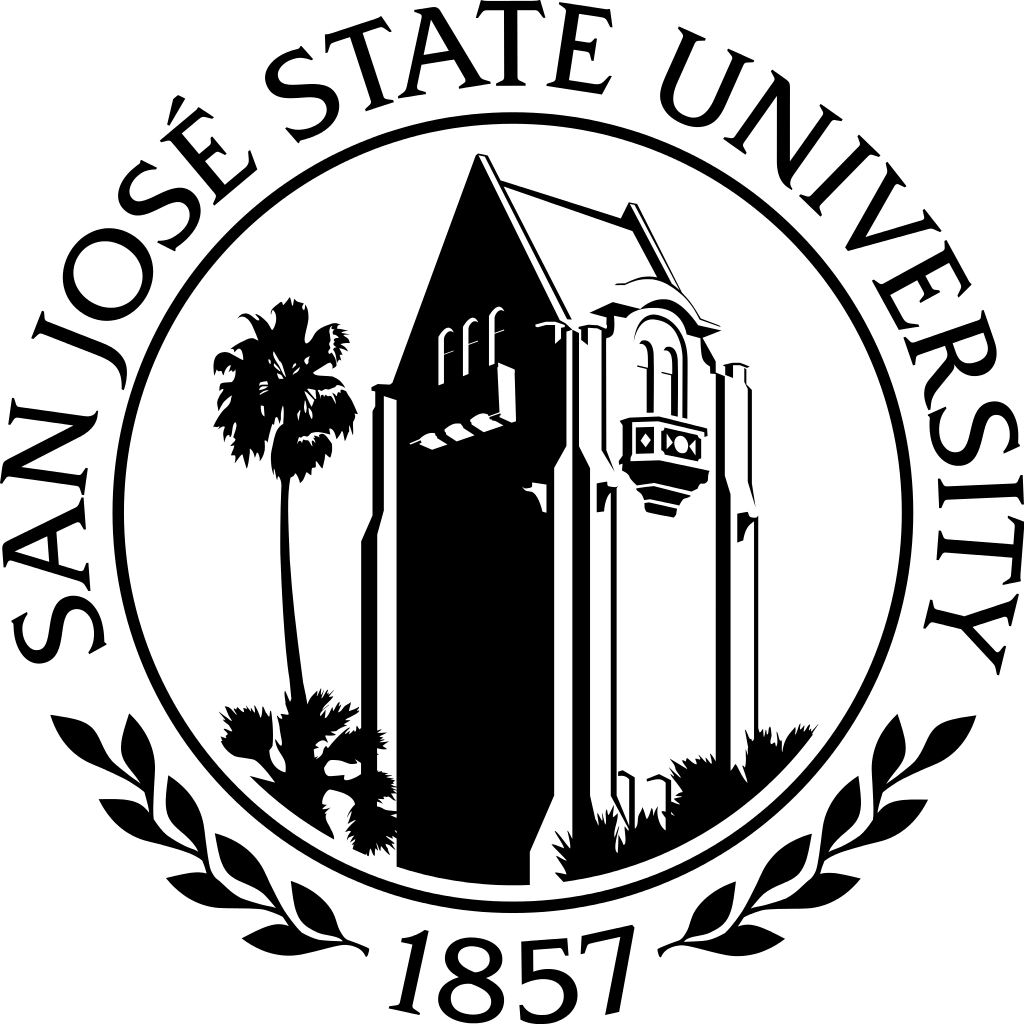
****

**CmpE Internship Summer 2018**

**Design Document**

***LPC40xx SSP Driver***

**Authors & Editors:**

**Khrysta Finch**

**Revision History:**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Description** |
| **A.1** | **16/6/18** | **Initial Design Documentation Draft** |

Table of Contents

[**Objective**](#_f3ut0x3v9lxf) **2**

[**Background**](#_52hjxc9p7ctt) **2**

[**Overview**](#_yy2djj5i8omd) **3**

[**Detailed Design**](#_avw48lx4cmyr) **4**

[**Caveats**](#_op45shsa5n8y) **5**

[**Testing Plan**](#_eejuf5av24ql) **5**

[**Unit testing scheme**](#_h5m3fobrdlly) **5**

[**Integration Testing**](#_fubwzje1thtf) **5**

[**Demonstration Project**](#_6z1lojsl95dl) **5**

# **Objective**

The objective of this driver is to provide a generic SSP template for the LPC40xx series of microcontrollers. The template must allow the user to configure any of the three SSP ports as needed.

# **Background**

LPC40xx User Manual, see Chapter 21: <https://www.nxp.com/docs/en/user-guide/UM10562.pdf>

# **Overview**

One page-ish high level description of what you are doing. Should be understandable by other engineers.

The PMOD 16-button input device allows for a specified input to be run into the machine before outputting to the output module PMOD Character LCD display.

