ELEKTRONIK TECHNISCHE INFORMATIK HTL Hollabrunn
ENTSCHEIDE DICH FÜR DIE ZUKUNFT

HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

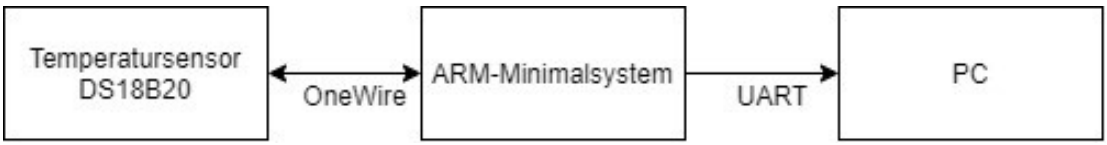
Klasse / Jahrgang: 5BHEL	Gruppe:	Übungsleiter: Prof. Reisinger
Übungsnummer: 3	Übungstitel: CM3 – OneWire Temperatursensor	
Datum der Übung:	Teilnehmer: Pruggmayer, Mottl	
Datum der Abgabe: 09.04.2021	Schriftführer:	Unterschrift:

Inhaltsverzeichnis

1 Aufgabenstellung

Es soll ein Programm geschrieben werden, mit dem man den OneWire Temperatursensor auf dem ARM-Minimalsystem auslesen kann. Darüber hinaus soll der UART1 Interrupt gesteuert verwendet werden.

2 Blockschaltbild

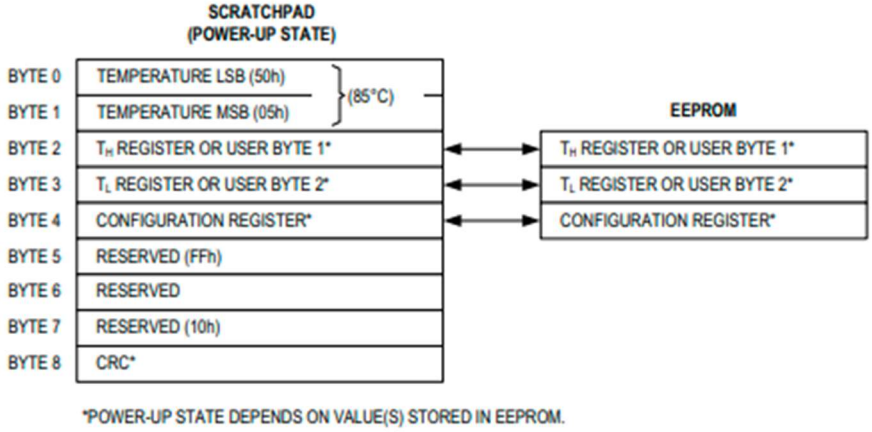


3 DS18B20 Temperatursensor

Der DS18B20 ist ein digitales Thermometer das 9-12-bit Temperaturmesswerte in Grad Celsius liefert. Dieser hat auch eine Alarm Funktion, dessen Schwellwerte vom Benutzer eingestellt werden können. Sein messbarer Temperaturbereich liegt zwischen -55C bis 125C.

4 Speicherverwaltung DS18B20

Um Temperaturwerte zu bekommen muss man das 9Byte große Scratchpad auslesen. Jedoch sind nur die ersten beiden Bytes wichtig. Für die Temperaturmessung.



5 Aufbau der Daten DS18B20

Da die Temperatur in 2Byte aufgeteilt ist sieht die aufteilung der Daten wie folgt aus.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

