External Flash (SPI)

External Flash (SPI)
I. Learning Objectives
II. SPI Introduction
III. Hardware Setup
IV. Environment Setup
V. Main Functions
VI. Experiment Phenomenon

I. Learning Objectives

1. Learn and understand basic SPI communication knowledge.
2. Learn to drive W25Q64 through SPI.

II. SPI Introduction

SPI protocol (Serial Peripheral Interface) is a commonly used synchronous serial communication protocol, specially designed for high-speed, short-distance communication between microcontrollers and external devices. It supports full-duplex communication, meaning data can be sent and received simultaneously between master and slave devices, commonly used for memory chips, sensors, display drivers, and wireless modules.

SPI Hardware Interface

SPI protocol typically uses four lines for data transmission:

- **SCLK (Serial Clock):** Clock signal generated by master device, used for synchronizing communication.
- **MOSI/PICO (Master Output Slave Input):** Master device sends data to slave device.
- **MISO/POCI (Master Input Slave Output):** Slave device sends data to master device.
- **SS/CS (Slave Select):** Master device activates target slave device through chip select signal.

SPI Advantages

- High communication speed, suitable for high-bandwidth applications.
- Supports full-duplex transmission, improving efficiency.
- Simple hardware implementation, direct interface.
- Supports multiple slave device connections.

SPI Disadvantages

- Lacks standardized error detection mechanism.
- Master device pin count increases with number of slave devices.
- Communication distance is limited, typically used for onboard communication.

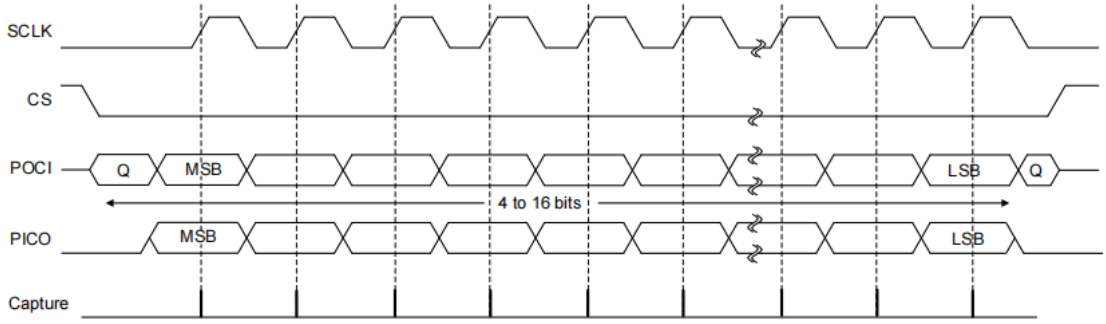
SPI Transmission Modes

Clock Polarity (CPOL): Controls the idle and active states of the clock signal (0: idle state low level, 1: idle state high level)

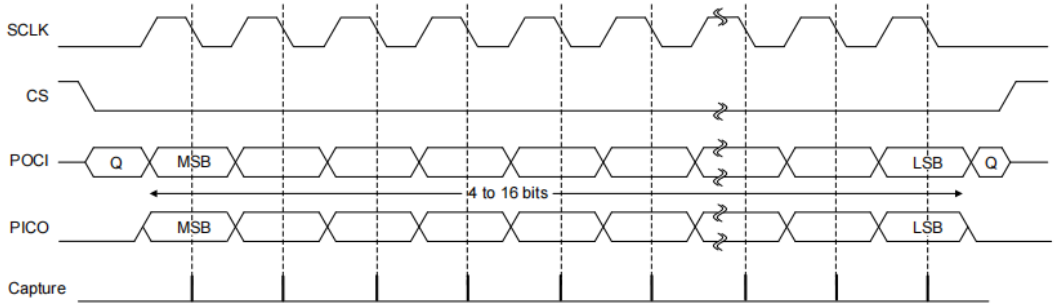
Clock Phase (CPHA): Controls data sampling and transmission timing (0: sample on odd edge, 1: sample on even edge)

Clock Polarity (CPOL)	Clock Phase (CPHA)	
0	0	SCL idle state low level; data sampled on rising edge, transmitted on falling edge
0	1	SCL idle state low level; data sampled on falling edge, transmitted on rising edge
1	0	SCL idle state high level; data sampled on falling edge, transmitted on rising edge
1	1	SCL idle state high level; data sampled on rising edge, transmitted on falling edge

