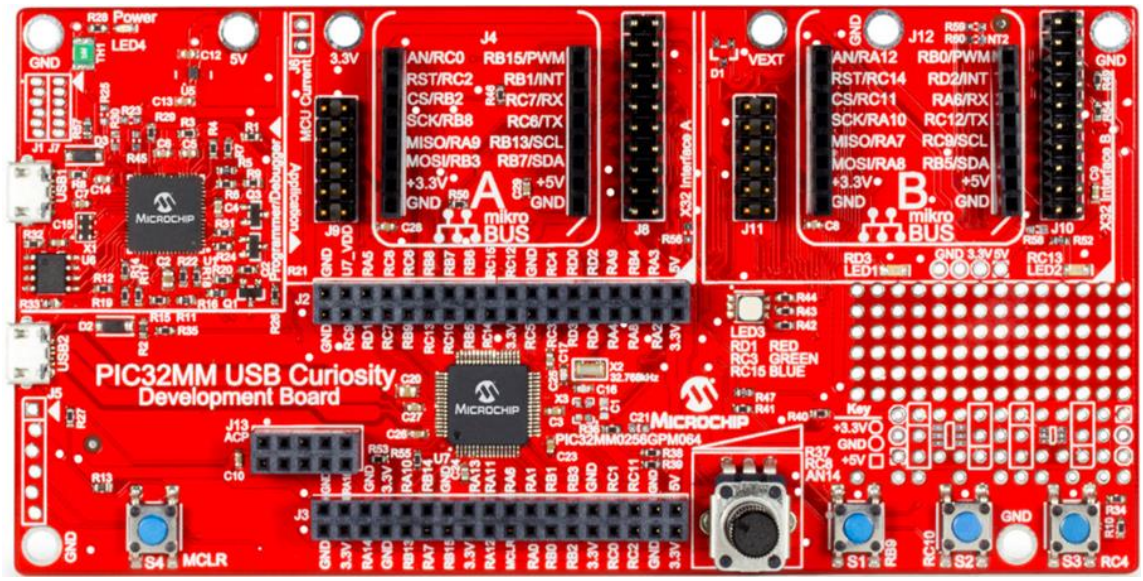
Rechnerarchitektur 2

DHBW Stuttgart, 2022
Prof. Dr.-Ing. Alfred Strey

Laborübung 1

Die vorliegende Platine des PIC32MM Curiosity Development Board mit PIC32MM0256GPM064 Mikrocontroller verfügt über diverse Komponenten, z.B.:



- User-Button `S1` verbunden mit PIC32MM0256GPM064 Pin 49 (TMS/RP14/SDA1/INT2 /**RB9**), zugehöriger Signalname auf dem Schaltplan ist `RB9_S1`
- User-Button `S2` verbunden mit PIC32MM0256GPM064 Pin 45 (OCM3F/**RC10**), zugehöriger Signalname auf dem Schaltplan ist `RC10_S2`
- User-Button `S3` verbunden mit PIC32MM0256GPM064 Pin 36 (OCM1E/INT3/**RC4**), zugehöriger Signalname auf dem Schaltplan ist `RC4_S3`
- Status `LED1` (rot) verbunden mit PIC32MM0256GPM064 Pin 33 (OCM3B/**RD3**), zugehöriger Signalname auf Schaltplan ist `RD3_LED1`
- Status `LED2` (rot) verbunden mit PIC32MM0256GPM064 Pin 47 (SCK1/**RC13**), zugehöriger Signalname auf Schaltplan ist `RC13_LED2`
- Reset-Button `MCLR` verbunden mit PIC32MM0256GPM064 Pin 9 (#MCLR).
- Zwei 36-polige Konnektoren `J2` und `J3` mit Signalen diverser E/A-Pins des PIC32MM0256GPM064 sowie Leitungen für 3.3V, 5V und GND
- Potentiometer zwischen 3,3V und GND, verbunden mit PIC32MM0256GPM064 Pin 52 (**AN14**/LVDIN/C2INC/RC8), zugehöriger Signalname auf Schaltplan ist `RC8_POT`
- Dreifarbige `RGB LED`,
  Farbe Rot verbunden mit Pin 53 (OCM1B/**RD1**), Signalname ist `RD1_RGB_RED`,
  Farbe Grün verbunden mit Pin 35 (OCM2B/**RC3**), Signalname ist `RC3_RGB_GREEN`,
  Farbe Blau verbunden mit Pin 42 (OCM3E/**RC15**), Signalname ist `RC15_RGB_BLUE`

Das Board arbeitet mit einer Spannung von 3,3V, die über einen Spannungsregler aus den 5V des USB-Anschlusses erzeugt werden.