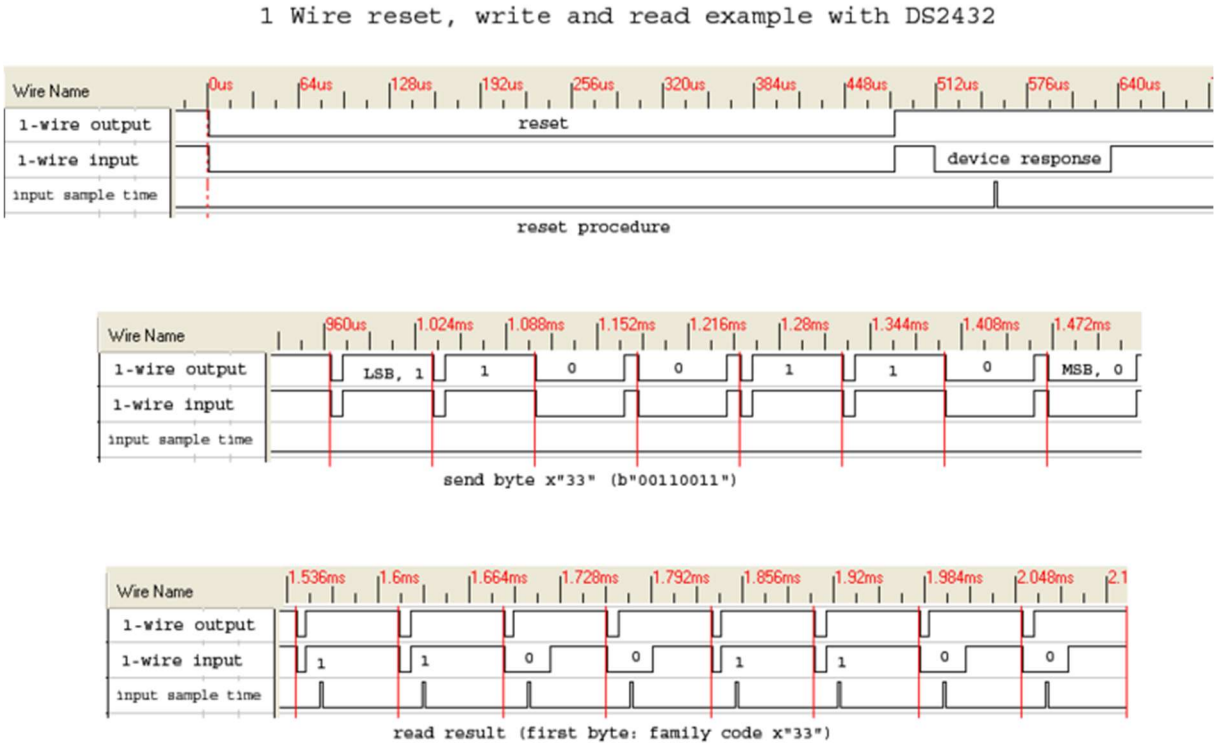
S = SIGN

Der Sensor verwendet 5 Sign bits die Anzeigen, ob der Temperaturwert positiv oder negativ ist.

6 OneWire – Protokoll

1-Wire bzw. **One-Wire** oder **Eindraht-Bus** beschreibt eine serielle Schnittstelle der Firma Dallas Semiconductor Corp., die mit einer Datenader (DQ) auskommt, die sowohl als Stromversorgung als auch als Sende- und Empfangsleitung genutzt wird. Der Begriff 1-Wire ist irreführend, weil zudem noch eine Masse-Verbindung (GND) erforderlich ist. Diese Masseverbindung wird bei Knopf-förmigen Anordnungen über eine gegenseitige Isolation der Gehäusehälften erreicht. Tatsächlich werden immer zwei physische Leiterverbindungen benutzt (GND, DQ).

Ein High-bit besteht aus 10us Low und 60us High, Low-bit besteht aus 60us Low und 10us High.



7 OneWire Implementation

Um das Protokoll anforderungsgemäß zu implementieren wurden drei wichtige Wartefunktionen geschrieben. Für zurücksetzen des Buses, senden von Highbite & Lowbite.
Diese Wartefunktionen wurden einmal mit normalen For-Schleifen implementiert, andererseits mit einen Timer der alle 10us ein Interrupt auslöst.

