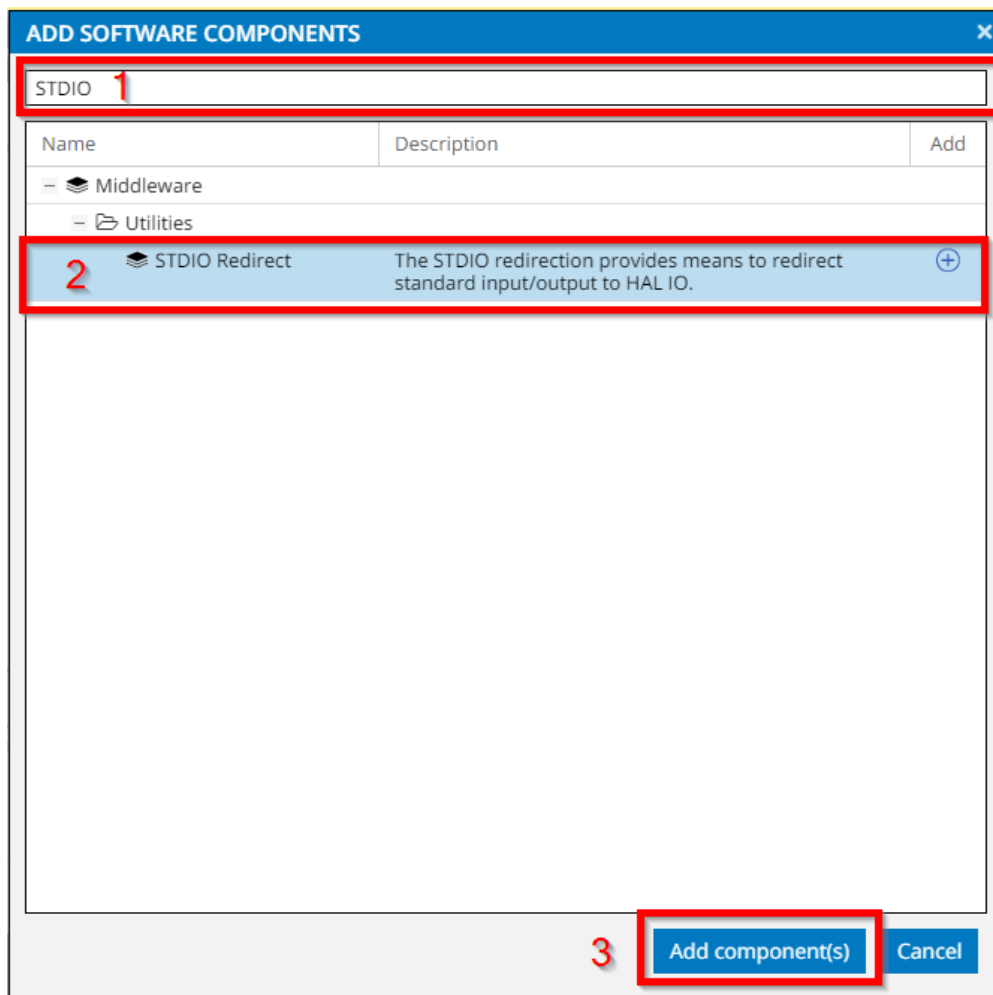
In Atmel Studio, create a new Atmel Start project.

When it is prompted, choose the correct device for your project. In this case, it is SAM D21 Xplained Pro.