The SSP driver should have two capabilities; 1) a default master mode and 2) customization on any of the SSP interfaces available.

Default master mode will set the following:

* Enable SSP0
* Master mode
* 8-bit data size transfer
* SPI frame format
* Clock Polarity and Phase
* Serial Clock Rate
* Clock Prescaler

Customization will include:

* Enable/Disable SSP0/1/2
* Master/slave mode
* Data size for transfers
* Frame format type
* Clock Out Polarity and Phase
* Serial Clock Rate
* Clock Prescaler

Both options must have the ability to either send or receive data based on the master/slave mode selected.

# **Detailed Design**

There will be two files associated with this driver, an lpc40\_ssp.hpp and lpc40ssp.cpp. The header file will contain the SSP class declarations while the source file with contain the class’s function definitions.

The SSP class will contain the following:

* A private LUT to select between SSP0, SSP1, or SSP2 peripherals.
* A private LUT to select the pins for the selected interface.
* A private LUT to select the data size.
* init() function that will accept the following parameters:
  + An SSP interface
  + Port-pins for SSEL, SCK, MISO, and MOSI
  + Operation mode (master or slave)
  + Data size, DSZE
  + Frame format, FRFT
  + Clock polarity, CPOL
  + Clock phase, CPHA
  + Clock rate, SCR
  + Clock prescaler, CPSR

From here, the appropriate interface can be enabled through PCONP and the peripheral clock(PCLKSEL) and other configurations will be set here.

If any of the peripherals are configured in master mode, the PCLKSEL must be scaled down using the prescaler (CPSDVSR). The prescaler can be set to any even number from 2-254.

Tables 1-6 at the end of this section list all the details for all configurations in this function.

* init\_master() function takes no parameters, but will send the following to init():
  + SSP = SSP0
  + SSEL = 0x0010, SCK = 0x000F, MISO = 0x0011, MOSI = 0x0012
  + Mode = master (1)
  + Data size = 7
  + Format = 0
  + CPOL = 1
  + CPHA = 0
  + SCR = 0
  + CPSR = 2
* get\_status() function will check and return the value for the fifth bit of the Status Register to see if the TX/RX FIFO is busy sending/receiving data.
* transfer() function that will accept the following parameters:
  + An SSP interface
  + Data packet – will need to be a union to work with variable data sizes options

This function will be used for both sending and receiving data.

Table 1: PCONP Register, resets (or disabled) when Bit = 0

|  |  |  |
| --- | --- | --- |
| Bit | Symbol | Description |
| 10 | PCSSP1 | SSP1 clock control bit |
| 20 | PCSSP2 | SSP2 clock control bit |
| 21 | PCSSP0 | SSP0 clock control bit |

Table 2: Peripheral Clock(PCLKSEL)

|  |  |
| --- | --- |
| Bit | Description |
| 4:0 | 0 = The divider is turned off., no clock will be provided to APB peripherals.  1 = The input clock is divided by 1 to produce the APB peripheral clock.  2 = The input clock is divided by 2 to produce the APB peripheral clock.  3 = The input clock is divided by 3 to produce the APB peripheral clock.  4 = The input clock is divided by 4 to produce the APB peripheral clock. This is also the value on reset. |

Table 3: SSP Control Register 1 (CR1)

|  |  |  |
| --- | --- | --- |
| Bit | Value | Description |
| 0 | 0 | Loop Back Mode during normal operation |
| 1 | Serial IN taken from Serial OUT rather than input pin |
| 1 | 0 | SSP controller disabled |
| 1 | SSP controller enabled |
| 2 | 0 | Controller is master and drives SCLK, MOSI, and SSEL |
| 1 | Controller is slave and drives MISO |
| 3 | Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SSP controller from driving the transmit data line (MISO). | |

Table 4: Status Register

|  |  |
| --- | --- |
| Bit | Description |
| 0 | Transmit FIFO Empty (TFE). Set to 1 if TFE is empty |
| 1 | Transmit FIFO Not Full (TNF). Set to 0 if full |
| 2 | Receive FIFO Not Empty (RNE). Set to 0 if empty |
| 3 | Receive FIFO Full (RFF). Set to 1 if full |
| 4 | Busy, set to 0 when idle and 1 when TX/RX or TxFIFO is not empty |

Table 5: SSP interface pin connections.