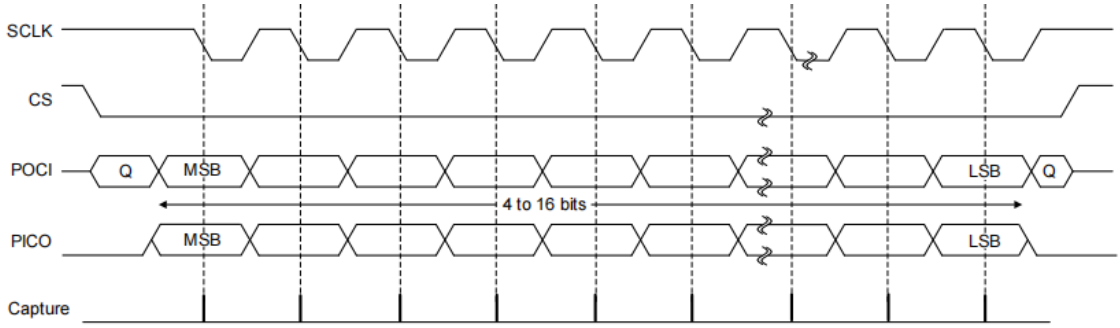
CPOL:0 CPHA:0



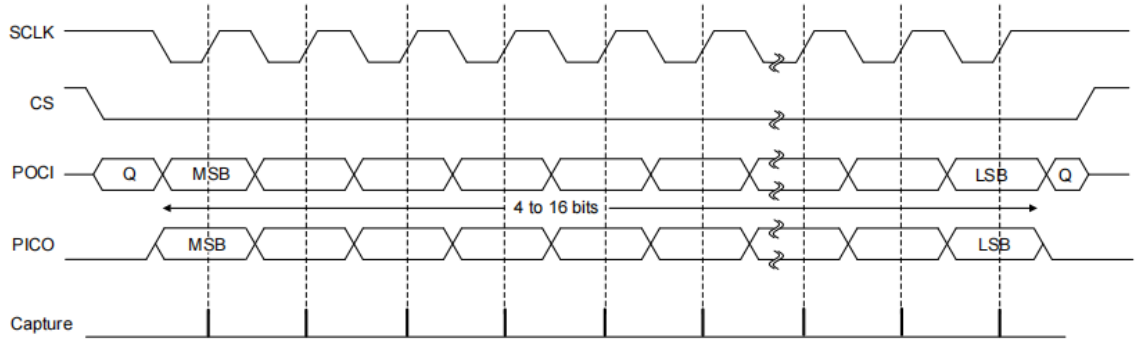
CPOL:0 CPHA:1



CPOL:1 CPHA:0



CPOL:1 CPHA:1



III. Hardware Setup

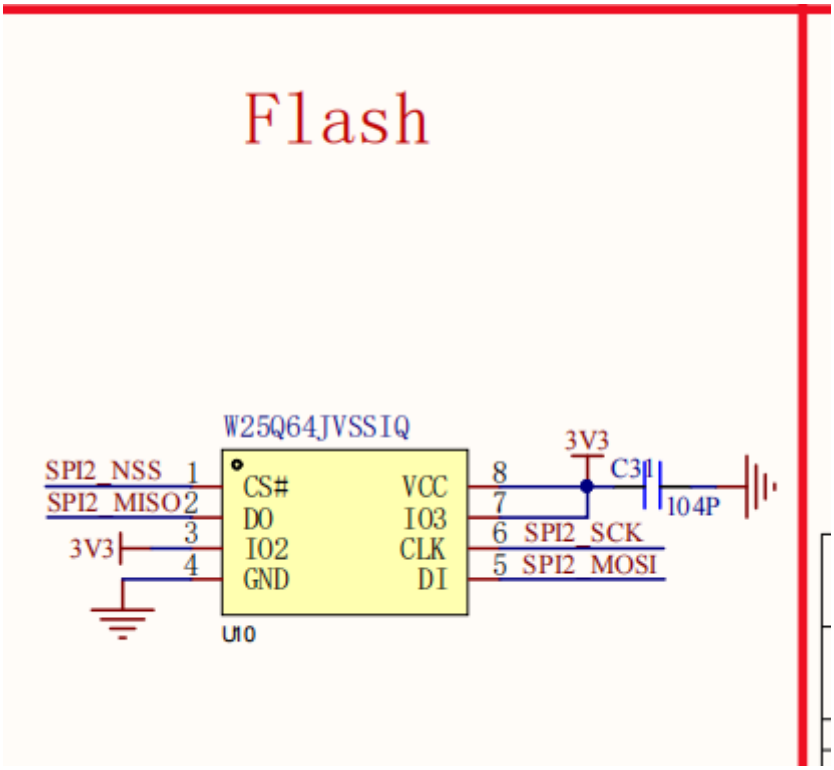
W25Q64 is a common serial flash memory device that uses SPI (Serial Peripheral Interface) protocol. It has high-speed read, write, and erase functions, widely used in embedded systems, storage devices, routers, and other high-performance electronic devices. W25Q32 has a capacity of 64 Mbit (i.e., 8 MB). The numbers in the model represent different capacity options, such as W25Q16, W25Q32, W25Q128, etc., to meet the needs of different application scenarios.