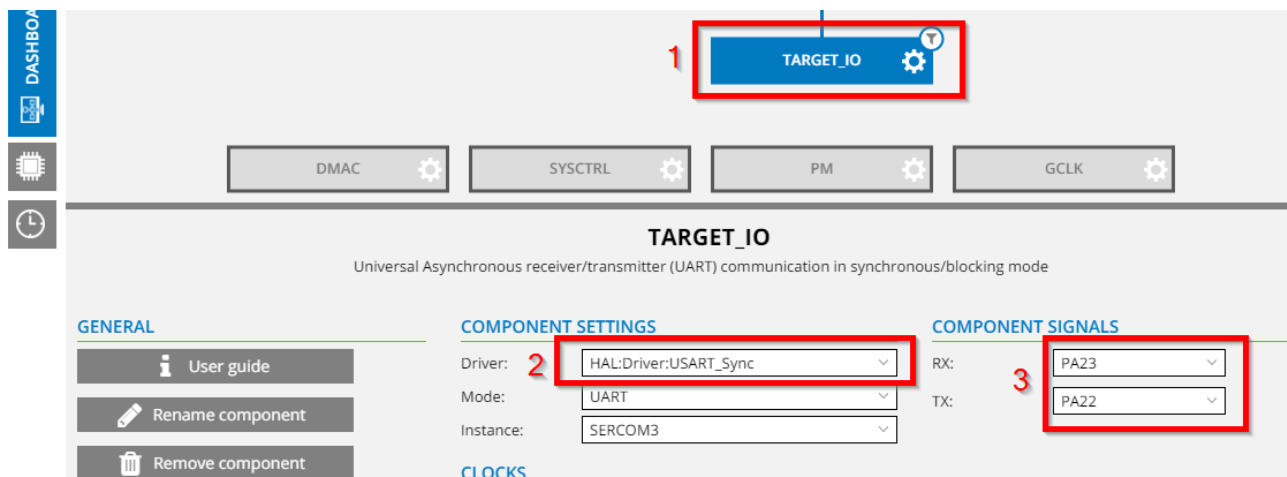
Give your project a name. For my case, I name it SAMD21_Lab5.

For this project, we need 2 components: SPI to interface with the serial flash and STDIO to send the debug messages to the terminal. We will start setting up the STDIO.

Click “Add software component”. In the prompted window, enter “STDIO” into the filter text box and select “STDIO Redirect”

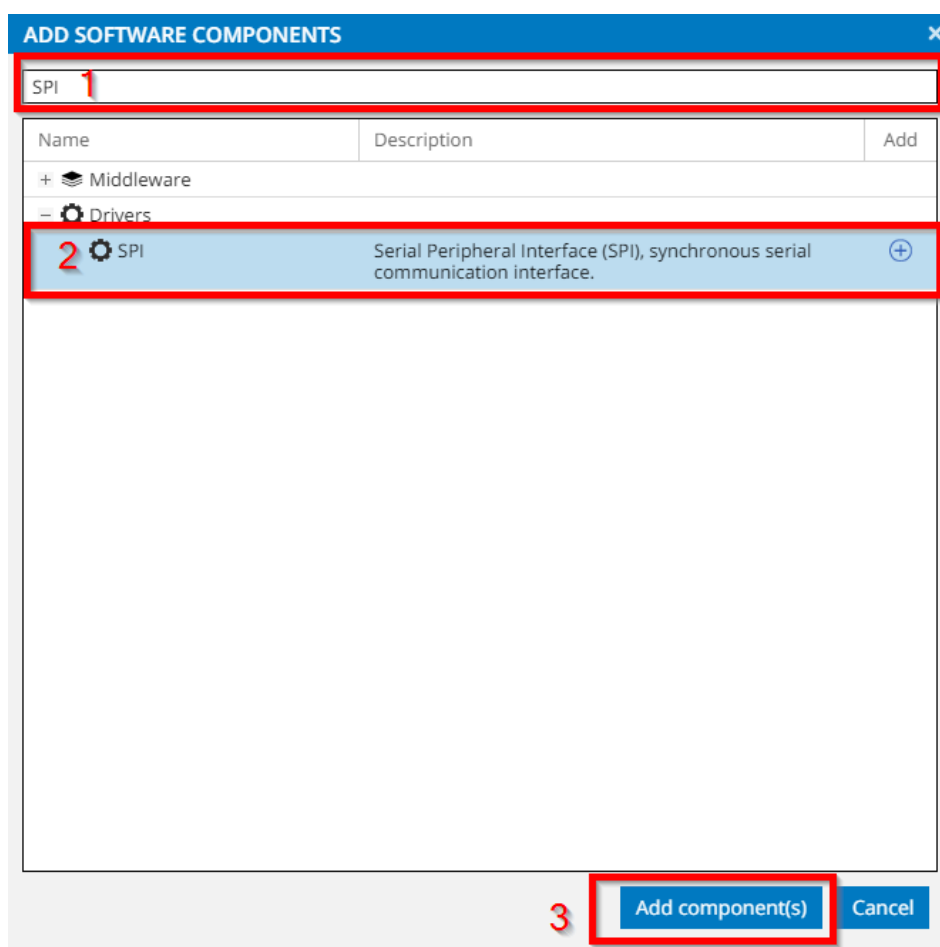


Click “TARGET_IO” and set the correct pin for the UART bus. From the “Board user guide”, we know that pin PA22 and PA23 are routed to output pins of USB Virtual Com. Therefore, they are configured as follows:



We will keep other configurations as they are.

