EMBEDDED SYSTEM DESIGN
LAB ASSIGNMENT -6

IMPLEMENTATION OF SERIAL PERIPHERAL INTERFACE (SPI) FOR LPC2148 MICROCONTROLLER

SUBMITTED BY- SANCHIT AGRAWAL (2016H140106)
ABHINAV GARG (2016H140103)

Objective –

Objective of this experiment is to learn and implement SPI (Serial Peripheral Interface) for LPC2148 microcontroller. LPC2148 microcontroller.

To do-

Send continuous data to 74HC595 from 0 - 255 with a delay of 1 second for between 2 consecutive values. 8 LEDs should display the count from 0-255 with 1 second interval between changing values.
System should repeat this task infinitely

Overview

SPI is a Master - Slave protocol, one device acts as the bus master while any number of devices can act asslaves. SPI is a serial synchronous protocol that means all the data going in or out is synchronized by a clock. SPI device have separate lines for data out and data in.
All SPI based communication happens over 4 lines named MOSI, MISO, SCK, SS.
MOSI - Master Out Slave In (master device sends data over this line)
MISO - Master In Slave out (master device accepts data over this line)
SCK - Serial clock (generated by bus master)
SS - Slave select (Useful in the case of multiple slaves)

IC 74HC595 is a 8 bit serial to parallel data converter, it takes data-in over serial interface and latches that data in the 8 bit parallel output. 8 LEDs are connected to LPC2148 SPI output through 74595 in our experiment board.

Project Workspace-

- Used keil u-vison5..
- Under ARM section, select the board as LPC 2148.
  Include all the header files and the source files that are required for the program
- Workspace consists of all the library files.
- Then, after ARM programming, select the target device that is to be programmed and burn the bin file into the device.
- Provided inter chip communication using SPI, sending it to serial to parallel data converter(74HC595) and displaying it on LEDs present on the board.

Critical Issues

- SHCP should be triggered from ‘low’ to ‘high’ on every data transmission to 74HC595,
- Synchronization of data transmission with the clock.

Register Description for SPI Communication: -

Name	Description	Access	Reset value[1]	Address
S0SPCR	SPI Control Register. This register controls the operation of the SPI.	R/W	0x00	0xE002 0000
S0SPSR	SPI Status Register. This register shows the status of the SPI.	RO	0x00	0xE002 0004
S0SPDR	SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI0 by writing to this register. Data received by the SPI0 can be read from this register.	R/W	0x00	0xE002 0008
S0SPCCR	SPI Clock Counter Register. This register controls the frequency of a master’s SCK0.	R/W	0x00	0xE002 000C
S0SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	R/W	0x00	0xE002 001C

Methodology –

- Out of SPI0 and SPI1 we selected SPI0 for implementation. SPI0 is operated in master mode, and IC 74HC595 which is a 8 bit serial to parallel data converter is selected as slave.
- Master clock frequency is prescaled by factor of 8.
- Port pin no.P0.7 is connected to SHCP and it acts as a internal clock such that in every clock pulse (i.e. pin status 0 to 1) data is latched on LED's.
- Based on data latched and counters count, the count sequence from 0 to 255 is shown on the LED's.

Hardware Design – Used LPC2148 board and all the required things such as LED's and IC were present on board.

