DESIGN NOTE

#048

AUTHOR: THOMAS DAIS

KEYWORDS: I²C

I^{2}C Bus Sniffer

This little project has been designed for debugging purposes. It can be used to translate data transfered from the I^{2}C protocol (specs ver. 1 which means up to 100 KHz clock rate) to RS-232 asynchronous protocol(at 115200).

AVR AT90S2313-10 microcontroller's External Interrupt pins are used to determine the falling or rising edges of the SDA line (Start or Stop condition) and are also checking to determine every bit of the transferred data after the rising edge of the SCK line.

These lines are not connected directly to the controller pins, but through:

1. A resistor for protection,
2. A diode and an AND (buffer) gate with pull-up resistor on its input. When the I^{2}C line has "1" (3V or 5V) the level at the port pin is high because of the pull-up resistor, and when the I^{2}C line has a "0" the voltage in the AND gate's input is about 0.6 Volts so the level at the controllers port pin is low.

The switches can be used to monitor certain I^{2}C address and the LEDs to show that there is a signal on the SCK or SDA.

Here is a sample program written in Codevision C compiler IDE:

def.h File

```
#ifndef Def_H
  #define Def_H

  typedef    char             BYTE;
  typedef    int              WORD;

  #define    TRUE             1
  #define    FALSE            0

  #define    I2C_STATE_IDLE   0
  #define    I2C_STATE_START  1
  #define    I2C_STATE_ACK    2
  #define    I2C_STATE_STOP   3

  #define    SDA              PIND.2
  #define    SCK              PIND.3
```

```
#endif
```

main.h File

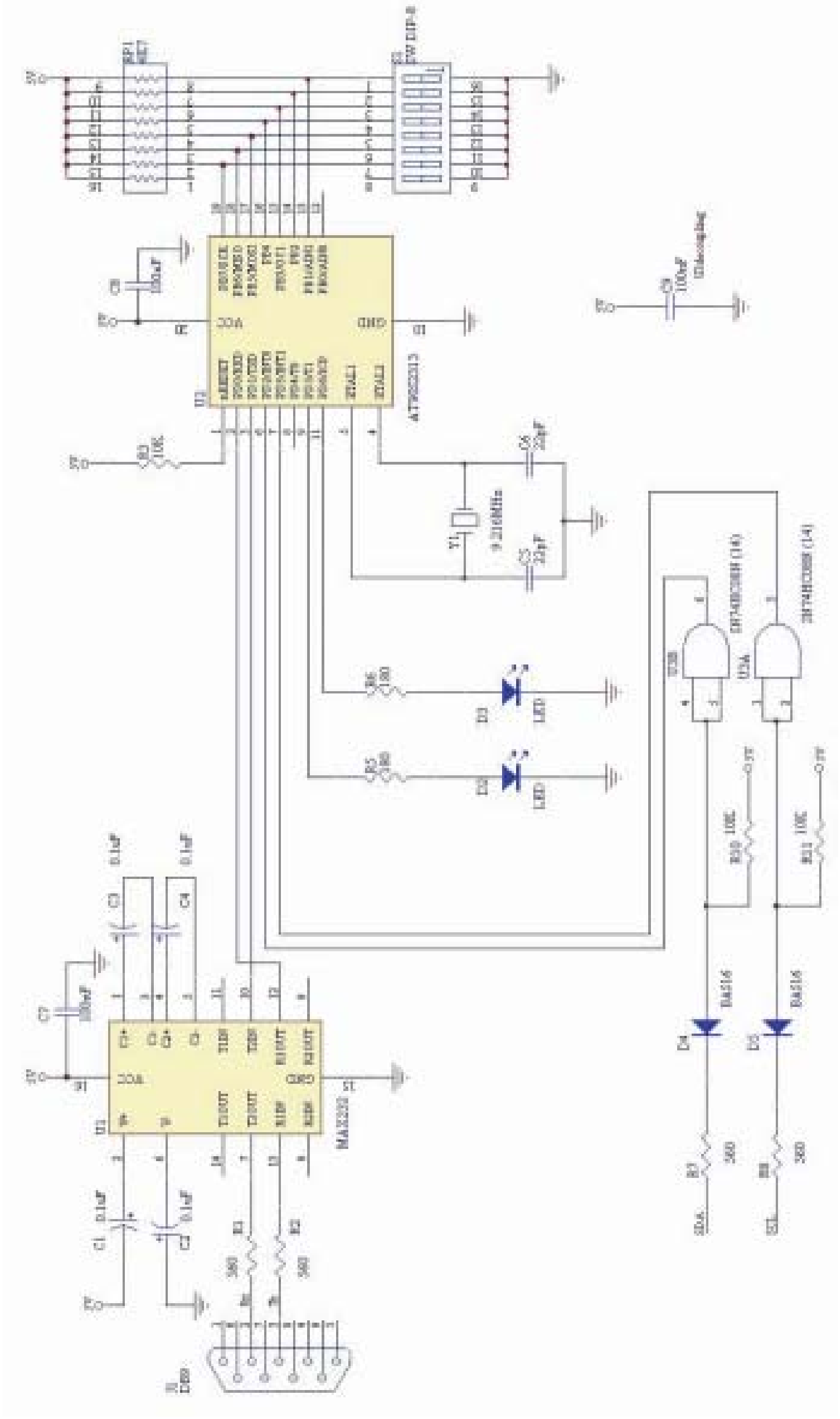
```
#ifndef Main_H
  #define Main_H
  #include "def.h"

  BYTE       I2CState;
  BYTE       BitCounter;
  BYTE       I2CByte;
  bit        I2CBit0;
  bit        I2CBit1;
  bit        I2CBit2;
  bit        I2CBit3;
  bit        I2CBit4;
  bit        I2CBit5;
  bit        I2CBit6;
  bit        I2CBit7;
  bit        I2CAck;
  bit        ByteEnd;
  bit        RdBit;
  #endif
```

