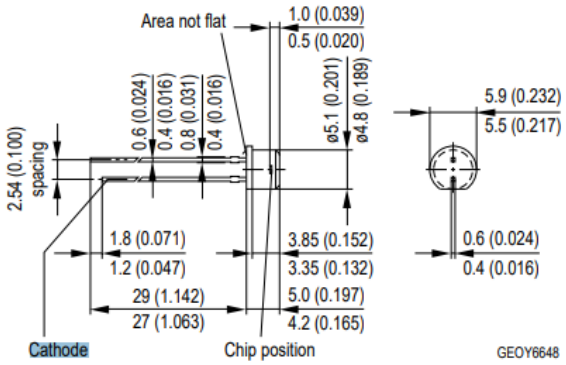
MPS Praktikum 5

Trung Thieu Quang – 771043

1. Die Port-Belegung ist funktionsfähig und sollte nicht verändert werden
    a. Funktion der Photodiode: Licht in einer elektrische Spannung oder einen elektrischen Strom umzusetzen oder mit Licht übertragene Informationen zu empfangen

       TIA: wandelt ein analoges Eingangssignal in einen Strom um und vergleicht diesen Strom mit einem Referenzstrom. Basierend auf dem Vergleich wird ein digitales Signal erzeugt

       Man kann mithilfe der Länge der Pins einer Photodiode die Cathode und Anode ermitteln




**Features:**

- Wavelength range (S10%) 400 nm to 1100 nm
- Short switching time (typ. 5 ns)
- 5 mm LED plastic package

Ja diese Photodiode kann auch Licht der Wellenlängen 400-700nm wahrnehmen

    b. ADC: benötigte Funktionen:

```
// Photo Diode ADC Initialization
TIA_PD_Start();             // Start TIA for photo diode
ADC_SAR_PD_Start();         // Start ADC
// +++ TODO +++
// !!! ADC Konversion starten, Befehl? !!!
// TODO!  // Start ADC conversion
ADC_SAR_PD_StartConvert(); // Start Conversion     ?????

/* !!! ADC-Wert in sarResult auslesen !!! */
 sarResult = ADC_SAR_PD_GetResult16();    //TODO!
/* !!! ADC-Wert anzeigen !!! */
sprintf( buffer, "ADC: %d\n\r", sarResult );
UART_PutString( buffer );
}
```

    c. …
    d. Grundfunktion des I2C PCF8574:
        - allows microcontroller or other I2C master device to remotely control and read the state of 8 digital input/output lines over a i2c

bus. It functions as a slave device on the i2c bus, responding to commands and data requests sent by master device

- This type of I/O expander typically includes an 8 bit input output port, which can be configured as either input or outputs. It also has an interrupt pun that can be used to signal the microcontroller when the state of one or more of the I/O lines has changed. This allows the microcontroller to respond to events or changes on the I/O lines without continuously polling the state of the lines

e. Basisadresse der PCF8574: 0x20

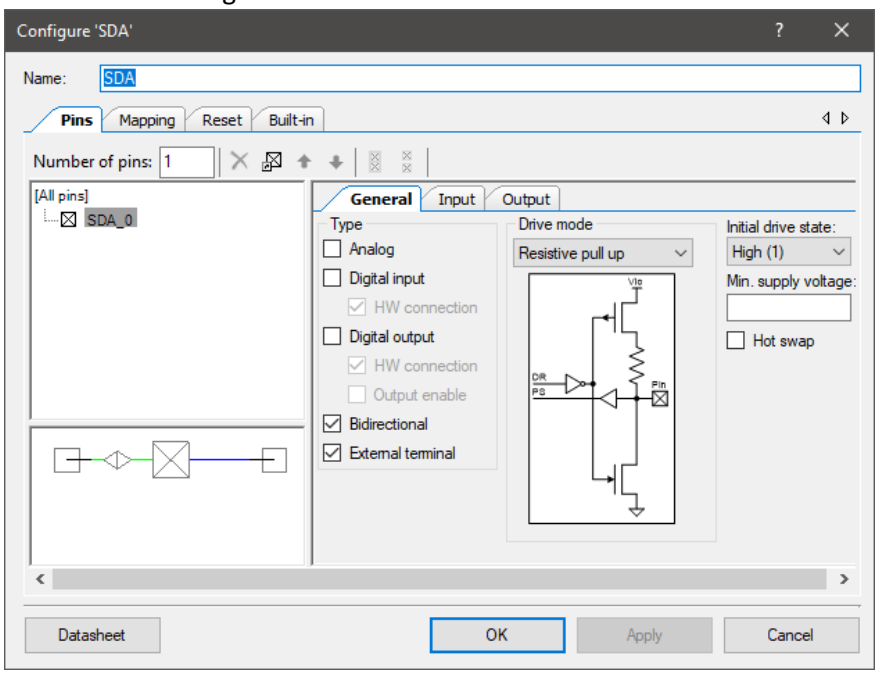
```
/* +++ settings for PCF8574 Philips/NXP) 7(!)-bit address +++ */
#define I2C_PCF8574_BASE_ADDR (0x20)     //!< TODO!, ÄNDERN, FUNKTIONIERT NICHT! device base address
#define I2C_PCF8574_DEV_ADDR  (0b001)    //!< TODO!ÄNDERN, FUNKTIONIERT NICHT! individual device address. !! ANPASSEN !!
#define I2C_PCF8574_ADDR      (I2C_PCF8574_BASE_ADDR | (I2C_PCF8574_DEV_ADDR)) //!< full 7-bit address
```

