AN10587

Interfacing NXP bridge IC with NXP ARM microcontroller

Rev. 01 — 23 February 2007 Application note

Document information

Info	Content
Keywords	SC16IS752, LPC2148, Bridge IC to microcontroller, serial interface
Abstract	The operation and description of NXP LPC2100 Series microcontroller such as LPC2148, ARM7-based microcontrollers to an NXP bridge IC such as SC16IS752, SPI/I^{2}C-bus to 2-channel UART/IrDA plus GPIO for high-speed serial data communication is discussed in this application note. In addition, the source code in C language, containing communication routines between the LPC2148 microcontroller and the SC16IS752 bridge IC is provided. This application note is also applicable to other bridge ICs such as SC16IS740, SC16IS750, SC16IS760 and SC16IS762.

Revision history

Rev	Date	Description
01	20070223	Application note; initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The NXP Bridge IC is a new generation of interface solutions for managing high-speed serial data communication among various bus interfaces such as SPI, I^{2}C-bus, UART including RS-232 and RS-485, IrDA, and GPIO. The Bridge IC is commonly used to overcome the limitation of the host bus interface to peripherals and provides easy to interface with existing different serial buses interface.

The description of the block diagram, hardware, firmware, and software are described in the next paragraph for users to quickly understand the implementation of the NXP LPC2100 Series microcontroller to NXP Bridge IC serial interface for RS-232 point-point communication, RS-485 multi-drop application, IrDA wireless links communication, and GPIO interface. The source code in C language for SPI-bus interface is provided to show how to write a simple communication program between the microcontroller and the Bridge IC serial interface. The goal is to help users to design the Bridge IC in their application and also shorten their product development cycle.

2. Block diagram

The block diagram depicted in Figure 1 shows the circuit connection between an NXP Bridge IC such as SC16IS752 and NXP LPC2100 Series microcontroller such as LPC2148. The Bridge IC offers simple, flexible, and minimal connection to the microcontroller. The Bridge IC embedded SPI and I^{2}C-bus for the host interface so the microcontroller can easily control the Bridge IC with few wires connection through SPI or I^{2}C-bus, which is a widely-used serial bus interface. If the I^{2}C-bus is selected, the Bridge can interface to the microcontroller with 2-wire connection. The signals on the 2 wires are SCL (Serial Clock) and SDA (Serial Data). If the SPI-bus interface is selected, the Bridge IC can interface to the microcontroller with 4-wire connection. The signals on the 4 wires shown in Figure 2 are SO (master input Slave Output), SI (master output Slave Input), $\overline{\text{CS}}$ (slave Chip Select), and SCLK (SPI Clock).

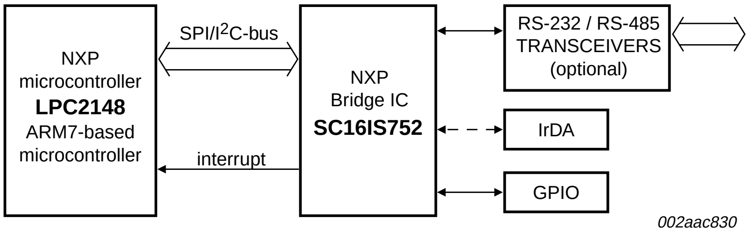

Fig 1. SC16IS752 bridge IC interface to LPC2148 microcontroller

After the microcontroller and Bridge IC are wired properly and their connection is established, the microcontroller can send data to and receive data from UART devices via the Bridge IC. The Bridge IC receives the data from the microcontroller via the SPI or I^{2}C-bus interface and sends the data to the UART devices via RS-232 or RS-485 bus interface and when the Bridge IC receives the data from the UART/IrDA/GPIO devices, the Bridge IC will notify the microcontroller by generating an interrupt signal, then the microcontroller can access the data from the Bridge IC via the SPI or I^{2}C-bus interface.

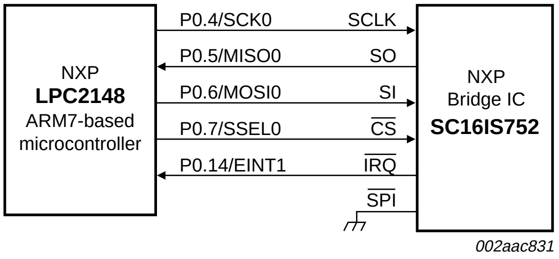

Fig 2. SPI bus interface between LPC2148 and SC16IS752

3. Hardware description

The hardware consists of the two major parts: the microcontroller and the Bridge IC. The Bridge IC provides a seamless interface convergence between the SPI or I^{2}C-bus and RS-232 or RS-485 protocols and vice versa. In addition, the Bridge IC checks the data transmission for errors and maintains very high data throughput.

