INSTITUTO POLITÉCNICO NACIONAL - UPIIZ

Practica 08

Comunicación I2C y SPI

Bernardo Martinez Medina

Adrián Torres Aranda

Fecha de Entrega: 14/06/2022

Programa Académico: Implementación de sistemas digitales

Docente: Ramon Jaramillo Martínez

8to Semestre

ING MECATRÓNICA

- # Resumen.

Analizar, diseñar e implementar una interfaz de entrada/salida, para resolver un problema específico por medio del módulo I2C y SPI.

- # Introducción.

El I2C es un bus de tipo serial que utiliza dos pines: SDA (Slave Data Adress) y el SCL (Slave Clock) que funcionan de la siguiente manera:

1. Se envía una señal al pin SCL para indicar que se quiere iniciar una transferencia.
2. Los siguientes 7 bits que se envían es la dirección del dispositivo subordinado, hay un límite de 128 dispositivos subordinados.
3. Los siguientes 8 bits son la transferencia de datos entre el dispositivo maestro y el subordinado [1].

La mayoría de los dispositivos con una interfaz I2C la utilizan para realizar actualizaciones del firmware interno o simplemente para iniciar el protocolo de comunicación entre dos dispositivos.

El SPI es un protocolo de comunicación ampliamente utilizado por una gran cantidad de dispositivos, transmite los datos ya que funciona con los siguientes pines:

1. MOSI (Master Out Slave In): Este pin indica que el dispositivo maestro es el que envía el dato hacía el dispositivo subordinado.
2. MISO (Master In Slave Out): Si este pin este activo entonces son los dispositivos subordinados quienes envían los datos hacía el dispositivo maestro.
3. SCLK: Envía la señal se reloj, la cual es necesaria para enviar los datos tanto desde la interfaz SPI maestro como desde la interfaz SPI subordinada. La velocidad de reloj indica la cantidad de datos que se van a enviar por segundo.
4. CS (Chip Select): Si este pin este activo entonces se pueden enviar los datos a ese dispositivo subordinado. Un dispositivo SPI maestro va a tener tantos pines CS como dispositivos SPI subordinados. haya en el sistema [1].

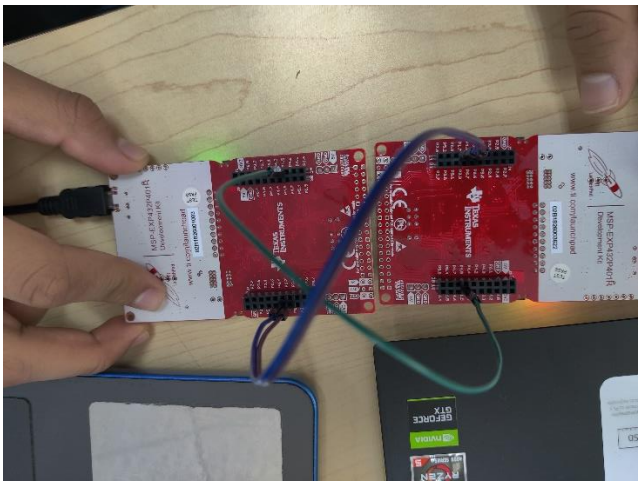
- # Desarrollo

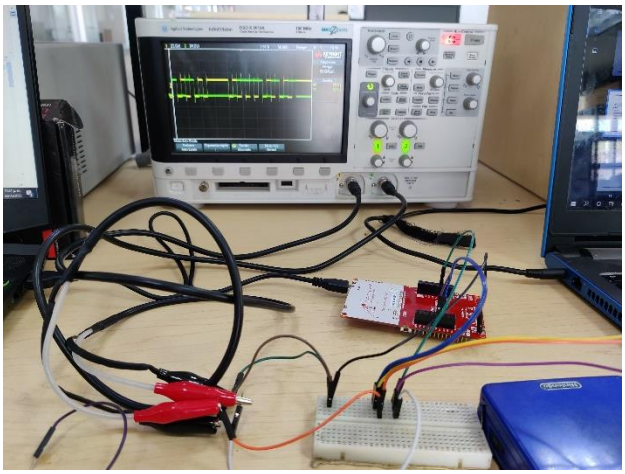
1. Eco a través del protocolo SPI utilizando 3 cables.

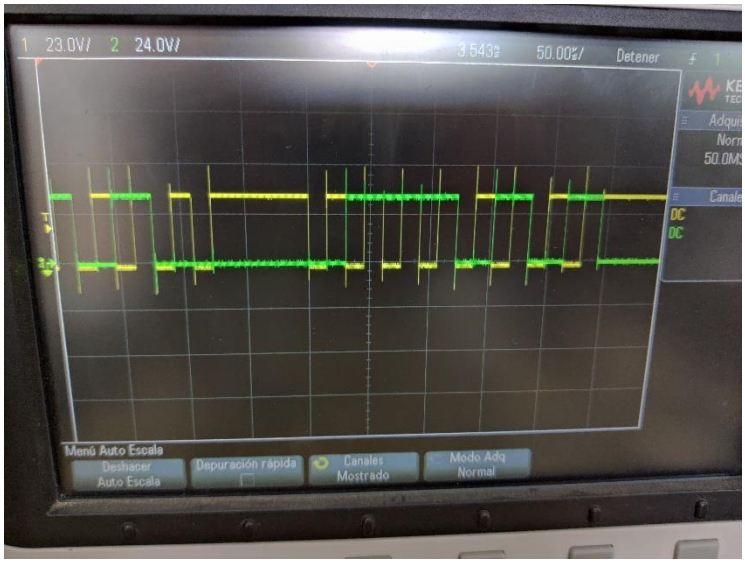
```
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
```