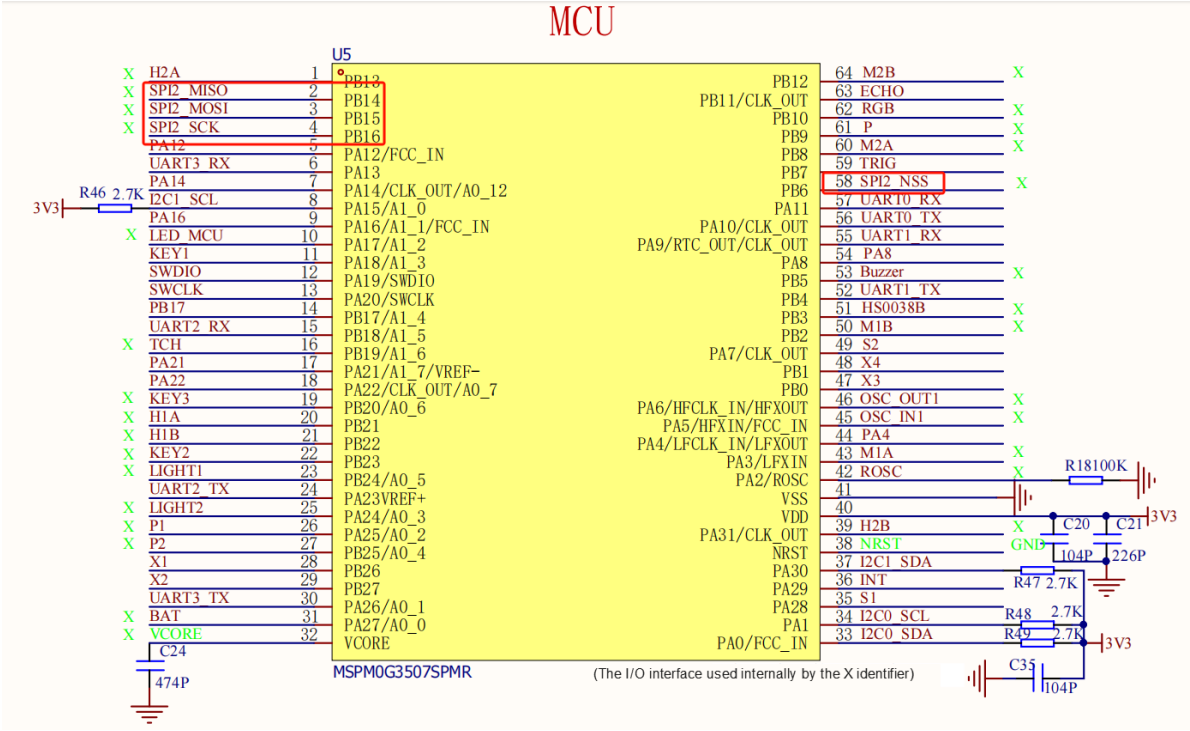
W25Q64 divides 8M bytes of capacity into 128 blocks, each block size is 64K bytes, each block is divided into 16 sectors, each sector is 4K bytes. The minimum erase unit of W25Q64 is one sector, which means 4K bytes must be erased each time.
Partial instruction table content is as follows

Data Input Output	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Number of Clock(1-1-1)	**8**	**8**	**8**	**8**	**8**	**8**	**8**
Write Enable	**06h**						
Volatile SR Write Enable	**50h**						
Write Disable	**04h**						
Release Power-down / ID	**ABh**	Dummy	Dummy	Dummy	(ID7-ID0)(2)		
Manufacturer/Device ID	**90h**	Dummy	Dummy	00h	(MF7-MF0)	(ID7-ID0)	
JEDEC ID	**9Fh**	(MF7-MF0)	(ID15-ID8)	(ID7-ID0)			
Read Unique ID	**4Bh**	Dummy	Dummy	Dummy	Dummy	(UID63-0)	
Read Data	**03h**	A23-A16	A15-A8	A7-A0	(D7-D0)		
Fast Read	**0Bh**	A23-A16	A15-A8	A7-A0	Dummy	(D7-D0)	
Page Program	**02h**	A23-A16	A15-A8	A7-A0	D7-D0	D7-D0(3)	
Sector Erase (4KB)	**20h**	A23-A16	A15-A8	A7-A0			
Block Erase (32KB)	**52h**	A23-A16	A15-A8	A7-A0			
Block Erase (64KB)	**D8h**	A23-A16	A15-A8	A7-A0			
Chip Erase	**C7h/60h**						
Read Status Register-1	**05h**	(S7-S0)(2)					
Write Status Register-1(4)	**01h**	(S7-S0)(4)					
Read Status Register-2	**35h**	(S15-S8)(2)					
Write Status Register-2	**31h**	(S15-S8)					
Read Status Register-3	**15h**	(S23-S16)(2)					
Write Status Register-3	**11h**	(S23-S16)					

This course requires no additional hardware equipment, directly using the onboard LCD screen and external storage flash on the MSPM0G3507 development board.

