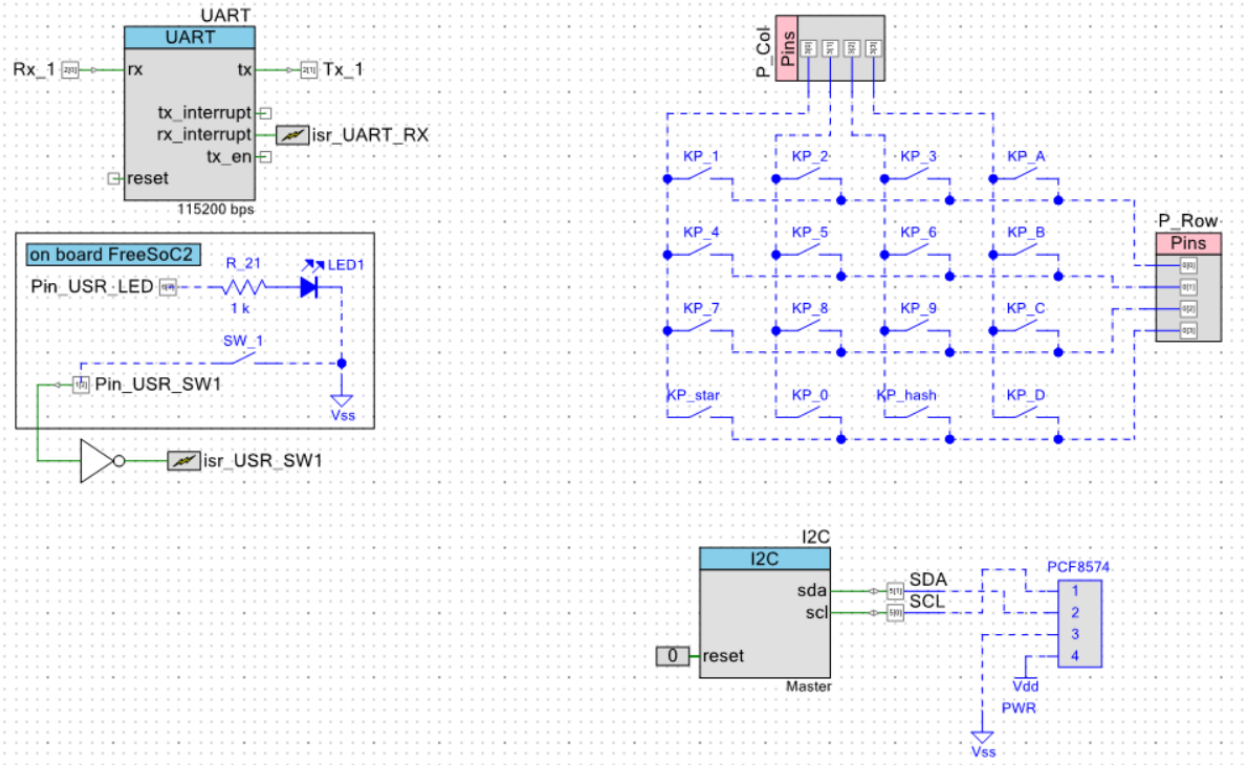
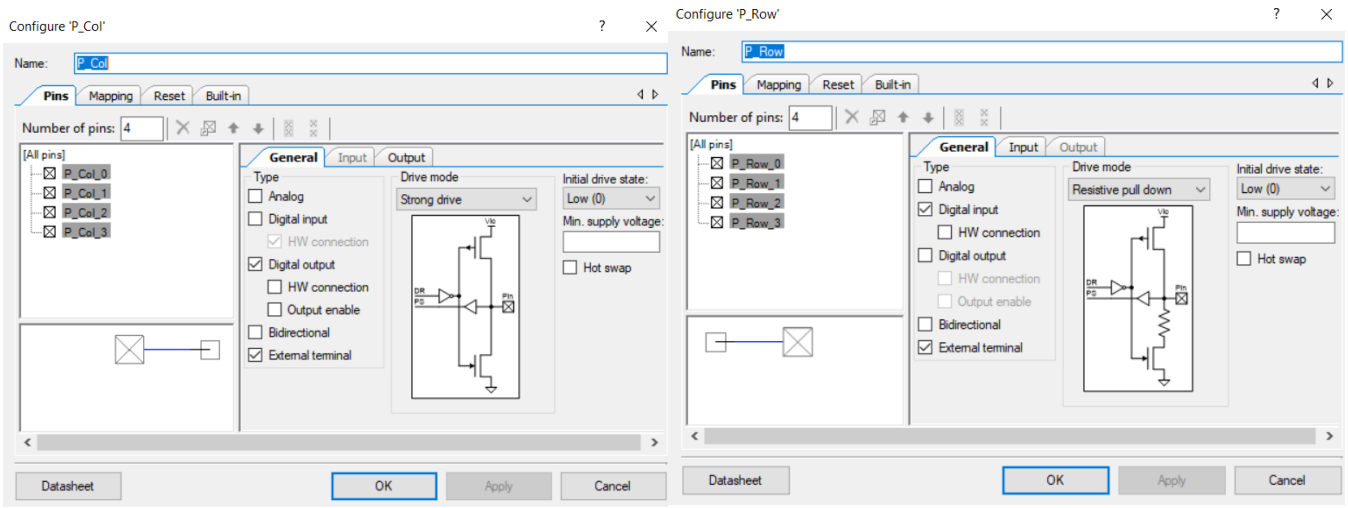
Mikroprozessorsysteme, Praktikum 5, SS22
Student 1: Damir Maksuti
Matrikelnr: 765984
Student 2: Jamil Boujada
Matrikelnr: 769479