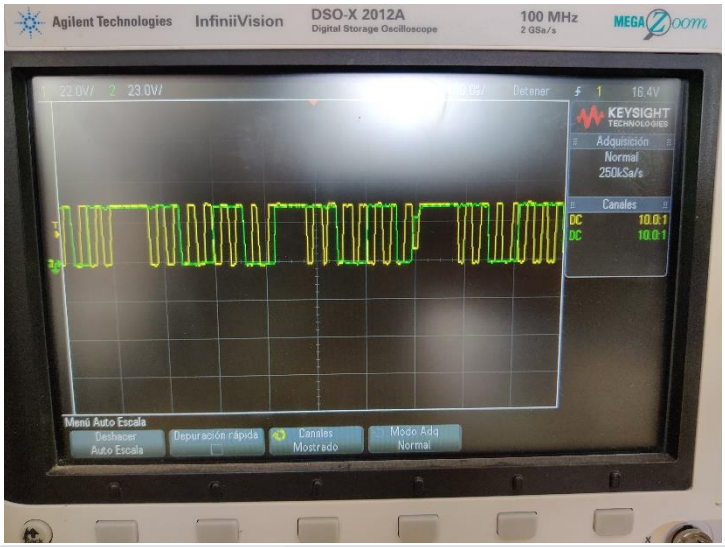
```c
// Definicion de variables
volatile uint16_t RxData;
volatile uint16_t TxData;
volatile uint16_t DatoPrevio;


int main(void)
{
    /* Stop Watchdog  */
volatile uint32_t i;

 WDT_A -> CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; // Detenemos watch dog
//Configuracion de GPIOS
 P1->DIR |- BIT0;
 P1->OUT &= ~BIT0;
 // Configuración de senales de
 // UCB0SIMO
 P1 -> SEL0 |= BIT5 + BIT6 + BIT7;
 // Reinicio de estado de maquina de eusib0
 EUSCI_B0-> CTLW0 |= EUSCI_B_CTLW0_SWRST;
 EUSCI_B0-> CTLW0 = EUSCI_B_CTLW0_SWRST + // Estado de maquina en reset
         EUSCI_B_CTLW0_MST+ // Seleccion SPI modo maestro
         EUSCI_B_CTLW0_SYNC+ // Habilitamos modo sincrono
         EUSCI_B_CTLW0_CKPL+ // Seleccionamos polaridad reloj en alto
         EUSCI_B_CTLW0_MSB+  // Primero se envia el bit mas significativo
         EUSCI_B_CTLW0_SSEL__ACLK; // Seleccion de fuente de reloj
     EUSCI_B0 -> BRW = 1;
     // Inicializamos el estado de maquina
     EUSCI_B0 -> CTLW0 &= ~ EUSCI_B_CTLW0_SWRST;
 // Buffer de Tx
     TxData=1;
    // Interrupcion global
     __enable_irq();
     NVIC -> ISER[0] = 1 << ((EUSCIB0_IRQn)&31);
     // Desíerta al salir de la rutina de la interrupcion
     SCB-> SCR &= ~SCB_SCR_SLEEPDEEP_Msk;
     // Garantizamos que la rutina anterior tenga efecto inmediato
     __DSB();

    while(1)
    {
        EUSCI_B0-> IFG |= EUSCI_B_IFG_TXIFG;
        EUSCI_B0-> IE |= EUSCI_B_IE_TXIE;
        // Modo bajo consumo
        __sleep();
        __no_operation();
        DatoPrevio = TxData=1;
        if(RxData != (DatoPrevio)){
            P1 -> OUT |= BIT0;
        }
        else {
            P1 -> OUT &= ~BIT0;
        }
        // Retardo crudo
        for(i=3000; i>0; i--);
        // Aumentamos dato a enviar
        TxData++;

    }
}
    void EUSCIA0_IRQHandler(void){
        if (EUSCI_B0 -> IFG & EUSCI_B_IFG_TXIFG){
            // CArgamos bufer de transmision
            EUSCI_B0 -> TXBUF =TxData;
            // Deshabilitamos interrupciones por tx
            EUSCI_B0 -> IE &= ~EUSCI_B_IFG_TXIFG;
            // espera a recibir el dato
            while (!(EUSCI_B0 -> IFG & EUSCI_B_IFG_RXIFG));
            // Guardamos dato
```

```
    RxData = EUSCI_B0->RXBUF;
    EUSCI_B0 -> IFG &= ~EUSCI_B_IFG_RXIFG;
}
    }
```