|  |  |  |  |
| --- | --- | --- | --- |
| SSP Interface | Name | Port-Pin | Value |
| SSP0 | SSP0\_SCK | LPC\_IOCON->P0\_15 | 2 |
| LPC\_IOCON->P1\_20 | 5 |
| LPC\_IOCON->P2\_22 | 2 |
| SSP0\_SSEL | LPC\_IOCON->P0\_16 | 2 |
| LPC\_IOCON->P1\_21 | 4 |
| LPC\_IOCON->P1\_28 | 5 |
| LPC\_IOCON->P2\_23 | 2 |
| SSP0\_MISO | LPC\_IOCON->P0\_17 | 2 |
| LPC\_IOCON->P1\_23 | 5 |
| LPC\_IOCON->P2\_26 | 2 |
| SSP0\_MOSI | LPC\_IOCON->P0\_18 | 2 |
| LPC\_IOCON->P1\_24 | 5 |
| LPC\_IOCON->P2\_27 | 2 |
| SSP1 | SSP1\_SCK | LPC\_IOCON->P4\_20 | 3 |
| LPC\_IOCON->P1\_19 | 5 |
| LPC\_IOCON->P1\_31\* | 2 |
| LPC\_IOCON->P0\_7\* | 2 |
| SSP1\_SSEL | LPC\_IOCON->P4\_21 | 3 |
| LPC\_IOCON->P0\_6 | 2 |
| LPC\_IOCON->P0\_14 | 2 |
| LPC\_IOCON->P1\_26 | 5 |
| SSP1\_MISO | LPC\_IOCON->P4\_22 | 3 |
| LPC\_IOCON->P1\_18 | 5 |
| LPC\_IOCON->P0\_12\* | 2 |
| LPC\_IOCON->P0\_8\* | 2 |
| SSP1\_MOSI | LPC\_IOCON->P4\_23 | 3 |
| LPC\_IOCON->P1\_22 | 5 |
| LPC\_IOCON->P0\_13\* | 2 |
| LPC\_IOCON->P0\_9\* | 2 |
| SSP2 | SSP2\_SCK | LPC\_IOCON->P1\_0 | 4 |
| LPC\_IOCON->P5\_2 | 2 |
| SSP2\_SSEL | LPC\_IOCON->P1\_8 | 4 |
| LPC\_IOCON->P5\_2 | 2 |
| SSP2\_MISO | LPC\_IOCON->P1\_4 | 4 |
| LPC\_IOCON->P5\_1 | 2 |
| SSP2\_MOSI | LPC\_IOCON->P1\_1 | 4 |
| LPC\_IOCON->P5\_0 | 2 |

\* - Glitch filtering option available.

Table 6: SSP Control Register 0 (CR0)

|  |  |  |
| --- | --- | --- |
| Bit | Value | Description |
| 3:0 | Data size selects the # of bits sent per frame, resets to 0000 | |
| 0011 | 4-bit transfer |
| 0100 | 5-bit transfer |
| 0101 | 6-bit transfer |
| 0110 | 7-bit transfer |
| 0111 | 8-bit transfer |
| 1000 | 9-bit transfer |
| 1001 | 10-bit transfer |
| 1010 | 11-bit transfer |
| 1011 | 12-bit transfer |
| 1100 | 13-bit transfer |
| 1101 | 14-bit transfer |
| 1110 | 15-bit transfer |
| 1111 | 16-bit transfer |
| 5:4 | Frame format for data transfers | |
| 00 | SPI |
| 01 | TI |
| 10 | Microwire |
| 6 | Clock Polarity – only available for SPI | |
| 0 | Controller maintains bus clock low between frames. |
| 1 | Controller maintains bus clock high between frames. |
| 7 | Clock Phase – only available in SPI | |
| 0 | Controller captures serial data on the first clock transition of the frame. |
| 1 | Controller captures serial data on the second clock transition of the frame. |
| 15:8 | Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR \* [SCR+1]). | |

# **Caveats**

Assuming the display functionality for the SJTwo is not yet available, terminal access will be required, possibly through an IDE. If terminal access is not available, LEDs can be used as blink codes for errors.

There are many port-pin combinations available for the SSP interfaces depending on the type of IOCON register being used. For initial design and testing, Type D IOCON register will be used and only one port-pin combination will be used with each interface. Pins selected will be chosen based on the initial schematic provided. Pin options may change depending on schematics.

No DMA capabilities at this time.

Raw and masked interrupts will not be used at this time.

# **Testing Plan**

### **Unit testing scheme**

Initially, all functions will use a Debug\_printf statements that will print any arguments passed to the function and whether or not the function executed properly. Error messages will include where and what triggered the error. SSP0 will be unit tested first. Testing will include both master and slave modes for each SSP interface in both sending and receiving data.

All debugging will be performed in either the LPCXpresso or MCUXpresso IDE’s.

### **Integration Testing**

Integration with the SJTwo will require the addition of an SJOne board and the two boards will perform a simple addition problem.

SJTwo in master mode:

The SJTwo must send two numbers to the SJOne board. The SJOne then performs the addition, displays the solution on its 7-segment display, and then sends the solution back to the SJTwo which will be displayed on a terminal.

SJTwo in slave mode:

Similar to the master mode test, except the roles of the SJTwo and SJOne are switched.

### **Demonstration Project**