Next step, we will set up our SPI bus. Click “Add software component”. In the prompted window, enter SPI into the filter text box and choose “SPI” under “Drivers category”.



According to the board user guide, MOSI, MISO, and SCK pins are located at pins PB22, PB16 and PB23.

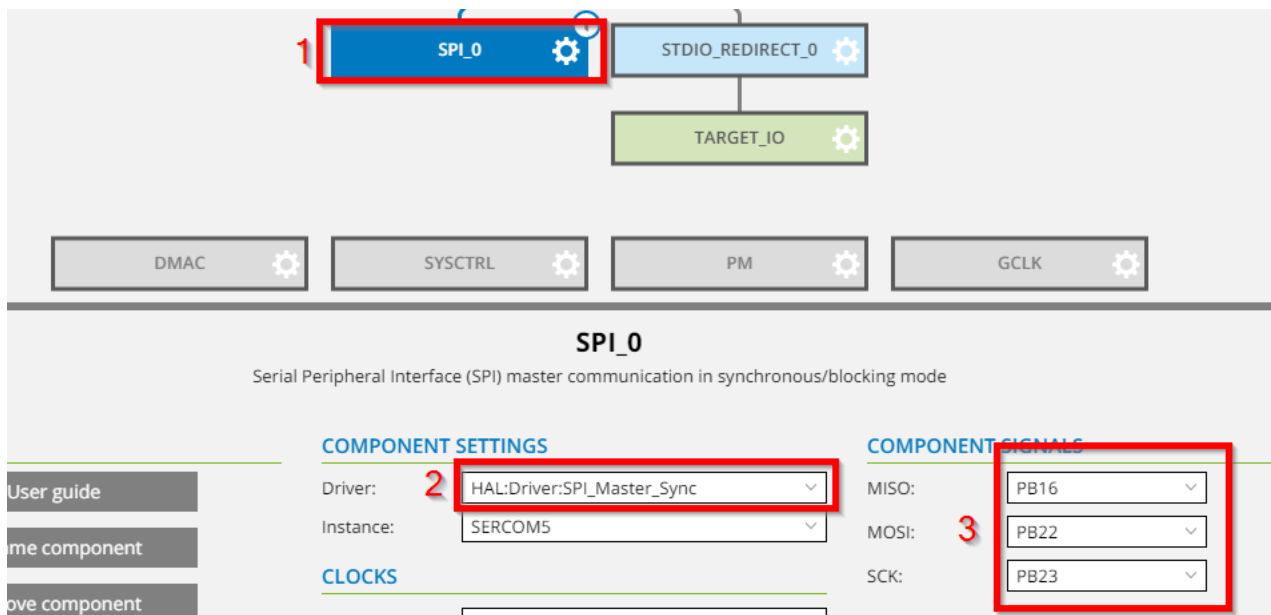
4.2.5. Serial Flash

The SAM D21 Xplained Pro has an onboard 8Mbit serial flash for non-volatile storage of data.