SCHEMATIC

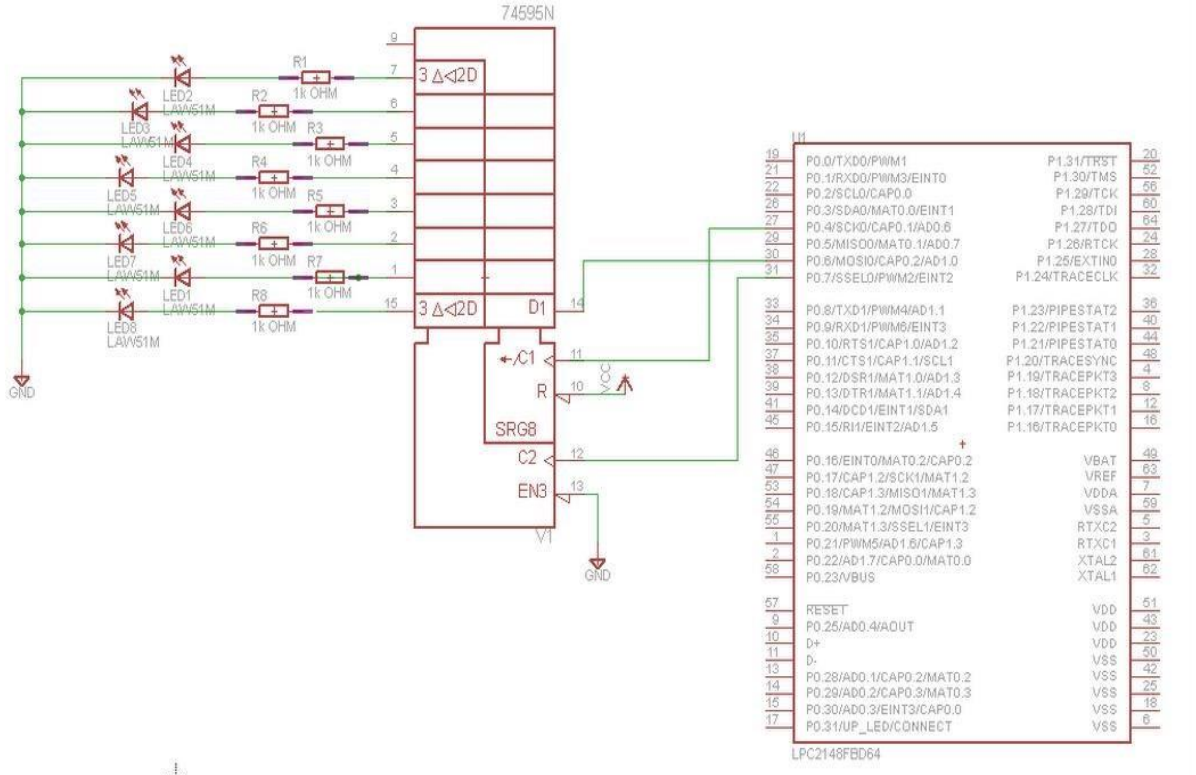
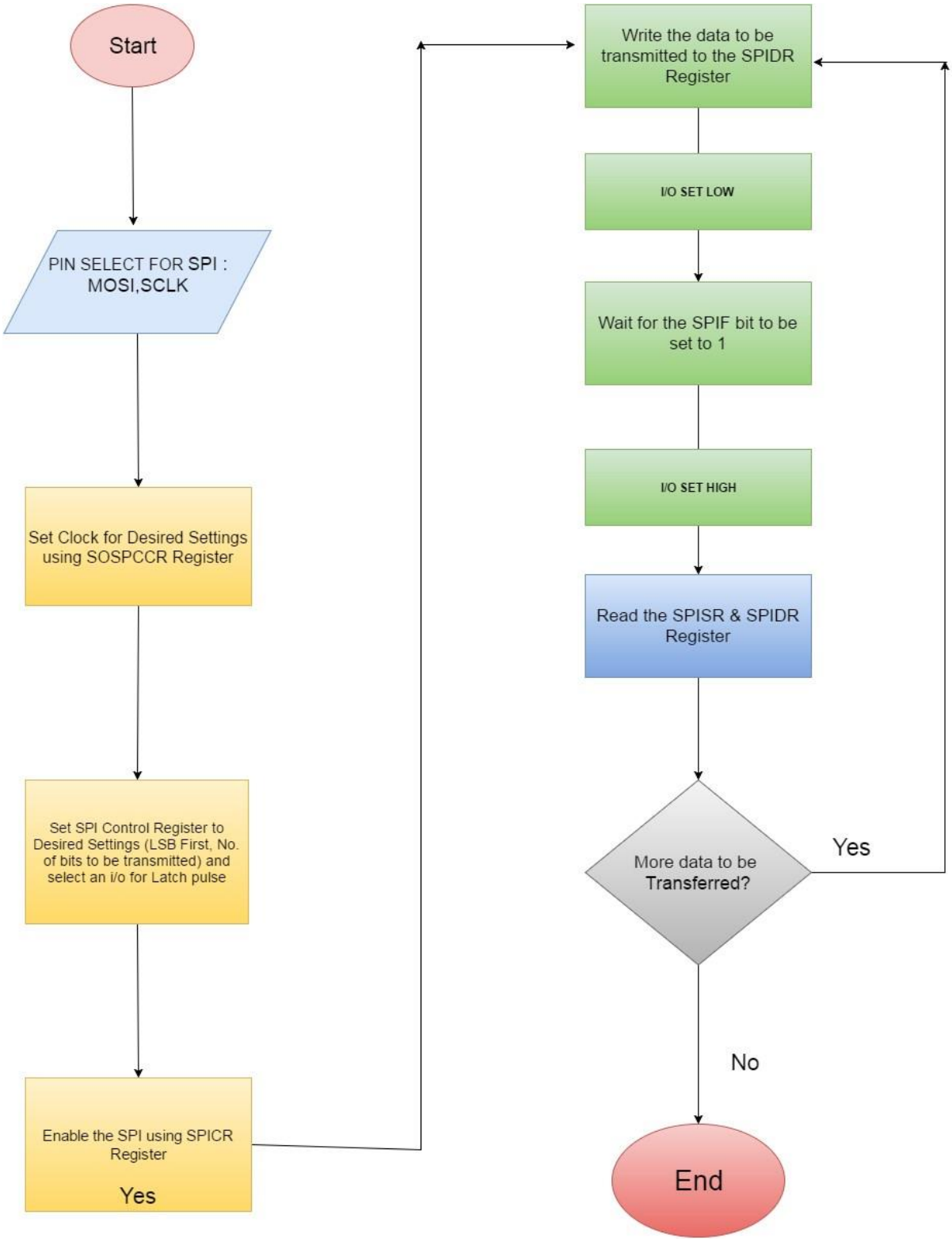


