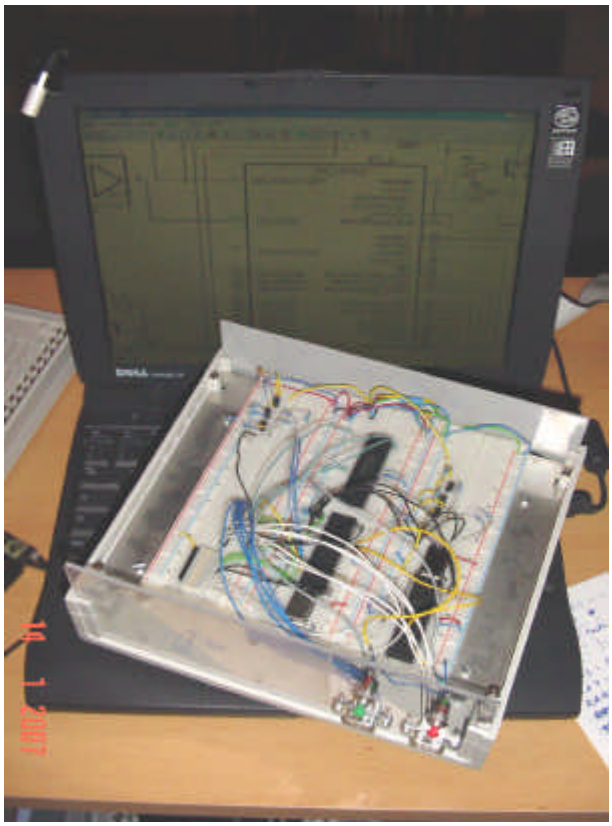
PIC-Programmer auf Basis eines PICs



Wettbewerb „Jugend Forscht“ 2007

Sebastian Hellberg (18 Jahre)

Arbeitsgemeinschaft „Jugend Forscht“

des Christian-Gymnasiums Hermannsburg

Leitung: StD Thomas Biedermann

Inhalt:

1. Einleitung

Die Idee zu meinem Projekt „PIC-Programmer auf Basis eines PICs“ entstand beim Arbeiten mit PIC-Mikroprozessoren. Ich hatte das Problem, dass meine Schaltung, in die der Microcontroller eingesetzt werden sollte, sich im Keller befand und auch mehr oder weniger ortsfest an verschiedene Messgeräte angeschlossen war. Mein PC, mit dem ich die Software in den PIC programmieren kann, befand sich zwei Stockwerke höher. Jede Softwareänderung bedeutete für mich, den Chip auszubauen, zum PC zu gehen, den Chip neu zu programmieren, wieder zurück in den Keller zu gehen und dort den Baustein wieder einzusetzen. Das war eindeutig zu aufwändig, außerdem ist es auch auf Dauer für ICs nicht besonders gut, wenn sie oft ein- und ausgebaut werden.

Somit war die Idee für mein Projekt geboren: Ein handliches Modul, dass ich mit dem PC verbinden kann, es mit Daten belade, an meinen zu programmierenden PIC anschließe und nur eine Taste betätigen muss, damit der Programmiervorgang gestartet wird.

Konkret bedeutet das: Ich werde in dieser Arbeit den Prototyp eines Gerätes vorstellen, dass ich am PC mit Daten beladen kann, welches dann völlig unabhängig vom PC den Speicher des zu beschreibenden PICs beschreiben kann. Das Herzstück dieses Gerätes ist ein PIC-Microcontroller. Als „Datenspeicher“ auf meinem Modul wird ebenfalls ein PIC benutzt. Dies ist ein neuartiges Konzept auf dem Gebiet des Beschreibens von Microcontroller-Programmspeichern.