Aufgabe 1: Erste Schritte mit MPLAB X und PIC32

1) Laden Sie von der Webseite https://www.microchip.com/mplab/mplab-x-ide die MPLAB X IDE herunter und installieren Sie diese mit den Gerätetreibern auf Ihrem Rechner. Am Ende werden Sie auf die Webseite des Microchip xc32 C-Compilers gelenkt, installieren Sie diesen ebenfalls.

2) Machen Sie sich mit dem Schaltplan des Boards und der Kurzdokumentation vertraut.

3) Starten Sie die MPLAB X IDE (z.B. über Icon auf Desktop), nicht die IPE!

4) Verbinden Sie das Board (am Connector USB1) über das USB-Kabel mit dem Rechner.

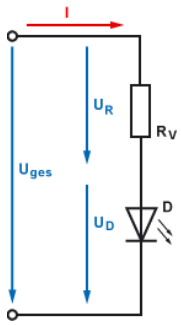
5) Beginnen Sie durch Auswahl von *File → New Project* ein neues Projekt, wählen Sie in den folgend erscheinenden Menus

- ein *Standalone Project* in der Kategorie *Microchip Embedded*
- den Prozessor *PIC32MM0256GPM064* aus der PIC32 Familie
- unter Tool das angezeigte *Starter Kit auswählen*
- als Compiler den *XC32* (*v2.xx* oder *v3.xx*)
- einen Namen für Ihr Projekt, das Flag bei *Set as main project* setzen bzw. gesetzt lassen

6) Fügen Sie die Dateien `main.c` und `system.c` aus dem Verzeichnis `Labor1` der Moodle-Seite unter *Source Files* der Vorlesung dem Projekt hinzu.

7) Compilieren Sie Ihr Programm (*Production → Build Main Project*) und laden Sie das generierte Binärprogramm im Hex-Format auf das Board herunter (Symbol *Run Main Project*). Dieses startet dann automatisch und lässt die rote LED1 blinken.

Aufgabe 2: Externe LED ein- und ausschalten



Nun soll an einen digitalen I/O-Port eine externe LED angeschlossen und diese über einen Taster eingeschaltet und durch nochmaliges Betätigen des Tasters wieder ausgeschaltet werden.

1) Schließen Sie hierzu auf dem Experimentierbrett über einen Vorwiderstand R_V eine LED an einen auf dem Board ungenutzten (!) digitalen I/O-Pin an. Berechnen Sie zunächst den erforderlichen Widerstand, wenn an der LED eine Spannung von U_D = 2,1 V abfällt und der Strom I = 10 mA nicht überschritten werden sollte. Die I/O-Pins des PIC32 erzeugen eine Spannung von 3,3V (für logisch 1).

Hinweis: Sie finden den zugehörigen Pin des I/O-Ports am Connector `J2` oder `J3` auf dem Board.

2) Erarbeiten Sie im *PIC32MM0256GPM064 Family Data Sheet* die Grundlagen der Programmierung der Register der I/O-Ports. Weitere Informationen zu den I/O Ports finden Sie auf der Microchip Webseite https://microchipdeveloper.com/32bit:frm oder können Sie als Datei `PIC32_IOPorts.pdf` von der Moodle-Seite der Vorlesung herunterladen.