main.c File

```
#include <90s2313.h>
#include "main.h"
#pragma warn- // this will prevent warnings

char I2cRead(void) {
#asm
    mov     r30,r2
#endasm
}

#pragma warn+                                  // enable warnings

interrupt [EXT_INT0] void ext_int0_isr(void)
{
  if (SCK == 1) {
      switch ( I2CState ) {
        case I2C_STATE_IDLE:
           I2CState = I2C_STATE_START;         // start condition
           MCUCR |= 0x03;                      // INT0 Rising edge
           ByteEnd = FALSE;
           break;
        case   I2C_STATE_ACK:
           I2CState = I2C_STATE_STOP;
           MCUCR &= 0xFE;                      // INT0 Falling edge
           break;
      }
    }
}

interrupt [EXT_INT1] void ext_int1_isr(void)
{
  RdBit = SDA;
  switch (BitCounter) {
      case 8:
          I2CBit7 = RdBit;
          //ByteTimoutF = TRUE;
          break;
      case 7:
          I2CBit6 = RdBit;
          break;
      case 6:
          I2CBit5 = RdBit;
          break;
      case 5:
          I2CBit4 = RdBit;
          break;
      case 4:
          I2CBit3 = RdBit;
          break;
      case 3:
          I2CBit2 = RdBit;
          break;
      case 2:
          I2CBit1 = RdBit;
          break;
      case 1:
          I2CBit0 = RdBit;
          break;
      case 0:
          I2CAck = RdBit;
          I2CState = I2C_STATE_ACK;
          BitCounter = 9;
          ByteEnd = TRUE;
  }
  BitCounter -- ;
  if (ByteEnd) {
      ByteEnd = FALSE;
      I2CByte = I2cRead();
      UDR = I2CByte;
  }

}
```

```
#include <stdio.h>

void InitVars(void) {
  BitCounter = 8;
  I2CState =  I2C_STATE_IDLE;
}

void main(void)
{
PORTB=0x00;
DDRB=0x01;

PORTD=0x00;
DDRD=0x62;

TCCR0=0x06;
TCNT0=0xFF;

TCCR1A=0x00;
TCCR1B=0x05;
TCNT1H=0xFF;
TCNT1L=0xD3;
OCR1H=0x00;
OCR1L=0x00;

GIMSK=0xC0;
MCUCR=0x0E;
GIFR=0xC0;

TIMSK=0x82;

UCR=0x08;
UBRR=0x04;

ACSR=0x80;

#asm("sei")

  while (1) {
        InitVars();
        while(1){
            if (USR.6 == 1){
            USR.6 = 1;
            }
            if ( I2CState == I2C_STATE_STOP) {
            I2CState = I2C_STATE_IDLE;
            }
        }
  }
}
```

Figure 1. Schematics



AT90S2313
U2
VCC
GND
RESET
XTAL1
XTAL2
Y1
9.216MHz
C5
22pF
C6
22pF
C8
100nF
R3
10K
RP1
10K
S1
SW DIP-8
R5
180
R6
180
D2
D3
LED
C9
100nF
U2 decoupling
U3A
U3B
SN74HC08N (14)
R10 10K
R11 10K
D4
D5
BAS16
R7
R8
560
SDA
SCL
MAX232
U1
C1+
C1-
C2+
C2-
V+
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
C1
C2
C3
C4
0.1uF
C7
100nF
R1
R2
560
J1
DB9