# Bau und Programmierung eines Roboters für ROBOCUP JUNIOR RESCUE Teil 2.3: I2C

Bisher haben wir uns nur mit dem kleinen PIC beschäftigt. Wir haben aber auch noch den Großen.
Das Programm dafür schauen wir uns erst mal an
Der grundsätzliche Aufbau ist natürlich genauso, wie beim kleinen.
Wir müssen den Mikrocontroller zunächst initialisieren
Wir sehen, dass der Prescaler auf 1:2 eingestellt ist, später mehr dazu
Was für uns imMoment vor allem wichtig ist, sind die Einstellungen für den sog. I2C Bus
Und da v.a. SSPSTAT, wo wir angeben, dass wir den Chip grundsätzlich in High-Speed-Mode betreiben wollen
Außerdem sagen wir ihm, dass wir mit 1MHz Daten übermitteln wollen
Und schließlich, dass wir ihn als sog. Master arbeiten lassen
Der Rest später
Der I2C-Bus, der Inter Integrated Circuit Bus, also: Kommunikation zwischen zwei integrierten Schaltkreisen, hier: zwischen zwei PICs, später werden wir noch auf die Verwendung von I2C-Sensoren oder Aktoren eingehen
Und da wir keine Zeit verplempern wollen, packen wir den ins Interrupt
Der Master übernimmt die Rolle eines Kommunikationsleiters
Der Slave ist abhängig von den Befehlen des Masters, (ich persönlich finde die Bezeichnungen nicht besonders gelungen, aber sie heißen nun mal so)
Die beiden Leitungen, über die sich die beiden unterhalten, sind SDA, da werden die Daten übertragen und SCL, das ist die Clock-Leitung
Die Clock-Leitung gibt den Takt vor, in unserem Falle 1MHZ
Die Datenleitung kodiert die Daten
Immer wenn die Clock-Leitung HI, also +5V, ist und die Datenleitung auch, wird eine 1 übertragen, ist die Datenleitung LO, also GND, eine Null
Zunächst ruft der Master in die Runde: „Hey, 20, ich will dir ein Datum übermitteln“, denn 20 ist die Adresse unseres Slaves
Beim Slave… [im Init] Beim Master… [bei der Variablendeklaration]
Ist ein integrierter Schaltkreis mit der entsprechenden Adresse angeschlossen,
ruft der zurück: „Verstanden“, englisch acknowledge
Und weil die Leitungen normalerweise über einen Widerstand auf +5V gehalten wird, übermittelt der Slave die Antwort, indem er im richtigen Moment die Leitung auf 0V hält. Daher der Strich über ACK für „nicht“
Der zweite Wert ist dann ein Datum, das entweder der Slave übermittelt oder der Master

Ganz speziell bei den PICs wird die Kommunikation gestartet mit dem Befehl: „SEN“
und beendet mit PEN
Nochmal im Überblick
Und das schauen wir uns jetzt zunächst beim Master an
In unserer Endlosschleife initialisieren wir zunächst die I2C-Routine
Da es auch I2C-Geräte gibt, die eine andere Frequenz benötigen, setzen wir bei jedem Aufruf die entsprechende Frequenz fest, hier: SSPADD = 4,
Zähler zurück setzen, sehen wir noch
Dann haben wir zwei Flags die dem Programm mitteilen, ob geschrieben oder gelesen werden soll
Wir wollen lesen
Adresse des Slaves, den wir ansprechen wollen, hier: 20
Mit SEN wird die Übertragung gestartet und löst sofort einen Interrupt aus
Diesmal ist der Interrupt nicht durch den Tmr0, sondern durch den I2C ausgelöst worden: SSP Interrupt Flag ist 1
Wie immer: SSPIF löschen, sonst bleiben wir im Interrupt hängen
I2C_Write ist Null, wird also ignoriert
I2C_Read ist 1, also wird der Programmteil ausgeführt
I2C_Count ist Null, in das Register SSPBUF, ein Pufferregister, wird die Adresse und die Aufforderung zu lesen geschrieben
I2C_Count wird um 1 erhöht
Die switch-Funktion wird verlassen und damit der Interrupt
Wir springen also hier hin [Zeile 110] zurück,

und damit kein Tohuwabohu entsteht warten wir jetzt erstmal eine ms, später wird das überflüssig
Während wir eine ms warten, sendet jetzt das SSP-Modul automatisch den ersten Wert über den Bus
Ist es damit fertig und hat der Slave „verstanden“ gesendet gibt es wieder einen Interrupt
Beim Slave passiert folgendes:
Er hört seine Nummer, und reagiert damit, dass er beim nächsten Clocksignal die Datenleitung auf LO zieht
Danach hält er die Clockleitung auf GND, er sagt dem Master damit, dass er warten soll, bis er die notwendigen Daten schicken kann
Er merkt sich, dass er ein Datum schicken soll
Das angeforderte Datum ist in diesem Fall vereinbarungsgemäß der Zustand des Tasters
Mit CKP lässt er die Clock-Leitung wieder los

Beim Master führt der nächste Interrupt hier her [Zeile214]
Wir sagen dem Chip, er soll sich empfangsbereit machen
Danach gibt es wieder einen Interrupt, sobald das Datum vom Slave angekommen ist, das steht im Puffer
Das Datum speichern wir im dafür vorgesehenen Speicher ab
und bereiten das Senden an das LCD vor
Jetzt sind wir fertig, wir sagen dem Slave: „Lass gut sein, mehr wollen wir nicht“
Zum Schluß beenden wir die Kommunikation indem wir den Bus wieder frei geben
Alles weitere lest ihr in den Kommentaren
Warum betreiben wir einen solchen Aufwand? Lest selbst...
Beim Schreiben von Daten an den Slave, geht das alles ziemlich ähnlich. Deshalb erkläre ich das hier nicht.
Steht alles eigentlich in den Kommentaren und es gibt natürlich sehr viele Tutorials darüber auf youtube
Nur so viel: Den Zustand des Tasters senden wir zurück an den Slave, der den dann auch an der übermittelten Stelle ausgibt, zusätzlich geben wir noch den Analogwert des IR-Sensors an das LCD, 2. Tabulatorstelle.
Welcher IR-Sensor das ist? Das kriegst du schon raus.
Jetzt programmieren wir sowohl den Slave als auch den Master und schauen uns das Ergebnis im Video an:

Im nächsten Video schauen wir uns die AD Wandlung an.