f. Sub-adresse:

| A2 | A1 | A0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

g. PIN Konfig: Resitive Pullup

2. I2C 8-bit I/O Expander
   a. Led geht gut
   b. SDA und SCL Konfig:




c. ...

```
/* +++ settings for PCF8574 Philips/NXP) 7(!)-bit address +++ */
#define I2C_PCF8574_BASE_ADDR (0x20)     //!< TODO!, ÄNDERN, FUNKTIONIERT NICHT! device base address
#define I2C_PCF8574_DEV_ADDR  (0b001)    //!< TODO!ÄNDERN, FUNKTIONIERT NICHT! individual device address. !! ANPASSEN !!
#define I2C_PCF8574_ADDR      (I2C_PCF8574_BASE_ADDR | (I2C_PCF8574_DEV_ADDR)) //!< full 7-bit address
```

d.
e. …
f. Die konkrete von I2C_PCF8574 wird durch Operator OR von Base_ADDR und DEV_ADDR bestimmt
g. Nach der Entfernung blinken die LEDs ohne Problem

3.

```
#ifdef READ_ADC
    /* +++ TODO +++ */
    /* ??? wie kann man erreichen, dass hier das Ende der Konversion abgewartet wird ??? */
    //sprintf( buffer, "vor Read: \n\r" );
    //UART_PutString( buffer );
    if (ADC_SAR_PD_IsEndConversion(ADC_SAR_PD_RETURN_STATUS)){

    // TODO!   // blocks until end of conversion
    /* !!! ADC-Wert in sarResult auslesen !!! */
     sarResult = ADC_SAR_PD_GetResult16();    //TODO!                          // read SAR
    /* !!! ADC-Wert anzeigen !!! */
    sprintf( buffer, "ADC: %d\n\r", sarResult );
    UART_PutString( buffer );
    }
```

4.

```
#else  // implementieren
    /* +++ TODO +++ */
    /* eigene Werte auf I/O-Expander anzeigen */

   sts = I2C_MasterClearStatus();  // clear master status
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error clear\n\r" );
    }
    sts = I2C_MasterSendStart( I2C_PCF8574_ADDR, 0);  // send address for write
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error address\n\r" );
    }
    sts = I2C_MasterWriteByte(~(sarResult / 16));              // write byte
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error write\n\r" );
    }
    sts = I2C_MasterSendStop();                       // stop comm
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error stop\n\r" );
    }
    CyDelay( 100 );
    i2cByte = ~i2cByte;
```

Der Wert von sarResult wird in READ_ADC geschrieben

5.

a. 2. Parameter in I2C_MasterSendStart(addr, ?):

```
* Parameters:
*  slaveAddress:  7-bit slave address.
*  R_nW:          Zero, send write command, non-zero send read command.
*
```

Lassen master auf Slave schreiben, oder nur lesen

b. Ja man kann neben dem I/O-Expander noch weitere I2C-Geräte anschließen
Da die Adresse ist ein 7-bit Wert, deshalb vermute ich, dass zusammen kann man ingesamt 127 Gerät schließen?