The functions of the two major parts are described as follows:

- NXP LPC2148 microcontroller

  The microcontroller connection to the Bridge IC is through the SPI or I^{2}C-bus interface. The microcontroller acts as a master controlling the Bridge IC with the embedded firmware code in the microcontroller ROM (Read Only Memory). The microcontroller initiates the data transfer on the bus and controls the Bridge IC with the chip select pin for SPI-bus interface. For the SPI-bus interface, these signals (V_{CC}, $\overline{SS}$, MISO, MOSI, SCLK, GND, $\overline{IRQ}$) have to be connected between the microcontroller and the Bridge IC.

- NXP SC16IS752 Bridge IC

  The Bridge IC acts as a slave interface convergence between the microcontroller and UART peripherals. The Bridge IC handles the SPI or I^{2}C-bus interface to the microcontroller and the UART that consists of transmitter and receiver. The transmitter sends the data received from the microcontroller to the UART peripherals. The receiver sends the data received from the UART peripherals to the microcontroller. In addition to the UART peripherals, the microcontroller can control IrDA and GPIO peripherals.

The Transceivers are optional and can be intended for RS-232 or RS-485 communication interface. The Transceivers consist of drivers and receivers. The drivers convert the UART-logic output levels to RS-232 or RS-485 signals, whereas the receivers convert the RS-232 or RS-485 signals to UART-logic input levels. If the UART-logic levels conversion is not required, the Transceivers are optional.

4. Software description

The programming of the Bridge IC can be done by writing firmware code, which requires the following software:

- Keil is one of the embedded system vendors that provide the software development tools for the NXP LPC2148 microcontroller. The software compiles the firmware code written in ‘C’ and generates an ‘Intel Hex’ file. The software evaluation development kit can be downloaded from the Keil web site.
- LPC2000 Flash Utility is a free Windows application software that allows easy programming of the NXP LPC2148 microcontroller. The software loads the ‘Intel Hex’ file to the microcontroller by using its in-system programming mode communicating through the serial port. The software can be downloaded from http://www.nxp.com/products/microcontrollers/support/software_download/lpc2000/.

5. In-System Programming mode

NXP LPC2148 microcontroller has an on-chip Flash program memory with In-System Programming (ISP), which allows the microcontroller to be programmed without removing the microcontroller from the board and also the microcontroller, which previously programmed can be reprogrammed without removal from the board.

The microcontroller must be powered-up in a special ‘ISP mode’ to perform the ISP operation. The ISP mode allows the microcontroller to communicate with a host device such as PC through a serial port. The host sends commands and data to the microcontroller. The commands can be erase, read, and write. After the completion of the ISP operation, the microcontroller is reconfigured and has to be reset or power cycled so the microcontroller will operate normally.

The ISP programming for the device can be done using a Windows application software, which uses an Intel Hex file as input to program it. For more information about the software, please refer to Section 4 “Software description”.

6. Firmware description

The firmware code for the LPC2148 microcontroller is written in C language for SPI-bus interface. It can be compiled by using an embedded C compiler. For more information about the compiler, please refer to Section 4 “Software description”.

The firmware code consists of three major blocks, which are Main Loop, Interrupt Service Routine, and Bus Interface layer described as follows.

6.1 Main Loop

The function of the Main Loop is to initialize the SPI port of the microcontroller (see Table 1) and also to initialize the Bridge IC by writing a byte to the Bridge IC registers (see Table 2). Inside the Main Loop, the microcontroller can select one of the two methods for communicating to the Bridge IC. The first method is polling the Bridge IC status register regularly. The second method is using an interrupt handler in the interrupt service routine until the Bridge IC generates an interrupt. If using the interrupt handler, the interrupt pins of the Bridge IC must be connected to the microcontroller and the interrupt bits register must be enabled. The other function of the Main Loop is to keep checking the event flags and pass to the appropriate subroutine for further processing.