Table 4-8. Serial Flash Connections

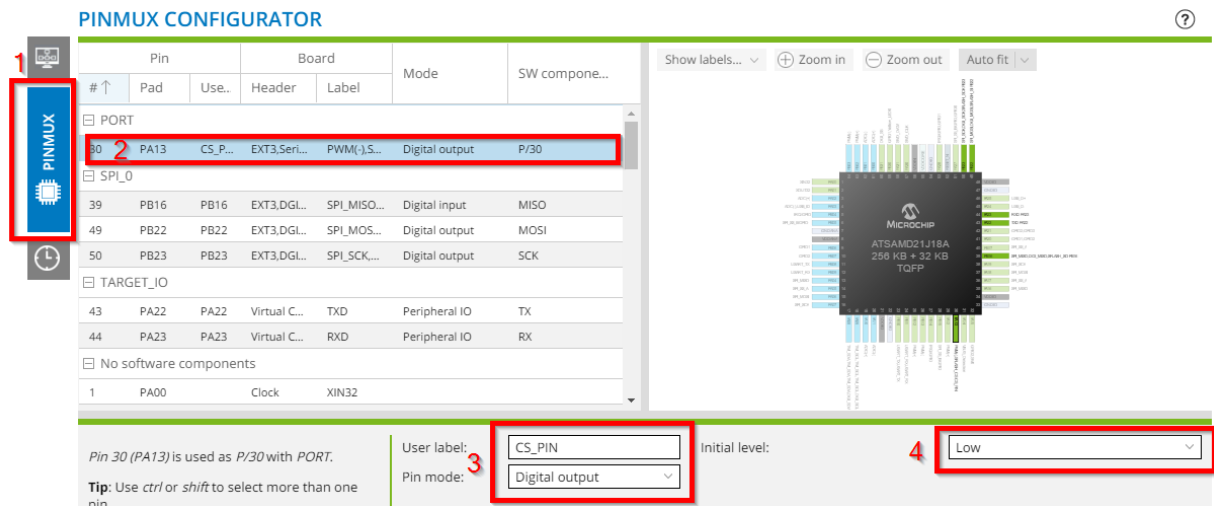
Pin on SAM D21	Function	Serial Flash	Shared functionality
PA13	GPIO	#CS	EXT3 PIN8 (default disconnected)
PB22	SERCOM5 PAD[2] SPI MOSI	SI	EXT3 and EDBG
PB16	SERCOM5 PAD[0] SPI MISO	SO	EXT3 and EDBG
PB23	SERCOM5 PAD[3] SPI SCK	SCK	EXT3 and EDBG

Therefore, we configure SPI as following:



For other configurations, we will keep them as they are.

We also must configure the CS pin, which is located at pin PA13. Therefore, jump to the “PINMUX” tab, look for pin PA13, give it a label and set it up as a digital output.



Generate the project.

Writing the code to read manufacturer and device ID

Compile the project to check if the generation process causes no errors.



If errors happen, please start Atmel Studio and re-generate the project again.

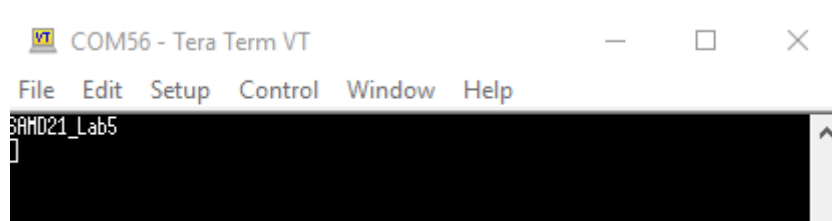
In the main.c file, we will add a hex_dump(args) function, which prints hex values to the terminal. In main.c, before the main() function, add the following code snippet:

```
void hex_dump(uint8_t * buff, uint32_t size)
{
    uint16_t i = 0;
    uint8_t line_count = 0;
    for(;i < size; i++) {
        printf("0x%02x, ",buff[i]);
        line_count++;
        if(line_count == 8) {
            printf("\r\n");
            line_count = 0;
        }
    }
    printf("\r\n");
}
```

In the main() function, below atmel_start_init() line, add the following line:

```
printf("SAMD21_Lab5")
```

Compile and run the application, you should get the result on the terminal console as following:



Next step, we will read the manufacture and device ID of the serial flash. To understand which APIs we must implement, let take a look at "hal/include/hal_spi_m_sync.h" file. In hal_spi_m_sync.h file, the two functions, which are interesting for us are: spi_m_sync_enable() and spi_m_sync_transfer().

spi_m_sync_enable will enable the selected SPI bus.

```
/** \brief Enable SPI
 *
 * \param[in] spi Pointer to the HAL SPI instance.
 *
 * \return Operation status.
 * \retval ERR_NONE Success.
 * \retval <0 Error code.
 */
void spi_m_sync_enable(struct spi_m_sync_descriptor *spi);
```

spi_m_sync_transfer will transfer or receive the message on the spi bus

```
/** \brief Perform the SPI data transfer (TX and RX) in polling way
 *
 * Activate CS, do TX and RX and deactivate CS. It blocks.
 *
 * \param[in, out] spi Pointer to the HAL SPI instance.
 * \param[in] xfer Pointer to the transfer information (\ref spi_xfer).
 *
 * \retval size Success.
 * \retval >=0 Timeout, with number of characters transferred.
 * \retval ERR_BUSY SPI is busy
 */
int32_t spi_m_sync_transfer(struct spi_m_sync_descriptor *spi, const struct spi_xfer *xfer);
```