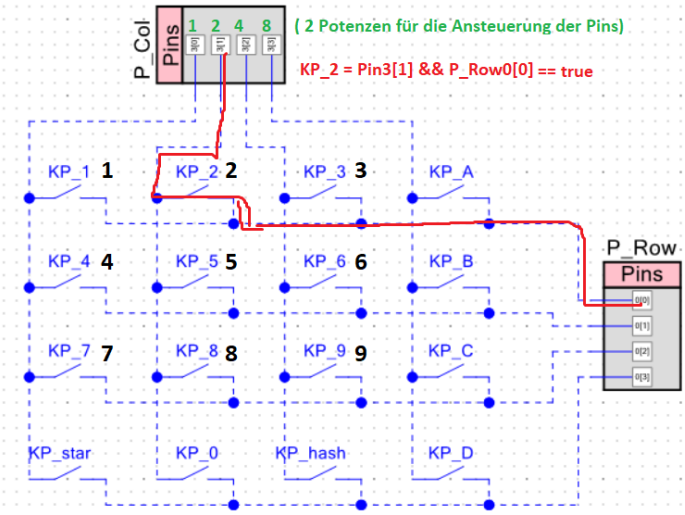
Aufgabe 1

a) Elektrische Pin- und Port-Konfiguration für die Tastatur vorschlagen



b) Funktion einer Matrixtastatur, siehe [2], erklären können.
Für die gedrückte Taste muss sowohl bei P_Col als auch bei P_Row ein High Signal anliegen. Da die anderen Tasten auf dem Keyboard nicht gedrückt wurden, liegt an den entsprechenden Row Pins ein Low an.

c) Skizze, wie Sie Zeilen und Spalten der Tastatur ansteuern und auslesen wollen, siehe [2].



Funktionen aus der Bibliothek sind:
P_Col_Read() & _Write()
P_Row_Read() & _Write()

d) C-Code für die Ansteuerung und Auslese der Tastatur vorbereiten.

```
char keyboard[4][4] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}

};
```

Hier haben wir unser Keyboard deklariert und initialisiert

```
    +++ wird ständig durchlaufen
for(;;)
{

    /* ++++++++++++++++++++++++++
     * +++ 4x4 Matrix Keypad +++
       ++++++++++++++++++++++++++ */
readRows(1);
readRows(2);
readRows(4);
readRows(8);
```

In der Main haben wir, wie üblich, die Endlosschleife, in der die Funktion readRows aufgerufen wird. Die Werte 1,2,4,8 entsprechen der binären Repräsentation, um die Columns ansteuern zu können.

```
int read(int binar){
    int zahl = -1;
    switch(binar){
        case 1: zahl = 0;
        break;
        case 2: zahl = 1;
        break;
        case 4: zahl = 2;
        break;
        case 8: zahl = 3;
        break;
    }
    return zahl;
}
```

```
void readRows(int col){
    P_Col_Write(col);

    int isValidRow = P_Row_Read();
    if(!isValidRow) {
        return;
    }
    int readRow = read(isValidRow);
    int readCol = read(col);

    currentKey = keyboard[readRow][readCol];

    // nur 1x print pro key
    if(currentKey != lastKey){
        print(readRow,readCol,lastKey);
    }
    lastKey=currentKey;

}
```