Table 1. Initialize SPI port of the microcontroller

```
Void init_mcu_port (void)
{
  // SPI port: P0.4=SCK0, P0.5=MISO0, P0.6=MOSI0, P0.7=SSEL0
  PINSEL0 |= 0x20001500; // set SCK0, MISO0, SSEL0, EINT1
       // Bit 29:28 = p0.14  => 10=EINT1
       // Bit 15:8  = p0.4-7 => SPI00 (SCLK, MISO, MOSI)

  VICIntSelect  |= 0x00008000; // enable a VIC Channel as FIQ
  VICIntEnable  |= 0x00008000; // enable EINT1
       // Bit 17-14 = EINT3-0

  VICDefVectAddr = (unsigned long) DefaultIRQ; // set Default interrupt vector

  // GPIO direction
  IO0DIR  |= 0x000000D0;  // set SCK0, MOSI0, SSEL0 as output and MISO0 as input
       // Bit 29:28 = p0.14 (set EINT1 as input)
       // Bit 15:8 = p0.7-4 (SSEL0, MOS10, MISO0, SCK0)

  // GPIO setting
  IO0SET  |= 0x00000080; // set p0.7 (SSEL0) to high (disable SPI slave chip select)

  // Control the rate of the APB clock in relation to the processor clock
  VPBDIV = 0x1;  // set APB clock to same as processor clock

  // SPI Clock Counter Register to control the frequency of master’s clock
  S0SPCCR = 0x6;  // set SCLK (SPI clock)

  // SPI Control Register to control the SPI operation
  S0SPCR = 0x20; // set SPI mode 0 (CPHA=0, CPOL=0), master mode, MSB first
                 // set SPI interrupt enabled, 8 bits of SPI data per transfer
}
```

Table 2. Initialize SC16IS752 registers

```
Void SC16IS752_Init_ChA (void) // program channel A for SPI-UART
{     // set 115200 baud, 8N1
      SPI_wr_752 (LCR, 0x80, 0);         // 0x80 to program baud rate
      SPI_wr_752 (DLL, 0x08, 0);         // 0x08 = 115.2K with X1 = 14.7456 MHz
      SPI_wr_752 (DLM, 0x00, 0);         // divisor = 0x0008 for 115200 bps
      SPI_wr_752 (LCR, 0xBF, 0);         // access EFR register
      SPI_wr_752 (EFR, 0X10, 0);         // enable enhanced registers
      SPI_wr_752 (LCR, 0x03, 0);         // 8 data bit, 1 stop bit, no parity
      SPI_wr_752 (FCR, 0x01, 0);         // enable FIFO mode
      SPI_wr_752 (SPR, 'A', 0);          // scratch pad = character A (0x41)
      SPI_wr_752 (IODIR, 0xFF, 0);       // set GPIO [7:0] to output
                                         // (default: 0x00=input)
      SPI_wr_752 (IOSTATE, 0x00, 0);     // set GPIO [7:0] to 0x00 (LEDs on)
      SPI_wr_752 (IER, 0x01, 0);         // enable Rx data ready interrupt
}

Void SC16IS752_Init_ChB (void)  // program channel B for SPI-IrDA
{
      SPI_wr_752 (LCR, 0x80, 2);     // 0x80 to access program baud rate
      SPI_wr_752 (DLL, 0x80, 2);     // set IRDA to 2400 bps divider 0x0180
      SPI_wr_752 (DLM, 0x01, 2);     // program baud rate high byte
      SPI_wr_752 (LCR, 0XBF, 2);     // access EFR
      SPI_wr_752 (EFR, 0X10, 2);     // enable enhanced registers
      SPI_wr_752 (LCR, 0x03, 2);     // 8 data bit, 1 stop bit, no parity
      SPI_wr_752 (FCR, 0x01, 2);     // enable FIFO mode
      SPI_wr_752 (SPR, 'B', 2);      // scratch pad = character B (0x42)
      SPI_wr_752 (EFCR, 0x00, 2);    // IrDA SIR 115.2 Kbps
      SPI_wr_752 (MCR, 0x40, 2);     // enable IRDA mode
      SPI_wr_752 (IER, 0x01, 2);     // enable receive interrupt
}
```

6.2 Interrupt Service Routine (ISR)

The microcontroller uses the Interrupt Service Routine (ISR) to handle an interrupt generated by the Bridge IC. As soon as the Bridge IC generates an interrupt, the ISR checks the interrupt status of the Bridge IC to determine the interrupt sources and sets up proper event flags to inform the Main Loop program for processing the interrupt request.