Die Anzahl der Schleifendurchläufe wurden ausgerechnet mithilfe der maximalen Frequenz und der Cycles pro Durchlauf.

```
void wait_10us(void)        //waits 10 us
{
    uint8_t i, j;
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 10; j++)
        {
            __NOP();
        }
    }
}
```

8 OneWire resetBus

Sobald der OneWire Bus reseted wird gibt es eine Rückmeldung des Sensors.
Diese muss eingelesen werden, um die Funktionalität des Sensors sicherzustellen.

```
uint8_t DS18B20::resetBus (void)
{
    uint8_t DevResponse = 0;

    DATA_OUT = 0;               //RESET Flanke as in One-Wire protocol
    wait_480us();

    DATA_OUT = 1;
    wait_60us();

    DevResponse = DATA_IN^0x01;

    wait_480us();               //wating for presence condition

    return DevResponse; //true or false
}
```

9 OneWire sendBit

Je nachdem ob das Bit 1 oder 0 werden die verschiedenen Werte mit den dazugehörigen Zeiten auf den Bus angelegt.

```
void DS18B20::sendBit(uint8_t bit)
{
        if(bit == 1)
    {
        DATA_OUT = 0;   //Write '1' bit
        wait_10us();
        DATA_OUT = 1;   //Releases the bus
        wait_60us();    //Complete the time slot and 10us recovery
    }
    else
    {
        DATA_OUT = 0;   //Write '0' bit
        wait_60us();
        DATA_OUT = 1; //Releases the bus
        wait_10us();
    }
}
```

10 OneWire sendByte

Um ein ganzes Byte zu senden wird die sendBit Funktion 8-mal aufgerufen. Und die zusendenden Daten werden Rightshifted, um immer das nächste Bit an erster Stelle zu haben.

```
void DS18B20::sendByte(uint8_t data)
{
    for (uint8_t i = 0; i < 8; i++)
    {
        sendBit(data & 0x01); //Send LSB
        data >>= 1;           //Shift right for next Bit
    }
}
```

11 OneWire readBit

Um ein Bit vom Bus zu lesen wird zuerst der Bus auf 0 gesetzt damit dem Sensor signalisiert wird, dass dieser Senden kann.
Es müssen die genauen Zeitslots für jede Übertragung eingehalten werden.

```
//Read a bit from the 1-Wire bus and return it. Provide 10us recovery time.
uint8_t DS18B20::readBit(void)
{
    uint8_t value = 0;

    DATA_OUT = 0;
    wait_10us();
    DATA_OUT = 1;                       //Releases the bus
    wait_10us();
    //value = DATA_IN & 0x01;   //Sample the bit value from the slave
    value = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_0);
    wait_60us();                        //Complete the time slot and 10us reco
very

    return value;
}
```

12 OneWire readByte

Um alle 8 Bytes zu lesen die der Sensor sendet, muss die readBit funktion 64 mal aufgerufen werden.

```
// Read 1-Wire data byte and return it
uint64_t DS18B20::readBytes(void)
{
    uint64_t idByte = 0;

    // read LSB to MSB
    for (uint8_t i = 0; i < 64; i++)
    {
        idByte >>= 1;       //shift the result to get it ready for the next bi
t
        if (readBit())
        {
            //if result is one, then set MS bit
            idByte |= 0x8000000000000000;
        }
    }
    return idByte;
}
```