2. Beschreibung von PIC-Microcontrollern

2.1. Allgemeines zu PIC-Microcontrollern

2.1.1. Aufbau und Funktionsweise

PICs der Familie 18F sind sehr hoch integrierte ICs der Firma Microchip, die zur Klasse der Microcontroller gehören. Vom Aufbau her ähneln sie sehr einem PC, das heißt, auf dem Chip sind eine RISC-CPU (Reduced Instruction Set Computing Central Processing Unit), ein mehrfach beschreibbarer Flash-Programmspeicher, ein RAM-Speicher, in vielen Fällen ein nicht-flüchtiger Datenspeicher (EEPROM) sowie bis zu 96 I/O-Pins, die als Schnittstellen wie zum Beispiel RS232, I²C, USB, Ethernet, als Anschluss für LC-Displays oder als frei

verwendbare Digital-I/Os genutzt werden können. Der Programmspeicher kann bis zu 128KiB groß sein, das RAM bis zu 4KiB und das EEPROM bis zu 1KiB. Die Geschwindigkeit der CPU liegt etwa bei maximal 10 MIPs (MIPs = Mega Instructions Per second), was etwa vergleichbar ist mit der Leistung eines Intel 80386 Mikroprozessors. Auch können weitere Funktionen wie hochauflösende A/D-Wandler mit integriert sein.

Die Funktionsweise des PICs ähnelt im weitesten Sinne der eines PCs: Ein Microcontroller kann von einem PC mit einer Software programmiert werden, sodass er genau das tut, was er tun soll. Dadurch sind diese Bausteine sehr flexibel einsetzbar.

Oft werden sie für Steuer- und Regelaufgaben in Geräten des täglichen Lebens eingesetzt, zum Beispiel in Waschmaschinen, Handys oder auch in der Steuerelektronik von Autos.

In der modernen Elektrotechnik verdrängen sie mehr und mehr konventionelle digitale Schaltungstechnik wie zum Beispiel die TTL- oder CMOS-ICs.

2.1.2. Programmierung von Software für PICs

Damit ein Microcontroller eine gewünschte Funktion übernehmen kann, muss er zunächst programmiert werden. Dazu wird in der Regel auf einem PC in einer Microcontroller-Entwicklungsumgebung ein Programm geschrieben. Übliche Programmiersprachen hierfür sind Assembler, C und seit einiger Zeit auch andere Hochsprachen wie BASIC. Dieses Programm wird dann compiliert und mithilfe eines Programmers, welcher an den PC angeschlossen ist, in den PIC geschrieben. Diese Geräte gibt es sowohl als reine Programmer als auch integriert auf so genannten Evaluation Boards. Ein Evaluation Board ist eine Platine, auf der, neben einem PIC, ein Programmer, verschiedene LEDs, Taster, eventuell ein Display und andere Bauteile, verbaut sind. Der Sinn eines Evaluation Boards ist, dass Benutzer von Microcontrollern sich mit ihren Funktionen vertraut machen können, ohne sich lange mit der Entwicklung von Hardware um den PIC herum beschäftigen müssen.

So programmiert, kann der PIC in seine Zielschaltung eingesetzt werden und seiner Funktion nachgehen.

2.2. Der von mir verwendete PIC18F452

2.2.1. Spezifikationen meines PICs

Der von mir verwendete PIC18F452 der Firma Microchip ist ein Microcontroller der „High Performance Enhanced“ - Baureihe. Er besitzt eine 8 Bit RISC-CPU mit einer Geschwindigkeit von maximal 10 MIPs, 32KiB Flash-Programmspeicher (bis zu 100 000 mal beschreibbar), 1,5KiB RAM sowie 256 Byte EEPROM – Datenspeicher.

Ein Befehlswort der CPU ist 16 Bit lang, sodass im Programmspeicher 16 KiWords Platz finden. Es gibt insgesamt 75 Assembler-Befehlsworte, die von der CPU unterstützt werden.

Die Architektur des Microcontrollers ist auf die Programmierung in C optimiert.

Nach außen hin besitzt er 35 digitale I/O – Leitungen; außerdem hat er verschiedene Schnittstellen wie zum Beispiel RS232 oder RS485 integriert. Auch besitzt er interne A/D – Wandler mit einer Auflösung von 10 Bit sowie Analog-Komparatoren.