Innerhalb der readRows Funktion legen wir an der entsprechenden Spalte ein High Signal an. Bei den Pin Columns haben wir den Drive Mode: Strong Drive ausgewählt. Wenn ein High am DR anliegt schaltet der MOSFET und am Pin und im PS Buffer liegt ein HIGH an. Eine Anbindung zu PIN_ROW besteht jedoch erst dann, wenn die dazugehörige Keyboardtaste gedrückt wird. Anderfalls liegt an PIN_ROW ein LOW an, woraufhin aus der Funktion raus returned wird. Um herauszufinden welche Reihe aktuell ein Spannungssignal empfängt, haben wir die Funktion read geschrieben, die die binäre Repräsentation in den richtigen Index für die 2D Matrix umwandelt. Zum Schluss printen wir die entsprechende Taste mithilfe der UART.

e) Grundfunktion des I2C PCF8574 erläutern können, [3] und/oder [4]

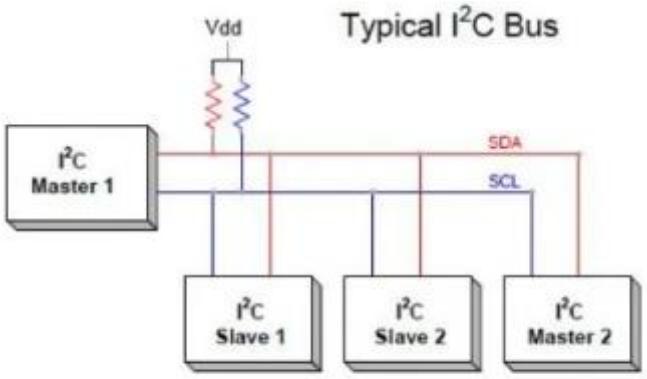


Bild 8: Beispiel für I2C-Bus,
Quelle: Cypress, AN50987

Der I2C übermittelt zunächst eine Adresse an die Slaves über die Datenleitung. Es gibt eine Clock und serielle Datenleitung. Jedem der angeschlossenen Teilnehmer wird eine eindeutige Adresse übermittelt, über die die Teilnehmer angesprochen werden können. Die Adresse ist 7 Bit lang und das niedrigwertiste Bit bestimmt ob auf die Slaves schreibend oder lesend zugegriffen wird. Die Hardware und Verbindung der Clock- und Datenleitung ist als Wired-And ausgelegt, sprich wenn nur einer der Teilnehmer auf Low geht, liegt an der gesamten Leitung ein Low an. Die Pegelerzeugung wird durch den Schalt- und Sperrvorgang des Transistors erreicht.

f) Bestimmen Sie die Basisadresse des PCF8574, [3], [4]
0x20 --> Basisadresse
0b101 --> Device Adresse
Basisadresse OR Device Adresse --> I2C Adresse

g) Welche Subadressen sind möglich (werden im Praktikum durch Jumper eingestellt)
Sofern wir die Frage richtig verstanden haben, haben wir durch den Jumper einen Addressraum von 3 Bits insgesamt. Somit müssten 8 Addresskombinationen möglich sein.

h) Elektrische Pin- und Port-Konfiguration für den I2C-Slave vorschlagen und begründen.
Siehe Aufgabe 1e)

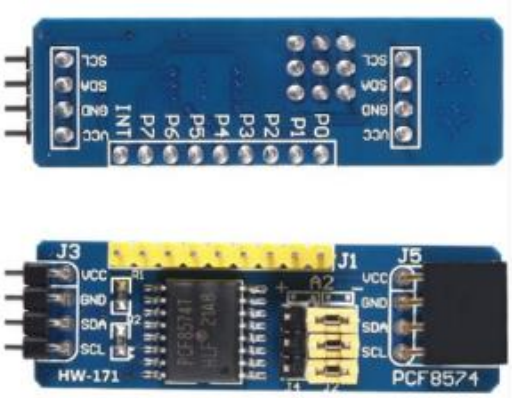
Aufgabe 2

b/c) Zeigen Sie die Tastendrücke (Spalte, Zeile) auf dem UART-Terminal an. Zeigen Sie das jeweils gedrückte Zeichen auf dem Terminal an.

