APPLICATION NOTE

ABSTRACT
This application note describes how a SC16C650 or SC16C650B can be used to implement an IrDA interface. This application note is also applicable to SC16C654B, SC16C654DB, and SC16C652B.

AN10219
Using SC16C650/SC16C650B to implement an IrDA interface

Author: Dong Nguyen
2004 Jul 12
Supersedes data of 2003 Jul 25

Philips Semiconductors

PHILIPS

Using SC16C650/SC16C650B to implement an IrDA interface

AN10219

INTRODUCTION

This application note describes how a SC16C650 can be used to implement an IrDA interface. There are two sections to this application note – hardware and software. The hardware section shows how the SC16C650 is electrically hooked-up to an IrDA transceiver module; the software section shows how the code can be written to configure the SC16C650 in IrDA mode (115.2 kbit/s max.).

HARDWARE INTERFACE

The electrical connection between the SC16C650 and an IrDA transceiver module is very straightforward, no external component is required except an inverter to invert the IrDA output signal. Figure 1 shows this connection between the SC16C650 and an IrDA transceiver module made by Vishay Semiconductors. Besides the power supply noise-filtering components, the only other component required is a 14 Ω resistor (this value is recommended by the manufacturer). This resistor sets the current through the IR emitter, hence the power output of the transceiver (please consult your transceiver module manufacturer datasheet for other values).

SOFTWARE CODING

Codes to be written to initialize the SC16C650 to operate in IrDA mode can be divided into two sections:

1. codes to set the communication parameters: baud rates, word length, parity and stop bits.
2. codes to turn on IrDA mode.

The following codes show, as an example, how the above two sections can be written:

```
/* This program configures the UART to operate in IrDA mode. It sits in a loop
waiting for a character to be received by polling the receive ready bit of the LSR
register, if the bit is set it will read the character and display it on the screen.
Otherwise it will poll the keyboard to see if there is a key struck, if there is
will send this key character out */

#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define PORT 0x2e8
  /* Defines Serial Ports Base Address */
  /* COM1 0x3F8                        */
  /* COM2 0x2F8                        */
  /* COM3 0x3E8                        */
  /* COM4 0x2E8                        */

   /*register addresses*/
#define RHR       PORT+0   /* R     ; Receive Hold Reg      */
#define THR       PORT+0   /* W     ; Transmit Hold Reg     */
#define IER       PORT+1   /* R/W   ; Interrupt enable reg  */
#define FCR       PORT+2   /* W     ; FIFO Control reg      */
#define IIR       PORT+2   /* R     ; Interrupt ident. reg  */
#define LCR       PORT+3   /* R/W   ; Line Control Reg      */
#define MCR       PORT+4   /* R/W   ; Modem Control Reg     */
#define LSR       PORT+5   /* R     ; Line Status Reg       */
#define MSR       PORT+6   /* R     ; Modem Status Reg      */
#define SPR       PORT+7   /* R/W   ; Scratch reg           */
```

```
   /*ext reg addr accessible only when LCR bit–7 is 1 */
#define DLL       PORT+0      /* R/W      ; Divisor latch LSB*/
#define DLH       PORT+1      /* R/W      ; ,,      ,,  MSB */

   /*ext reg addr accessible only when LCR is set to 0xBF */
#define EFR       PORT+2      /* R/W; Enhanced Feature Reg  */
#define XON1      PORT+4      /* R/W                        */
#define XON2      PORT+5      /* R/W                        */
#define XOFF1     PORT+6      /* R/W                        */
#define XOFF2     PORT+7      /* R/W                        */

int read(int);          /*read from register in UART*/
void write(int,int);    /*write to register in UART*/

void main(void)
{
 int c;
 unsigned int c, ch;
 int key;
 unsigned int n;

 write(MCR, 0x10);      /* put UART in loopback mode  */
 while(read(LSR)&0x01)  /* clear RX FIFO              */
      read(RHR);

 write(IER, 0x00);   /* Turn off interrupts – Port1 */

 /*         PORT 1 – Communication Settings         */

 write(LCR, 0x80);  /* SET DLAB ON */
 write(DLL, 0x02);  /* Set Baud rate – Divisor Latch Low Byte */
                    /* Default 0x03 =  38,400 BPS */
                    /*         0x01 = 115,200 BPS */
                    /*         0x02 =  57,600 BPS */
                    /*         0x06 =  19,200 BPS */
                    /*         0x0C =   9,600 BPS */
                    /*         0x18 =   4,800 BPS */
                    /*         0x30 =   2,400 BPS */
 write(DLH, 0x00);  /* Set Baud rate – Divisor Latch High Byte */
 write(LCR,0xBF);   /* To access EFR reg. LCR must be set to BF */
 write(EFR, 0x10);  /* enable enhanced features: IrDA*/
 write(LCR, 0x03);  /* 8 Bits, No Parity, 1 Stop Bit */
 write(FCR, 0x0f);  /* FIFO reg.: DMA mode 1, clear TX and RX fifo */
 write(MCR, 0x4b);  /* turn on IrDA */

 do { c = read(LSR);       /* Check to see if char has been received */
      if (c & 1) {ch = read(RHR);   /* If so, then get Char          */
              printf("%c",ch);}     /* Print Char to Screen          */

      if (kbhit()){ch = getch();    /* If key pressed, get Char */
               write(THR, ch);}     /* Send Char to Serial Port */
    } while (ch !=27);  /* Quit program when ESC(ASC 27) is pressed */
```

```
 }

int read(int regist)
      {
      int rd;
      rd=inportb(regist);
      return (rd);
      }
void write(int regist, int code)
      {
      outportb(regist,code);
      }
```