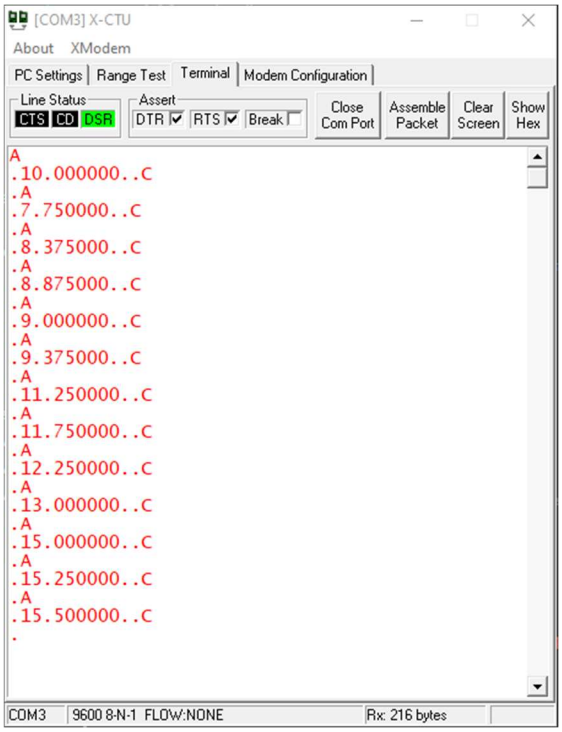
13 Interpretierung der Sensordaten

Um die Sensordaten richtig Ausgeben zu können. Müssen die sign bits ausmaskiert werden. Darüber hinaus muss die byte order behoben werden, um richtig dargestellt zu werden.

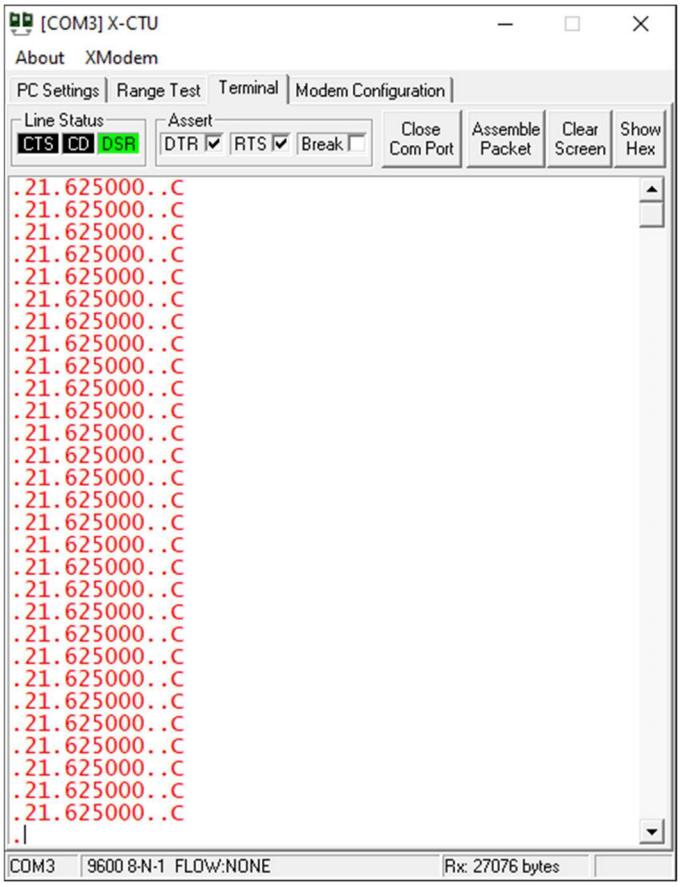
```
float DS18B20::interpret(uint64_t input)
{
    uint16_t buf, in;
    float result = 0;
    in = 0; // endianess (byte order) problem
    in += input & 0x00ff >> 0;
    in += input & 0xff00 >> 8;
    buf = in & 0x07ff; // mask out sign bits
    result = buf / (16.0); // divide to let LSb represent 2^-4
    result *= ( (in & 0xf800) != 0) ? -1 : 1; // interpret sign bit
    return result;
}
```

14 Funktionsbeweis

Um die Funktionalität des Sensors zu beweisen wurde das ARM-Minimalsystem für 20minuten in den Kühlschrank gelegt. Danach wurde es auf Raumtemperatur erwärmt.



Hier sieht man die Temperatur direkt nach dem Kühlschrank und wie sie sich langsam erwärmt.



Das System wurde auf Langzeit (5h) betrieben. Um die Raumtemperatur zu messen. Die Abweichung betrug nur ungefähr 1,5 Grad von der tatsächlichen Raumtemperatur (zweites Thermometer im Zimmer)