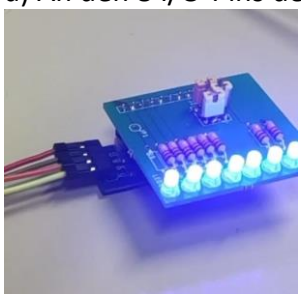
```
Row: 0| Column: 0  key pressed :Key: 1
Row: 0| Column: 1  key pressed :Key: 2
Row: 0| Column: 2  key pressed :Key: 3
Row: 1| Column: 0  key pressed :Key: 4
Row: 1| Column: 1  key pressed :Key: 5
Row: 1| Column: 2  key pressed :Key: 6
Row: 2| Column: 0  key pressed :Key: 7
Row: 2| Column: 1  key pressed :Key: 8
Row: 2| Column: 2  key pressed :Key: 9
```

```
Row: 3| Column: 0  key pressed :Key: *
Row: 3| Column: 1  key pressed :Key: 0
Row: 3| Column: 2  key pressed :Key: #
Row: 3| Column: 3  key pressed :Key: D
Row: 0| Column: 3  key pressed :Key: A
Row: 1| Column: 3  key pressed :Key: B
Row: 2| Column: 3  key pressed :Key: C
```

Aufgabe 3



a) An den 8 I/O-Pins des Expanders ist jeweils eine LED angeschlossen, welche active-low leuchtet.



b) Wie müssen die Pins SDA und SCL elektrisch konfiguriert werden, damit die Kommunikation funktionieren kann?
Resistive Pull up.

d) Definieren Sie die Device-Adresse I2C_PCF8574_BASE_ADDR siehe [3], [4]
0x20

e) Definieren Sie die Sub-Adresse I2C_PCF8574_DEV_ADDR Ihres I/OExpanders. Betrachten Sie dazu die Erläuterungen am Ende des Dokuments.
0b101

f) Wie wird daraus die korrekte vollständige Adresse I2C_PCF8574_ADDR bestimmt?
Durch eine Oder Verknüpfung der Base und Sub Addresse

g) Entfernen Sie den Kommentar vor #define I2C_PROJ_ON, wenn die Adresse stimmt, dann blinkt's!
(Siehe Aufgabe 3a)

Aufgabe 4
a) Senden Sie an den I/O-Expander nacheinander ein Byte so, dass ein Binärzähler entsteht, bei 0 (alles aus) beginnend. (Wählen sie ein Delay, z.B. 200ms damit man etwas sieht!)

```
// bei korrekter Adresse und Port-Konfig vollständig funktionsfähig
    /* ++++++++++++++++++++++++
     * +++ I2C I/O-Expander +++
     * ++++++++++++++++++++++++ */
    // TODO richtige Konfiguration der Ports vornehmen
    uint32  sts = I2C_MSTR_NO_ERROR;    // I2C transfer status (no error)
    // +++ blinking all LED's +++
    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(~i2cByte);              // write byte
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error write\n\r" );
    }
    i2cByte++;            // invert all bits for blinking
    sts = I2C_MasterSendStop();                       // stop comm
    if ( sts != I2C_MSTR_NO_ERROR ) {
        UART_PutString( "I2C error stop\n\r" );
    }

    CyDelay( 1500 );                // some delay
```

(Siehe Video im Anhang)

Aufgabe 5
a) Erklären Sie den 2. Parameter in I2C_MasterSendStart(addr, ?);

Laut der Dokumentation:

I2C_MasterSendStart()	Sends only a Start to the specific address.

uint8 I2C_MasterSendStart(uint8 slaveAddress, uint8 R_nW)
Der zweite Parameter gibt an ob lesend oder schreibend auf dem Slave zugegriffen wird

b) Könnte man mit dem I/O-Expander auch das Keypad ansteuern?
Man würde statt den 8 Ports am PSoC in unserer Lösung nur 2 für I2C benötigen, (und könnte obendrein noch andere I2C-Geräte anschließen)