Using SC16C650/SC16C650B to implement an IrDA interface

AN10219

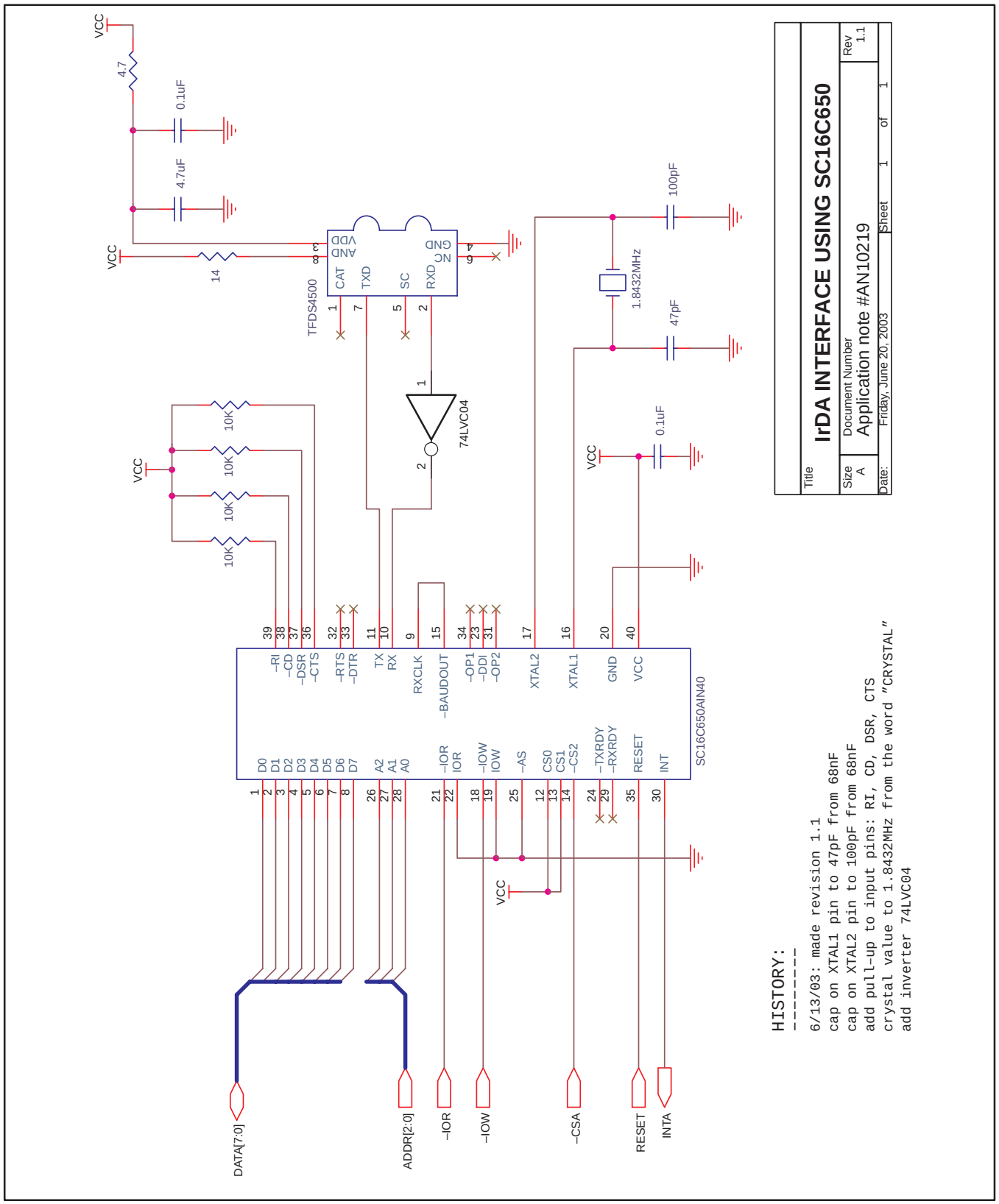

Figure 1.

Using SC16C650/SC16C650B to implement an IrDA interface

AN10219

REVISION HISTORY

Rev	Date	Description
_3	20040712	**Application note; third version (9397 750 13516). Supersedes data of 25 July 2003 (939775011631).** Modifications: ● Added SC16C650B to title. ● 'Abstract' re-written to reference additional part-types.
_2	20030725	**Application note; second version (9397 750 11631). Supersedes data of 24 April 2003 (939775011423)**
_1	20030424	**Application note; initial version (9397 750 11423).**

Contact information

For additional information please visit
http://www.semiconductors.philips.com. Fax: **+31 40 27 24825**

For sales offices addresses send e-mail to:
sales.addresses@www.semiconductors.philips.com.

Date of release: 07-04

Document order number: 9397 750 13516

Let's make things better.

Philips Semiconductors

PHILIPS