Application Note
On
SPI Communication using UART
Introduction

In embedded system design the serial communication is often performed by the UART, IIC and SPI. The former two-methods are asynchronous. Hence, the data communication is relatively slow compare to the synchronous - SPI bus. Since Fujitsu microcontroller doesn't have dedicated SPI peripheral, therefore, in order to perform the SPI communication, Fujitsu recommends using the UART or SIO peripheral. Using these peripheral for SPI communication is very simple and generates very less software overheads. The purpose of this application note is to present SPI communication using Fujitsu UART. For the demonstration purpose, SPI device M95040, (ST serial SPI EEPROM) is used here.

SPI overview

SPI is a full duplex, synchronous serial bus developed by Motorola. The bus communication is based on Master-slave configuration. SPI consists of four signals:

MOSI: Master Out Slave In
MISO: Master In Slave Out
SCK: Serial Clock
SS: Slave select

In SPI transfer, data is simultaneously transmitted and received. The data is clocked simultaneously into Slave and Master based on SCK pulses supplied by the master. The slave/chip select pin selects the slave device for the communication. The baud rate can be achieved as high as 10 Mbit/s depending upon the SPI device selection.

Fujitsu UART

Fujitsu UART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It is capable of performing both synchronous and asynchronous communication with external devices. In general, the UART support following modes of communication:

- Normal Mode (provides synchronous and asynchronous bi-directional communication function)
- Multiprocessor mode (Provide master-slave communication function in master/slave systems)
- LIN-bus mode (working both as master or as slave device).

The first two modes are standard feature of UART. However, the LIN bus feature is available only in limited series. Some of the examples are MB90340 series, MB90350 series, and MB90360 series.

In general the UART supports the following features:

- Full duplex double buffer
- Supports CLK synchronous and CLK asynchronous start-stop data transfer.
- One-to-n communication (one master to n slaves) in multi processor mode.
• Supports flexible baud rate setting using an external clock input or internal timer.
• Variable data length (7 to 9 bits, [no parity]; 6 to 8 bits [with parity]).
• Error detect function (framing, overrun, and parity)
• Interrupt function (receive and transmit interrupts, LIN-Synch-break detect)
• NRZ type transfer format
• Special synchronous Clock Mode for delaying clock (useful for SPI)

The chart below shows the data transfer format supported in various UART modes:

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>Data length</th>
<th>Synchronization of mode</th>
<th>Length of stop bit</th>
<th>data bit direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>normal mode</td>
<td>7 or 8</td>
<td>asynchronous</td>
<td>1 or 2</td>
</tr>
<tr>
<td>1</td>
<td>multiprocessor</td>
<td>7 or 8 + 1</td>
<td>--</td>
<td>1 or 2</td>
</tr>
<tr>
<td>2</td>
<td>normal mode</td>
<td>8</td>
<td>synchronous</td>
<td>0, 1 or 2</td>
</tr>
<tr>
<td>3</td>
<td>LIN mode</td>
<td>8</td>
<td>asynchronous</td>
<td>1</td>
</tr>
</tbody>
</table>

1: means the data bit transfer format: LSB or MSB first.
2: "+1:" means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

EEPROM features

M95040 is a 4K-byte memory, composed of two pages of 2K bytes. All address, data and instructions are shifted out MSB first. The device has write protect pin to disable the write operation. The device also allows taking pause temporarily during serial communication using hold signal. M95040 has a status register to check the status of write operation. The status register configuration is described below:

1 1 1 1 BP1 BP0 WEL WIP

Here BP1 and BP0 bits of the register are used for write protection. The WEL bit indicates the status of write enable latch. The WIP bit indicates the status of write operation.

Instruction set

Typically the EEPROM device has six instructions set for communication. The table below shows the instruction set for the M95040 EEPROM for various operations:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Instruction Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>WREN</td>
<td>Set write enable latch</td>
<td>0x06</td>
</tr>
<tr>
<td>WRDI</td>
<td>Reset write enable latch</td>
<td>0x04</td>
</tr>
<tr>
<td>RDSR</td>
<td>Read status register</td>
<td>0x05</td>
</tr>
<tr>
<td>WSR</td>
<td>Write status register</td>
<td>0x01</td>
</tr>
<tr>
<td>READ</td>
<td>Read data from Memory</td>
<td>0000 A8,011</td>
</tr>
<tr>
<td>WRITE</td>
<td>Write data to Memory</td>
<td>0000 A8,010</td>
</tr>
</tbody>
</table>

Notes:  
- It is required to select the device before sending the instructions command from UART.
- A8 = 1, Upper page selected
Interfacing UART to a SPI device

The diagram below shows the connection between Fujitsu microcontroller with an SPI device. In the SPI network the number of devices to be connected on the bus depends on the slave select pins on the master device. However, Fujitsu doesn't have dedicated slave select pin, therefore it is required to use the general-purpose pin for slave select pins.