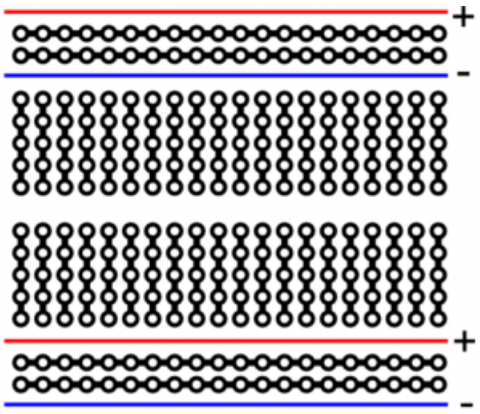
3) Schreiben Sie nun ein C-Programm, das direkt die Register `TRISx`, `PORTx`, und `LATx` (also z.B. `TRISC`, `PORTC`, … für Port C) anspricht, um in einer Endlosschleife den Taster `S1` abzufragen und die externe LED ein-/auszuschalten. Auch die Initialisierung der I/O Ports soll direkt über die Register erfolgen. Bitte beachten Sie, dass Sie nur die jeweils benötigten Bitleitungen konfigurieren und die anderen Bitleitungen unverändert lassen!

Hinweise:

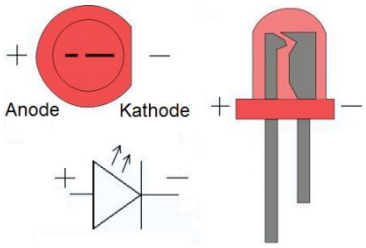
- Auch bei längerem Drücken des Tasters soll es nur eine Zustandsänderung geben!
- Sie finden die Zuordnung der Register zu den jeweiligen Adressen im Adressraum des PIC32MM übrigens in der über `#include <xc.h>` inkludierten Datei

  `C:\Programme\Microchip\xc32\v3.xx\pic32mx\include\proc\PIC32MM-GPM-0XX\p32mm0256gpm064.h`

– Bitte beachten Sie die interne Verdrahtung des Experimentierboards (s. Abb.)! Die Masseleitung (–) ist mit dem GND-Pin vom Konnektor `J2` oder `J3` des PIC32-Boards zu verbinden.



– Achten Sie auf die korrekte Polarität der LED:



Aufgabe 3: Blinken mit Timer

Nun soll ein Timer des PIC32MM eingesetzt werden, um die externe LED mit einer Frequenz von 1 Hz blinken zu lassen, nachdem diese über den User-Button S1 eingeschaltet wurde.

1) Machen Sie sich im *PIC32MM0256GPM064 Family Data Sheet* mit der Programmierung des Timers des PIC32MM vertraut.

2) Berechnen Sie den benötigten Wert des Prescalers für das Taktsignal des Zählers und den Wert für die Periodendauer `PRx` des Zählers `TMRx`.

3) Modifizieren Sie Ihr C Programm aus Aufgabe 2, damit die LED im eingeschalteten Zustand blinkt. Überprüfen Sie hierzu durch Abfragen des zum Timer gehörigen Interrupt-Bits im Register IFS0, ob die Zählperiode abgelaufen ist.

Hinweise: Das Interrupt-Bit im Register IFS0 muss per Software zurückgesetzt wird, damit es beim nächsten Ablauf des Timers wieder von der Hardware gesetzt werden kann. Die Generierung eines Interrupts ist in dieser Aufgabe nicht erforderlich.

4) Überprüfen Sie mit dem Sekundenzeiger einer Uhr, ob die LED ein Mal pro Sekunde aufblinkt; passen Sie ggf. die Periodendauer des Zählers an.

Aufgabe 4: LED Lauflicht

Es soll nun mit 5 LEDs ein Lauflicht implementiert werden, das von links (nur erste LED leuchtet) nach rechts (nur fünfte LED leuchtet) und dann wieder zurück läuft.

1) Schließen Sie hierzu fünf LEDs mit den Vorwiderständen an 5 logisch benachbarte (!) und auf dem Board ungenutzte (!) Pins des Ports A über die Konnektoren `J2` und `J3` an.