The spi_xfer has the following format:

```

/** \brief Transfer descriptor for SPI
 * Transfer descriptor holds TX and RX buffers
 */
struct spi_xfer {
    /** Pointer to data buffer to TX */
    uint8_t *txbuf;
    /** Pointer to data buffer to RX */
    uint8_t *rxbuf;
    /** Size of data characters to TX & RX */
    uint32_t size;
};

```

Go back to main.c, in the main() function, add the following code to initialize buffer for transceiving data and enable SPI bus.

```

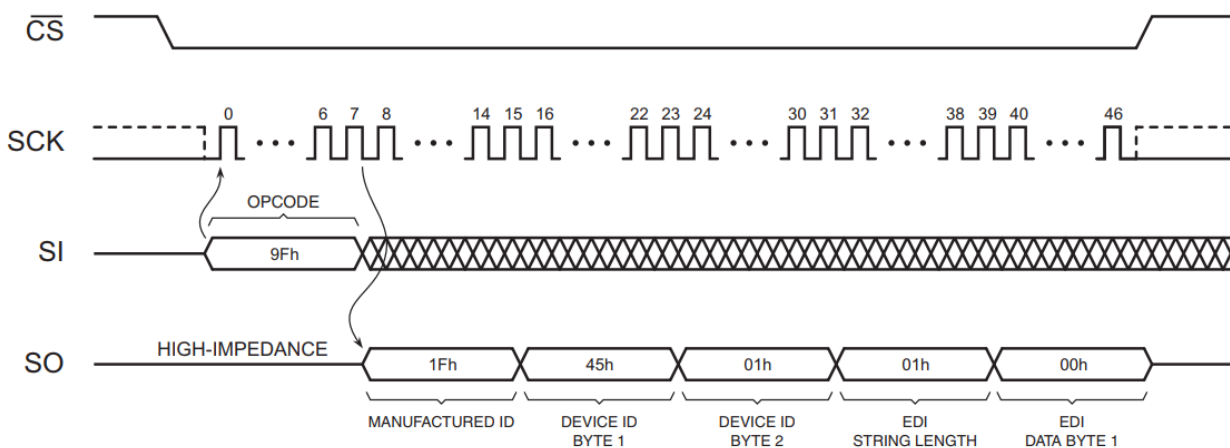
uint8_t spi_tx[5] = {0x00};
uint8_t spi_rx[5] = {0x00};
struct spi_xfer spi_packet;


spi_m_sync_enable(&SPI_0);

```

According to the datasheet of the Serial Flag, to read out the manufacture and device ID, we have to pull the CS pin low, send the command code “0x9F” on the SPI bus, read back 4 bytes from SPI bus and return the CS line to high.

Figure 12-2. Read Manufacturer and Device ID



Note: Each transition  shown for SI and SO represents one byte (8 bits)

In main() function, add the following snippet:

```

gpio_set_pin_level(CS_PIN, false);

//TX packet
spi_packet.txbuf = spi_tx;
spi_packet.rxbuf = 0;
spi_packet.size = 1;
spi_tx[0] = 0x9F;
spi_m_sync_transfer(&SPI_0, &spi_packet);

//RX packet
spi_packet.txbuf = 0;
spi_packet.rxbuf = spi_rx;
spi_packet.size = 4;
spi_m_sync_transfer(&SPI_0, &spi_packet);

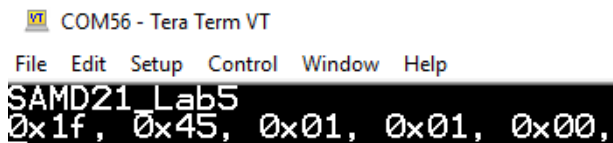
gpio_set_pin_level(CS_PIN, true);

hex_dump(spi_rx, 10);

while(1){
}

```

Compile and Run the application. This is the result on the terminal:



```

COM56 - Tera Term VT
File Edit Setup Control Window Help
SAMD21_Lab5
0x1f, 0x45, 0x01, 0x01, 0x00,

```

Congratulation, you have finished lab 5

Contact information

Quang Hai Nguyen, B. Eng.
Field Application Engineer

P +49 6102 50308228
M +49 1511 6242003
quanghai.nguyen@arrow.com

Arrow Central Europe GmbH
Frankfurter Straße, 211
63263 Neu-Isenburg

THE END