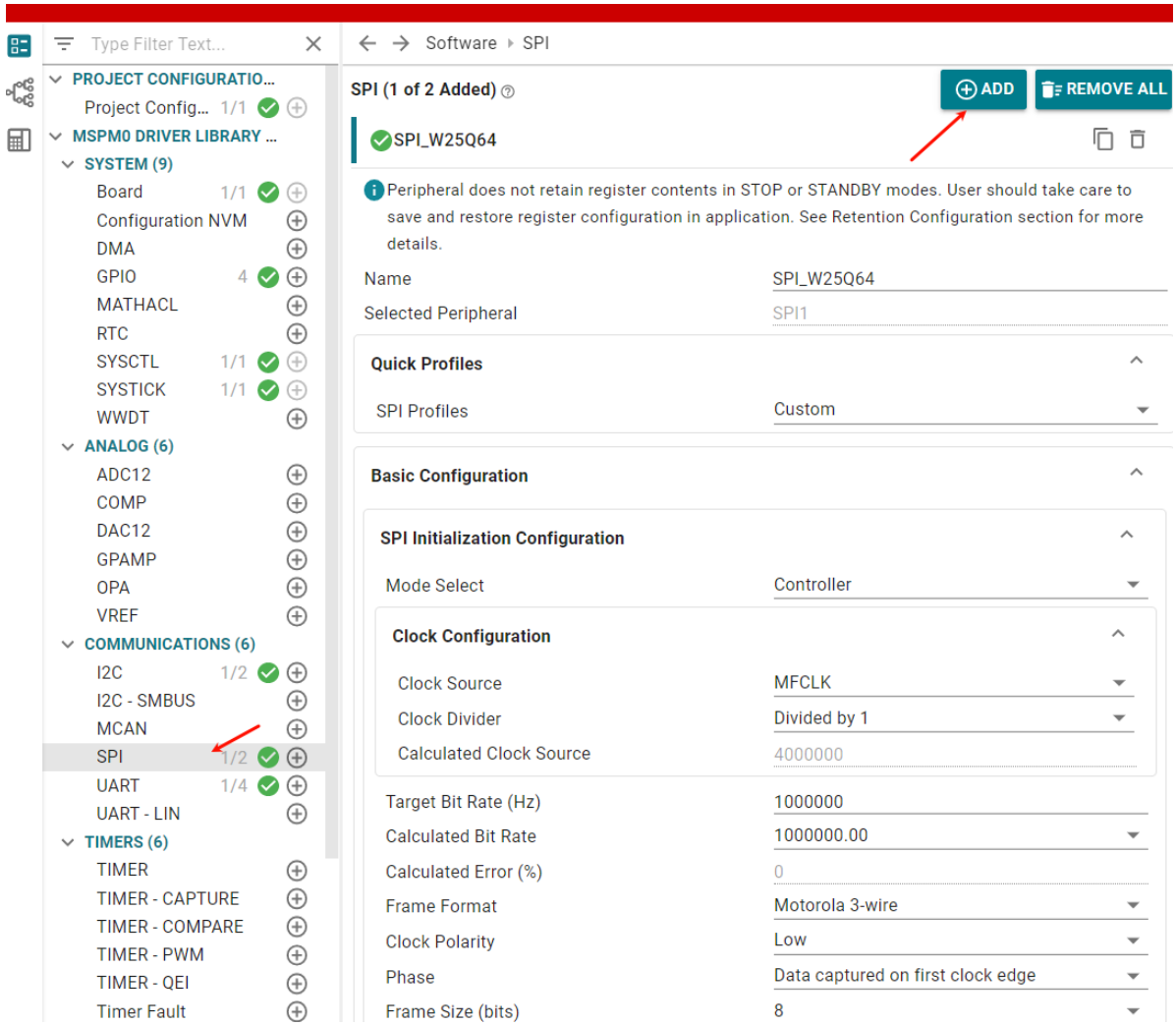
Connection relationship is as follows

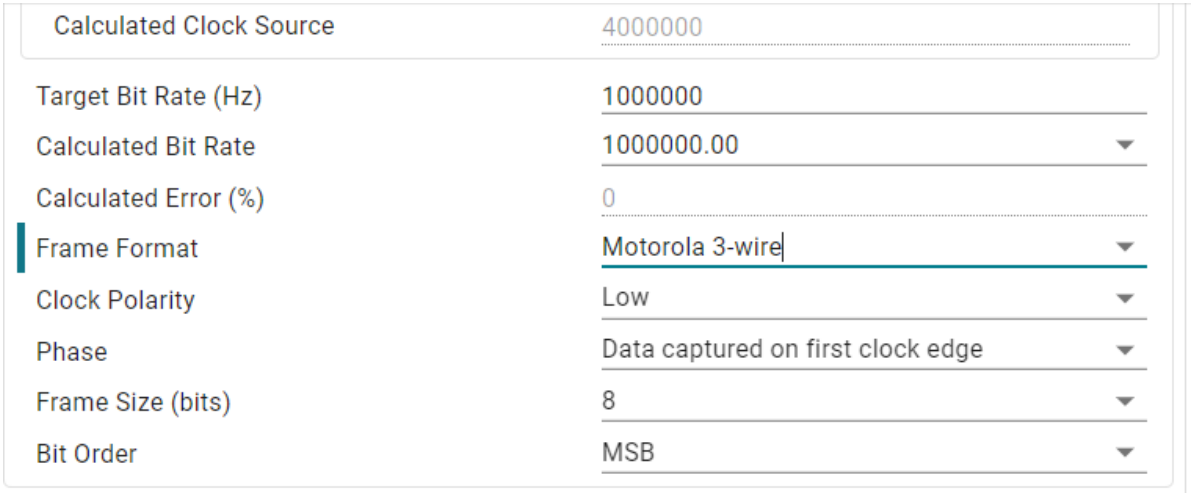


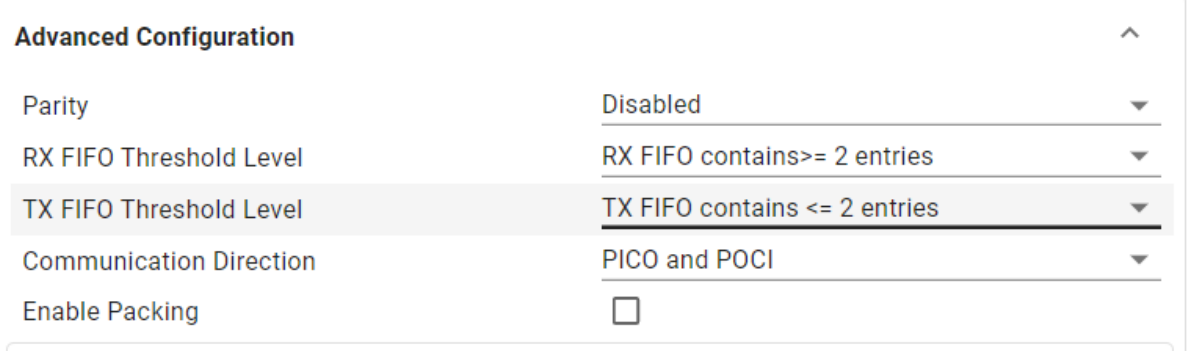


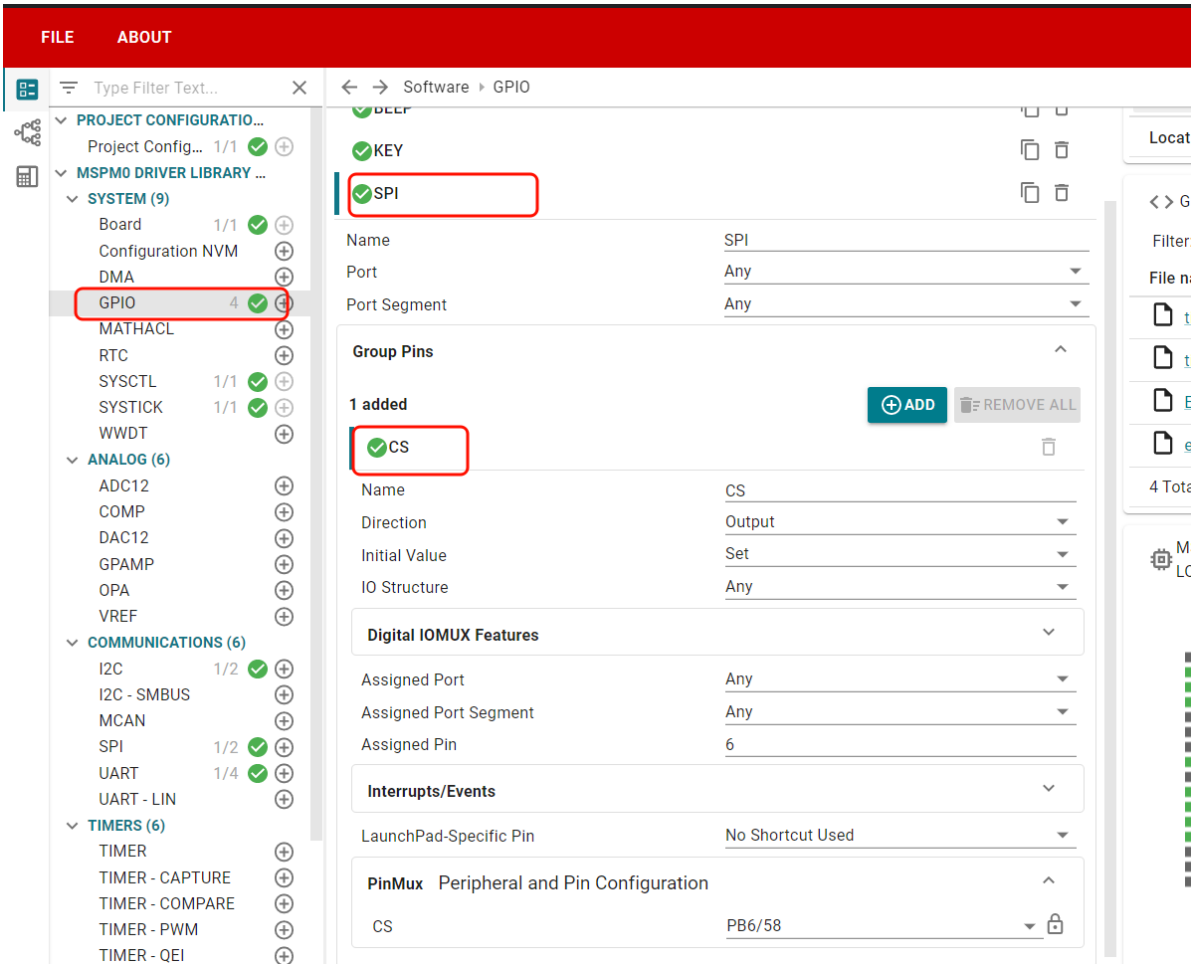
IV. Environment Setup

First, we add configuration through sysconfig as follows

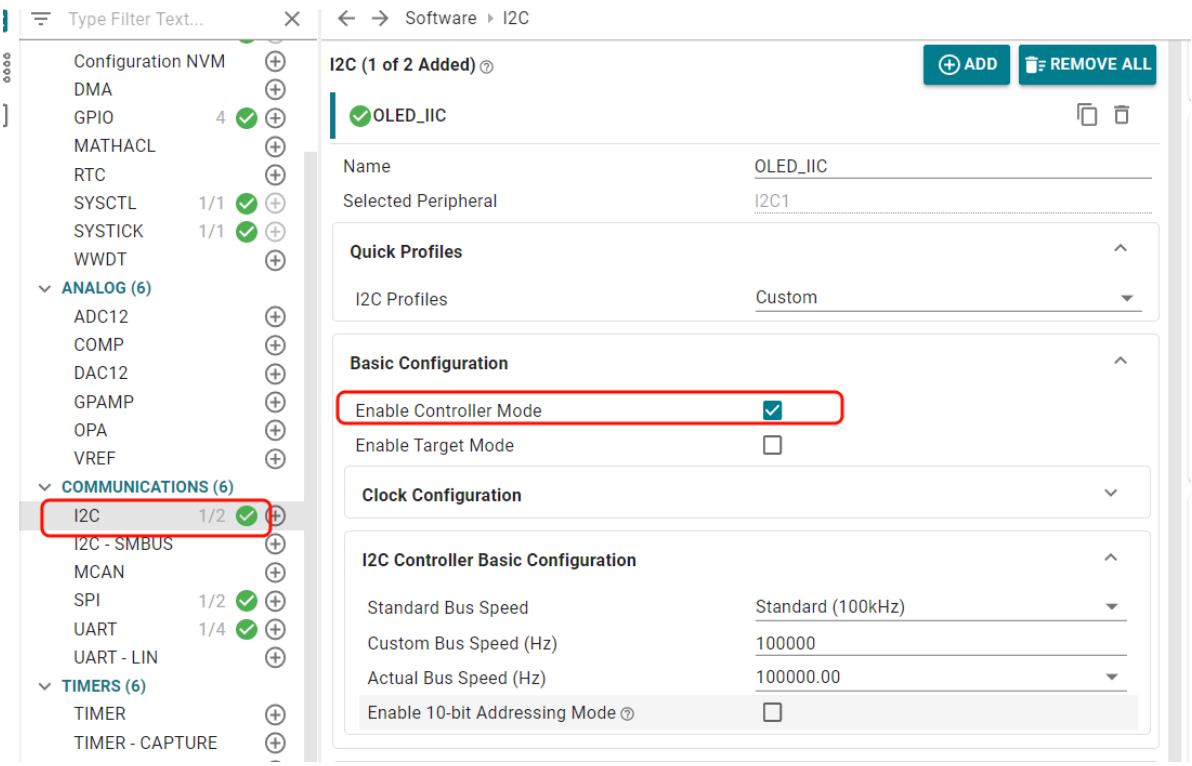

Type Filter Text...
PROJECT CONFIGURATIO...
Project Config... 1/1
MSPM0 DRIVER LIBRARY ...
SYSTEM (9)
Board 1/1
Configuration NVM
DMA
GPIO 4