Figure above shows the schematic for the interfacing SPI using LPC2148

FLOW CHART


Start
PIN SELECT FOR SPI : MOSI,SCLK
Set Clock for Desired Settings using SOSPCCR Register
Set SPI Control Register to Desired Settings (LSB First, No. of bits to be transmitted) and select an i/o for Latch pulse
Enable the SPI using SPICR Register
Yes
Write the data to be transmitted to the SPIDR Register
I/O SET LOW
Wait for the SPIF bit to be set to 1
I/O SET HIGH
Read the SPISR & SPIDR Register
More data to be Transferred?
Yes
No
End

Software Design –

Used Basic SPI initializing instructions to set it in master mode as well as for other configurations.

Code Explanation –

- The Header file (LPC214x.h) includes all the input-output functions and their definitions.
- Functions such as delay and SPI initialisation wew declared after the header file.
- In SPI initialisation, SPI0 is configured and MOSI operation is selected. We connected SHCP to pin P0.7 by setting PINSEL0 to 0x00001100
- Master clock was prescaled by factor of 8 by setting S0SPCCR (clock prescaler control register) to 8.
- SPI0 control register S0SPCR is set to 0x0860 so as to select master mode,
  8 bit data stream, CPOH=0, CPOL=0
- SPI is enabled by writing 1 on 6^{th} bit of S0SPCR (SPI control reg.)
- CPOH=0, CPOL=0 is set such that data is latched in the rising edge and transmitted on falling edge.
- Delay function is introduced so as to provide required delay between the counts.
- Connected port pin P0.7 to SHCP such as as soon as pin is set, data is latched to the LED's.
- PB0 is configured as output port to latch STCP.
- Data to be transmitted is placed on S0SPDR (data reg.) and until that time SPIF bit of S0SPSR (status reg.) is 0
- Now, as soon as 8 bit stream is transmitted from TX buffer, SPIF bit becomes 1 and data is latched on LED's by setting pin P0.7 .
- So following on the above steps, in accordance to the loop count from 0 to
  255, respective values were shown with the help of data latched on LED's.

The c code for the above is sent as an attachment for reference.

Observation & Result –

Data was transmitted from SPI and latched successfully to LED's.
The count sequence from 0 to 255 was successfully shown with the help of LED's according to data latched.

Conclusion

Data was transmitted over the MOSI Pin to serial to Parallel converter and displayed as binary pattern on the LED.

References-

LPC214x user manual.