The above diagram shows Fujitsu MCU as a master and SPI device, EEPROM M95040 as a slave device. The EEPROM device is configured for write protection-disabled mode with no pause while serial communication. Since there is no dedicated chip select signal in microcontroller, a general-purpose IO pin ‘Pxx’ is used as a slave select pin.

UART Configuration

Typically, UART input/output signal pins in microcontroller have dual functionality. Therefore, it is required to configure these pins as a UART pins. Since SPI is a synchronous communication, the UART should be configured for clock synchronous mode communication. These all settings are done in the UART serial mode register, SMR1.

Set this register as follows:
- MD1, MD0 = 10 (Set the UART as a Clock synchronous mode)
- CS2, CS1, CS0 = 100 (Dedicated baud rate generator)
- SCKE = 1 (Configure SCK pin as clock output pin)
- SOE = 1 (Configure SOT pin as UART output pin)

After defining the UART mode, it is required to configure the serial control register for data format, and enables the various flags for transmission/reception operation. This setting is done in the serial control register SCR1.
Set this register as follows:

<table>
<thead>
<tr>
<th>TXE</th>
<th>RXE</th>
<th>REC</th>
<th>A/D</th>
<th>CL</th>
<th>SBL</th>
<th>P</th>
<th>PEN</th>
</tr>
</thead>
</table>

A/D = 1 ; Address data frame
PEN = 0 ; no parity
SBL = 0 ; 1-bit stop length
TXE = 1 ; Transmit enable
RXE = 1 ; Receive enable
CL = 1 ; 8 bits data length

**Baud rate Generator**

As the clock is given by the master microcontroller, the user has to configure the baud rate of the UART. The baud rate can be generated in three different ways:

- Dedicated baud rate generator
- Baud rate by internal timer
- Baud rate by external clock

The M95040 SPI device can accept the clock up to 5 MHz. However, the maximum baud rate can be programmed depends on the microcontroller.

Note: The example, demo code (Contact Local support) is designed using dedicated baud rate generator of 125kbps. However, baud rate in the program can be changed by changing the setting in SMR1 and Communication prescaler register (CDCR1).

**Clock phase and polarity**

The clock polarity (CPOL) and clock phase (CPHA) controls the steady state value of clock when no data is transmitted. M95040 EEPROM accepts CPOL, CPHA either both 0 or 1. However, UART in microcontroller can handles only the data format CPOL = 0 and CPHA = 0. The UART timing for CPOL =0 and CPHA = 0 is shown in the diagram below:
**Reverse the Byte**

The SPI device shifts in and out data as MSB first. However, normally Fujitsu UART transfers the data in the reverse way. Therefore, it is required to set Bit direction setting Bit ‘BDS’ On serial status register ‘SSR’ to MSB first option or if this feature is not available then it is required to implement a function that rotates the LSB to MSB before sending the data in and out from UART.

The SSR register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

<table>
<thead>
<tr>
<th>PE</th>
<th>ORE</th>
<th>FRE</th>
<th>RDRF</th>
<th>TDRE</th>
<th>BDS</th>
<th>RIE</th>
<th>TIE</th>
</tr>
</thead>
</table>

**Communication Protocol**

The basic communication protocol used to communicate with EEPROM is explained below. For detail information about the protocol, please refer to M95040 SPI EEPROM data sheet.

- **write_enable**: The EEPROM contains a write enable latch. The latch must be set prior to any write or before checking the status register.
- **read_status**: This function reads the status register of EEPROM. This function can be used to check for any protection of block of memory.
- **read**: This function reads the data from EEPROM. After sending the read instruction, the address for read operation is send. Since MB90F387 is acting as a master. Therefore, EEPROM should receive the clock from MCU. This is done by sending a dummy byte 0x00 to EEPROM. This operation will generate 8 clock bits. However, this dummy byte will not be seen by the EEPROM.
- **write**: This function writes the data in EEPROM. After sending the write instruction, the address and data is send respectively.
- **mirror_byte**: This function reverse the byte. This function is necessary to change the LSB and MSB of the data/address/Instruction for the SPI device.
- **write_disable**: This subroutine disables the write enable latch of EEPROM. This disables any write or checking the write status register.
- **clear_buffer**: This buffer clears the receive buffer.
The flowchart for SPI communication using UART is shown below:

1. **Main**
2. **UART Initialization**
3. **Write_enable**
4. **Write the data from address X**
5. **Read_status**
6. **Write in progress**
7. **Increment X**
8. **End of writing?**
   - **No**
   - **End**
   - **Yes**
     - **Write_disable**
     - **Read the data from Address X in EEPROM**
     - **Increment X**
     - **End of Reading?**
       - **No**
       - **End**
       - **Yes**