MATHACL
RTC
SYSCTL 1/1
SYSTICK 1/1
WWDT
ANALOG (6)
ADC12
COMP
DAC12
GPAMP
OPA
VREF
COMMUNICATIONS (6)
I2C 1/2
I2C - SMBUS
MCAN
SPI 1/2
UART 1/4
UART - LIN
TIMERS (6)
TIMER
TIMER - CAPTURE
TIMER - COMPARE
TIMER - PWM
TIMER - QEI
Timer Fault
Software ▸ SPI
SPI (1 of 2 Added)
ADD
REMOVE ALL
SPI_W25Q64
Peripheral does not retain register contents in STOP or STANDBY modes. User should take care to save and restore register configuration in application. See Retention Configuration section for more details.
Name
SPI_W25Q64
Selected Peripheral
SPI1
Quick Profiles
SPI Profiles
Custom
Basic Configuration
SPI Initialization Configuration
Mode Select
Controller
Clock Configuration
Clock Source
MFCLK
Clock Divider
Divided by 1
Calculated Clock Source
4000000
Target Bit Rate (Hz)
1000000
Calculated Bit Rate
1000000.00
Calculated Error (%)
0
Frame Format
Motorola 3-wire
Clock Polarity
Low
Phase
Data captured on first clock edge
Frame Size (bits)
8