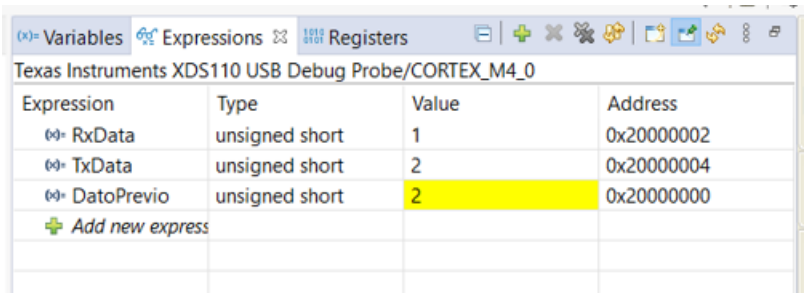


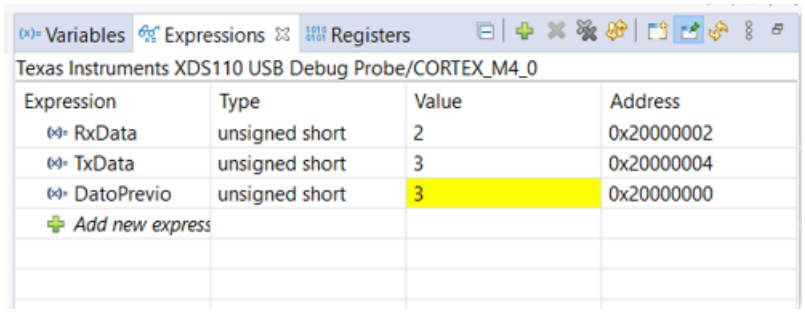


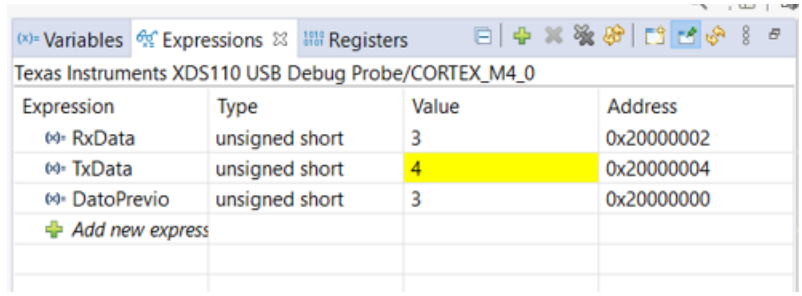
Variables | Expressions | Registers

Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0

Expression	Type	Value	Address
RxData	unsigned short	1	0x20000002
TxData	unsigned short	2	0x20000004
DatoPrevio	unsigned short	1	0x20000000
Add new express			







2. Implementar una comunicación entre un maestro y 4 esclavos.

- # Análisis de Resultados

Al implementar la práctica se tuvo una muestra de lo que es el protocolo spi a partir de dos dispositivos, en este caso son la tarjeta, done una actúa como esclavo y otra como maestro. Un maestro que será aquel dispositivo encargado de transmitir información a sus esclavos. Los esclavos serán aquellos dispositivos que se encarguen de recibir y enviar información al maestro.

Al igual que en el protocolo UART el protocolo SPI requiere configuraciones que consisten en indicar el bit de paridad, aunque podemos destacar las ventajas encontradas a lo largo de la práctica que entre ellas está la velocidad de transmisión ya que es configurable a través de software y dependerá también de los dispositivos utilizados en el sistema. Con respecto a otros protocolos seriales que trabajan a modo half duplex, el SPI tiene velocidades de transmisión mucho

mayores debido a que éste trabaja en modo full duplex. Otros parámetros configurables a través de software son la frecuencia del relo, entre otras.

- # Conclusiones

Adrián Torres Aranda

Al realizar la practica los primeros puntos se realizaron con facilidad con solo un maestro y un esclavo ya que con lo visto en clase y además realizar códigos similares se pudo completar los puntos, pero el implementar un maestro con 4 esclavos fue la parte más confusa y no se completó esa parte, pero el profesor nos mostró mando el código

Bernardo Martínez Medina

Por medio de esta práctica se revisó la comunicación del tipo SPI que está disponible en microcontroladores con la cual al ver características la más importante es su capacidad de conectarse con diferentes dispositivos a la vez. Esto por medio de un maestro y la cantidad de esclavos que se requieran. Aunque en ciertos momentos de la practica este protocolo presento problemas donde la comunicación no va del todo bien.

Al investigar este problema se dio que podría ser por ruido electromagnético o por el pin SS que suelen tener varias placas basadas en microcontroladores. Aunque de cierta forma con ese problema al spi se le encuentra una mayor utilidad cuando se requiere alta velocidad y comunicación entre varios dispositivos.

- # Referencias.

[1] «HZ hard zone, » 09 diciembre 2020. [En línea]. Available: https://hardzone.es/reportajes/que-es/i2c-spi-uart/. [Último acceso: 18 junio 2022].