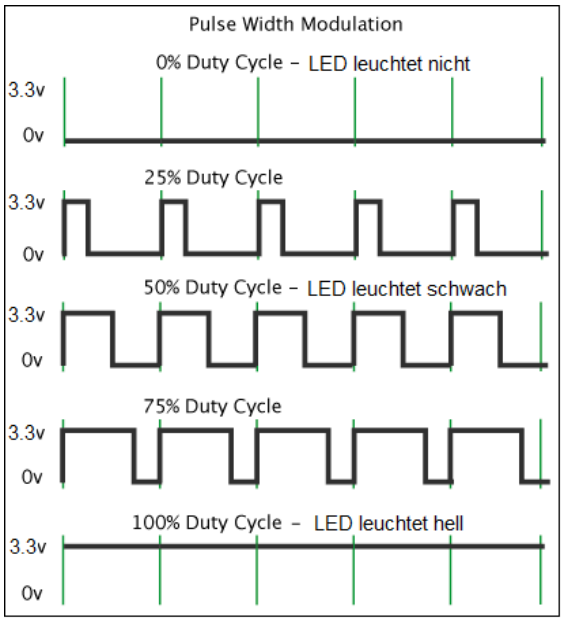
2) Schreiben Sie ein C-Programm, das durch Programmierung der Register von Port A das Lauflicht erzeugt. Das Lauflicht soll für jeden Durchlauf in einer Richtung 1 Sekunde benötigen. Verwenden Sie hierzu wie in Aufgabe 3 den Timer.

3) Machen Sie sich mit den Debug-Möglichkeiten des MPLAB X Tools vertraut. Setzen Sie hierzu einen Breakpoint (durch Mausklick auf die Zeilennummer des Quelltextes), übersetzen und starten Sie Ihr Programm über den Menupunkt *Debug→ Debug Main Project*. Schauen Sie sich am Breakpoint die Inhalte von Variablen an

Aufgabe 5: Fading einer LED mit PWM

Man kann die Helligkeit einer LED steuern, indem man sie durch ein PWM-Signal ansteuert und das Tastverhältnis ändert (s. Abb. unten). Hierzu muss die LED über einen Vorwiderstand an einen der *Output Compare* Ausgänge OCMxA bis OCMxF angeschlossen werden (vgl. Schaltplan des PIC32MM Curiosity Development Boards).

1) Machen Sie sich mit der Programmierung der Register der MCCP / SCCP Unit (*Master/Slave Capture/Compare/PWM* Unit) zur Generierung eines *Output Compare* Modus im *PIC32M Family Reference Manual, Section 30* (*Capture/Compare/PWM/Timer*) vertraut, das Sie von der Microchip Webseite https://microchipdeveloper.com/32bit:frm oder als Datei `PIC32MM_MCCP_SCCP.pdf` von der Moodle-Seite dieser Vorlesung herunterladen können. Insbesondere beachten Sie bitte im Kapitel 30.7 die Informationen zu den verschiedenen *Output Compare* Modi.



2) Überlegen Sie, wie die Register `CCPxCON1` bis `CCPxCON3` des gewählten Ausgangs OCMxA-F für die Generierung eines PWM-Signals konfiguriert werden müssen.

Hinweis: Die MCCP / SCCP Unit ist sehr komplex; die meisten Features werden hier nicht benötigt. Die zugehörigen Bits in den Kontrollregistern können daher auf 0 gesetzt bzw. belassen werden.

3) Desweiteren muss je nach gewählter Konfiguration des Ausgangs OCMx der integrierte Timer initialisiert werden. Hier müssen die Register `CCPxPR`, `CCPxRA` und ggf. `CCPxRB` des Timers konfiguriert werden.

4) Nun Schreiben Sie ein Programm, das ein "Fading" der LED realisiert, d.h. die Helligkeit der LED soll in einer Endlosschleife langsam und kontinuierlich von einem Minimum (Schwaches Glimmen der LED) zu einem Maximum (volle Helligkeit) ansteigen und dann wieder langsam vom Maximum zum Minimum abnehmen. Hierzu ist das Register `CCPxRA` entsprechend zu verändern.

Hinweise:

- Die Verwendung von Interrupts ist hier nicht erforderlich!
- Die nötigen Verzögerungen können entweder wie in Aufgaben 3 und 4 mit einem Timer oder durch Verwendung der Funktion `void delay_us(unsigned int us)` realisiert werden, die in der zur Verfügung gestellten Datei `system.c` definiert ist.

5) Schauen Sie sich das generierte PWM-Signal auf dem Oszilloskop an.

Wichtiger Hinweis am Schluss: Sichern Sie alle Quelldateien, damit Sie die entwickelten Programme ggf. als Vorlage für die nächste Laborübung wiederverwenden können!