Table 3 shows the interrupt handler read the interrupt identity register to check the interrupt sources such as receiver interrupt. If the receiver interrupt is detected, the interrupt handler will read data from the receiver FIFO of the Bridge IC and store the data.

Table 3. Interrupt Service Routine (ISR)

```
Void FIQ_Handler (void) __fiq
{  // Interrupt Service Routine (ISR)
     char ch;

     // check interrupt channel B for SPI-IrDA
     ch = SPI_rd_752 (IIR, 2);  // read interrupt identity register
     if (ch & 0x04) {  // if receiver (RHR) interrupt
            ch = SPI_rd_752 (SC16IS_RHR, 2);   // read from RxFIFO chB
            SPI_wr_752 (SC16IS_THR, ch, 0);    // write to TxFIFO chA
     } // end data ready

     EXTINT = 0x00000002;   // clear the peripheral interrupt flag
            // Bit 3-0 = EINT3-0
}
```

6.3 Bus Interface layer

The Bus Interface layer handles the SPI bus interface between the microcontroller and the Bridge IC. The two functions in the bus interface layer are ‘Bus Interface Read’ and ‘Bus Interface Write’.

6.3.1 Bus Interface Read

The microcontroller reads data from the SPI bus and stores the data for further processing.

Table 4. Bus Interface Read

```
BYTE SPI_rd_752 (BYTE reg, BYTE channel)  // mcu read from SC16IS752
{
     BYTE dataRead;
     reg = (reg<<3) | 0x80;             // register address byte
     if (channel==0x02) reg |= channel; // channel address byte
     SPICommand[0] = reg;               // register address
     IO0CLR |= 0x00000080;              // P0.7 goes low; enable slave chip select
     SPISend(SPICommand, 1);            // 1-byte address is sent
     SPIReceive(SPIDataRead, 1);        // 1-byte data is read
     dataRead = SPIDataRead[0];         // store the read data
     IO0SET |= 0x00000080;              // P0.7 goes high; disable slave chip select
     return dataRead;
}
```

6.3.2 Bus Interface Write

The microcontroller writes data to the SPI bus for the Bridge IC to read.

Table 5. Bus Interface Write

```
Void SPI_wr_752 (BYTE reg, BYTE DataByte, BYTE channel)  // mcu writes to SC16IS752
{
    reg <<= 3;                         // register address byte
    if (channel==0x02) reg |= channel; // channel address byte
    SPIDataWrite[0] = reg;             // register address
    SPIDataWrite[1] = DataByte;        // register data
    IO0CLR |= 0x00000080;              // P0.7/SSEL0=low; enable slave chip select
    SPISend(SPIDataWrite, 2);          // 2-byte data is sent
    IO0SET |= 0x00000080;              // P0.7/SSEL0=high; disable slave chip select
}
```

7. Conclusion

The NXP Bridge IC provides easy interface to a host controller such as the NXP LPC2148 microcontroller, enables seamless and high-speed SPI or I^{2}C-bus to RS-232 or RS-485 protocols convergence including GPIO for general-purpose input/output and IrDA for wireless links, and offers low voltage operation, low power consumption, and compact design which is suitable for battery-operated applications. In addition, the Bridge IC reduces software overhead, frees up the host controller’s resources, increases design flexibility, and improves overall system performance. For more details about the Bridge ICs, please visit our web site at http://www.NXP.com/interface to download the data sheets.

8. Abbreviations

Table 6. Abbreviations

Acronym	Description
FIFO	First In, First Out
GPIO	General Purpose Input/Output
I^{2}C-bus	Inter-Integrated Circuit bus
IC	Integrated Circuit
IrDA	Infrared Data Association
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter

9. Legal information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

9.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^{2}C-bus — logo is a trademark of NXP B.V.

10. Contents

1 Introduction 3
2 Block diagram 3
3 Hardware description 4
4 Software description 5
5 In-System Programming mode. 5
6 Firmware description 5
6.1 Main Loop. 5
6.2 Interrupt Service Routine (ISR). 8
6.3 Bus Interface layer 8
6.3.1 Bus Interface Read. 8
6.3.2 Bus Interface Write 9
7 Conclusion 9
8 Abbreviations. 9
9 Legal information. 10
9.1 Definitions. 10
9.2 Disclaimers 10
9.3 Trademarks. 10
10 Contents 11

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section ‘Legal information’.

© NXP B.V. 2007. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 23 February 2007
Document identifier: AN10587_1

founded by
PHILIPS