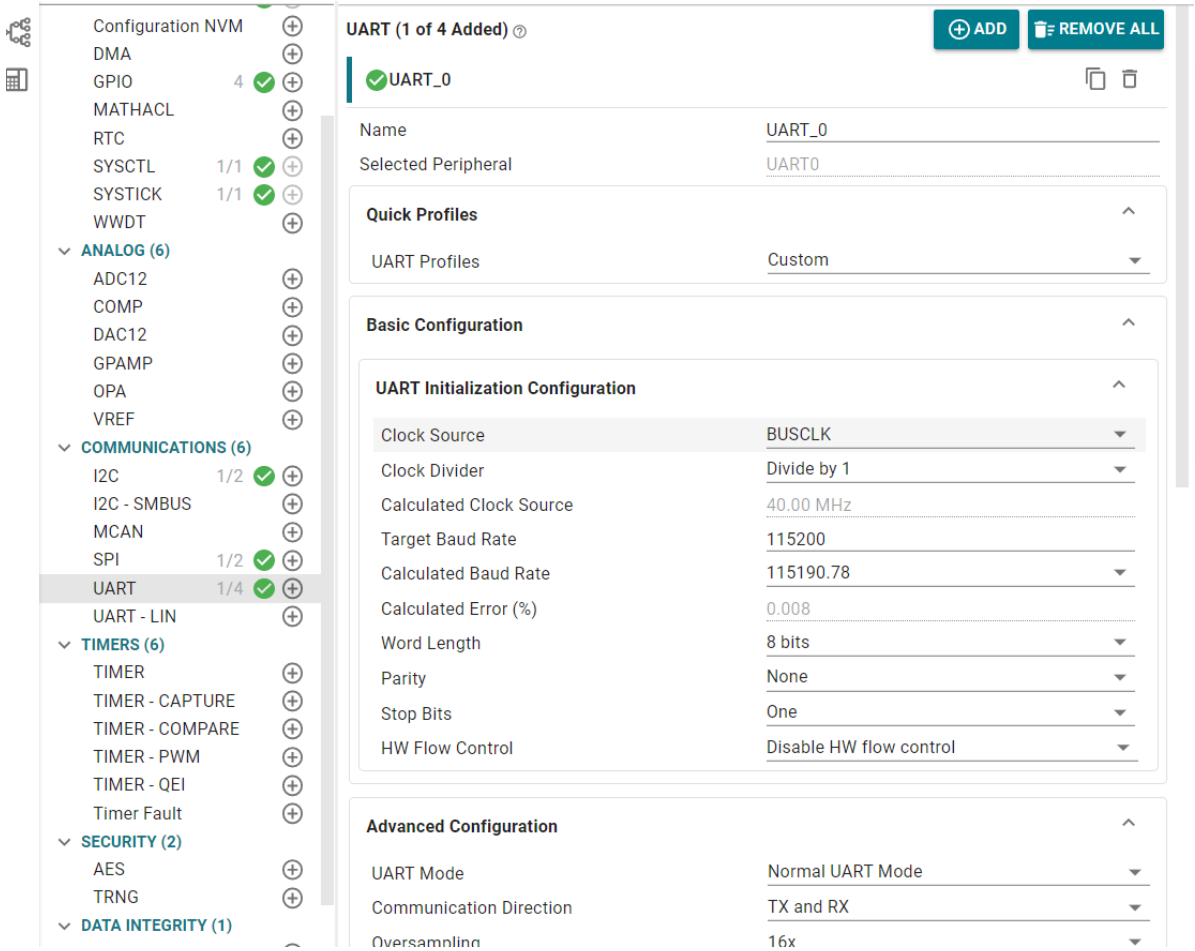

Calculated Clock Source
4000000
Target Bit Rate (Hz)
1000000
Calculated Bit Rate
1000000.00
Calculated Error (%)
0
Frame Format
Motorola 3-wire
Clock Polarity
Low
Phase
Data captured on first clock edge
Frame Size (bits)
8
Bit Order
MSB


Advanced Configuration
Parity
Disabled
RX FIFO Threshold Level
RX FIFO contains>= 2 entries
TX FIFO Threshold Level
TX FIFO contains <= 2 entries
Communication Direction
PICO and POCI
Enable Packing



Unlike the previous section, this section uses hardware I2C to drive the OLED screen





V. Main Functions

Mainly introduces the user-written functional code, **for detailed code, you can open the project files we provide and enter the Bsp folder to view the source code.**

W25Qxx_Init

Function Prototype	**void W25Qxx_Init(void)**
Function Description	Initialize W25Q64 Flash, main operation is to pull SPI_CS pin high and delay 300ms to ensure stable initialization
Input Parameters	None
Return Value	None

spi_read_write_byte

Function Prototype	**uint8_t spi_read_write_byte(uint8_t dat)**
Function Description	Complete SPI single byte transmit/receive: first send input dat byte, wait for SPI bus idle, then receive byte returned from slave device
Input Parameters	dat (uint8_t type, single byte data to be sent through SPI)
Return Value	uint8_t type, byte returned from slave device received by SPI bus

W25QXX_Self_Test

Function Prototype	**void W25QXX_Self_Test(void)**
Function Description	Flash self-test: loop read Flash ID and identify model (W25Q32/W25Q64/W25Q128), display model through OLED and print via serial port; simultaneously test Flash read/write function (write "Yahboom" and verify), print read/write results
Input Parameters	None
Return Value	None

W25Qxx_readID

Function Prototype	**uint16_t W25Qxx_readID(void)**
Function Description	Read W25Q64 device ID: send 0x90 instruction and 0x000000 address, receive 2-byte ID and concatenate to return
Input Parameters	None
Return Value	uint16_t type, Flash device ID (e.g., W25Q64 corresponding ID is 0xEF16)

W25Qxx_write_enable

Function Prototype	**void W25Qxx_write_enable(void)**
Function Description	Send "write enable" instruction (0x06) to Flash, allowing subsequent write/erase operations
Input Parameters	None
Return Value	None

W25Qxx_wait_busy

Function Prototype	**void W25Qxx_wait_busy(void)**
Function Description	Wait for Flash operation to complete: loop read status register, detect BUSY bit (lowest bit), until BUSY bit is 0 (indicating Flash is idle)
Input Parameters	None
Return Value	None

W25Qxx_erase_sector

Function Prototype	**void W25Qxx_erase_sector(uint32_t addr)**
Function Description	Erase specified sector: first convert input address to sector address (1 sector = 4096 bytes), send sector erase instruction (0x20) and address, wait for erase completion
Input Parameters	addr (uint32_t type, sector number, note: actual erase address = addr×4096)
Return Value	None

W25Qxx_write

Function Prototype	**void W25Qxx_write(uint8_t* buffer, uint32_t addr, uint16_t numbyte)**
Function Description	Write data to specified Flash address: first erase sector containing target address, send write enable, then send page write instruction (0x02) and address, finally continuously write numbyte bytes of data
Input Parameters	buffer (uint8_t * type, data source buffer to write) addr (uint32_t type, Flash target write address (24-bit)) numbyte (uint16_t type, total bytes to write)
Return Value	None

W25Qxx_read

Function Prototype	**void W25Qxx_read(uint8_t* buffer, uint32_t read_addr, uint16_t read_length)**
Function Description	Read data from specified Flash address: send read instruction (0x03) and target address, continuously read read_length bytes of data and store in buffer
Input Parameters	buffer (uint8_t * type, target buffer to store read data) read_addr (uint32_t type, Flash target read address (24-bit)) read_length (uint16_t type, total bytes to read)
Return Value	None

OLED_Init

Function Prototype	**void OLED_Init(void)**
Function Description	OLED initialization, execute SSD1306 initialization commands, clear screen and update display, set cursor default values, finally display "OLED init Success!"
Input Parameters	None
Return Value	None

SSD1306_UpdateScreen

Function Prototype	**void SSD1306_UpdateScreen(void)**
Function Description	OLED screen update display, loop write SSD1306_Buffer data to screen
Input Parameters	None
Return Value	None

SSD1306_Fill

Function Prototype	**void SSD1306_Fill(SSD1306_COLOR_t color)**
Function Description	OLED screen clear (not refreshed display), fill SSD1306_Buffer with all black or all white according to color
Input Parameters	color (SSD1306_COLOR_BLACK or SSD1306_COLOR_WHITE)
Return Value	None

SSD1306_DrawPixel

Function Prototype	void SSD1306_DrawPixel(uint16_t x, uint16_t y, SSD1306_COLOR_t color)
Function Description	Draw pixel at specified coordinates (x,y), set SSD1306_Buffer according to color and whether inverted
Input Parameters	x (horizontal coordinate), y (vertical coordinate), color (SSD1306_COLOR_BLACK or SSD1306_COLOR_WHITE)
Return Value	None

SSD1306_GotoXY

Function Prototype	void SSD1306_GotoXY(uint16_t x, uint16_t y)
Function Description	Set current cursor position
Input Parameters	x (horizontal coordinate), y (vertical coordinate)
Return Value	None

SSD1306_Putc

Function Prototype	char SSD1306_Putc(char ch, FontDef_t *Font, SSD1306_COLOR_t color)
Function Description	Draw single character, draw character at current cursor position according to font and color and update cursor
Input Parameters	ch (character to draw), Font (font structure pointer), color (character color)
Return Value	Return drawn character on success, return error character on failure

SSD1306_Puts

Function Prototype	char SSD1306_Puts(char *str, FontDef_t *Font, SSD1306_COLOR_t color)
Function Description	Draw string, call SSD1306_Putc one by one to draw each character in the string
Input Parameters	str (pointer to string to draw), Font (font structure pointer), color (character color)
Return Value	Return '\0' at end of string on success, return error character on failure

OLED_ON

Function Prototype	void OLED_ON(void)
Function Description	Wake OLED from sleep, send wake command
Input Parameters	None
Return Value	None

OLED_OFF

Function Prototype	void OLED_OFF(void)
Function Description	Put OLED to sleep (very low power consumption in sleep mode), send sleep command
Input Parameters	None
Return Value	None

OLED_Clear

Function Prototype	void OLED_Clear(void)
Function Description	OLED clear screen, call SSD1306_Fill to fill with black
Input Parameters	None
Return Value	None

OLED_Refresh

Function Prototype	void OLED_Refresh(void)
Function Description	Refresh OLED screen, call SSD1306_UpdateScreen
Input Parameters	None
Return Value	None

OLED_Draw_String

Function Prototype	void OLED_Draw_String(char *data, uint8_t x, uint8_t y, bool clear, bool refresh)
Function Description	Write characters, clear screen according to parameters, set cursor position to draw string, refresh screen
Input Parameters	data (string to display), x (horizontal coordinate), y (vertical coordinate), clear (whether to clear screen), refresh (whether to refresh)
Return Value	None

OLED_Draw_Line

Function Prototype	**void OLED_Draw_Line(char *data, uint8_t line, bool clear, bool refresh)**
Function Description	Write a line of characters, call OLED_Draw_String to display on corresponding line according to line number (1-3)
Input Parameters	data (string to display), line (line number, 1-3), clear (whether to clear screen), refresh (whether to refresh)
Return Value	None

SSD1306_DrawLine

Function Prototype	**void SSD1306_DrawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, SSD1306_COLOR_t c)**
Function Description	Draw line segment, draw line segment between (x0,y0) to (x1,y1)
Input Parameters	x0 (start horizontal coordinate), y0 (start vertical coordinate), x1 (end horizontal coordinate), y1 (end vertical coordinate), c (line segment color)
Return Value	None

VI. Experiment Phenomenon

After the program is flashed, it will run the flash self-test program, write "yahboom" to flash, then read flash content. If it is "yahboom", the OLED screen will display "Flash RDWT OK!"