Es gibt ihn sowohl im DIP40-Gehäuse als auch im PLCC- oder TQFP – SMD – Gehäuse mit 44 Pins. (Foto)

Foto 1: PIC-Microcontroller im PLCC- sowie im DIP40-Gehäuse

2.2.2. Verwendetes Evaluation Board

Auch ich habe, um erste Erfahrungen mit Microcontrollern zu sammeln, mit einem Evaluation Board, bestückt mit einem PIC18F452, zu arbeiten begonnen. Dieses Board wurde als Bausatz in der Zeitschrift „elektor“ 2/2005 vorgestellt. (Foto). Auf der Platine vorhanden sind vier LEDs, drei Taster, ein Anschluss für ein LC-Display, eine RS232-Schnittstelle, Leistungsausgänge für größere Lasten, ein Anschluss für einen Hardware-Debugger sowie ein Parallelport-Anschluss zum PC, über den der PIC programmiert wird.

Der auf der Platine vorhandene Programmer funktioniert nach dem MTSP Standard, wobei MTSP abgeleitet ist aus den Anfangsbuchstaben von „My Tait Serial Programmer“. Wie der Name schon sagt, wird der PIC durch diesen Programmer über den Parallelport eines PCs seriell programmiert, und zwar nach dem ICSP- (In Circuit Serial Programming) Standard im High Voltage Programming (HVP) Modus. Was diese Art der Programmierung bedeutet und wie sie funktioniert, wird unter Punkt 3.1. noch näher beschrieben.

Dieses Board habe ich zusammen mit einem LC-Display und einer Stromversorgung in ein Gehäuse eingebaut. Außerdem habe ich die Leitungen, die für die Programmierung des PICs zuständig sind, mit einer Buchse herausgeführt, damit ich eine externe Fassung für zu programmierende PICs anschließen kann.

Das ermöglicht mir, auch andere als den fest auf dem Board eingebauten PIC zu programmieren.

2.2.3. Verwendete Programmierumgebung

Die von mir verwendete Programmierumgebung ist die MPLAB IDE von Microchip. Sie kann als Freeware von der Homepage [6] heruntergeladen werden. Als Ergänzung habe ich dazu noch den MPLAP C18 Compiler derselben Firma verwendet, der es ermöglicht, in der MPLAB IDE mit der Programmiersprache C zu arbeiten. Auch dieser ist als Freeware als „Student Edition“ mit eingeschränktem, für die meisten Zwecke aber völlig ausreichendem Funktionsumfang, auf derselben Homepage zu finden.

3. Beschreiben des Programmspeichers – in der Theorie

3.1. Das ICSP-Programmierverfahren

Wie schon unter Punkt 2.2.2. erwähnt, lassen sich PIC-Microcontroller nach dem In- Circuit Serial Programming – Verfahren beschreiben. Wie der Name dieses Verfahrens schon sagt, kann das Beschreiben des Programmspeichers im eingebauten Zustand erfolgen, was sich als sehr vorteilhaft erweist. Es kann aber auch im ausgebauten Zustand erfolgen. Außerdem findet es über einen speziellen seriellen Bus statt, was den Vorteil hat, dass nur wenige Leitungen für die Programmierung benötigt werden.

Bei dieser Methode gibt zwei Modi, mit denen gearbeitet werden kann: Den High Voltage (HVP) – Modus und den Low Voltage (LVP) – Modus. Der LVP – Modus ist zwar hardwareseitig etwas einfacher zu realisieren als der HVP – Modus, da keine Programmierspannung von +12V erforderlich ist; allerdings lassen sich in diesem Modus nicht alle Bereiche des Speichers komplett beschreiben, was ihn zur Komplettprogrammierung von PICs als nicht hundertprozentig geeignet erscheinen lässt.

Deshalb werde ich in meinen weiteren Ausführungen nur noch den HVP – Modus beschreiben.

Für das Programmieren eines PIC-Microcontrollers im eingebauten Zustand sind nur vier Leitungen erforderlich: Die Leitungen MCLR, SDATA, SCLK und natürlich eine für die Masse.

Auch außerhalb einer Anwendungsschaltung wird nur noch eine zusätzliche Leitung für die Betriebsspannung benötigt.

Über die Leitung MCLR, was die Abkürzung für Memory Clear ist, werden die Betriebszustände des PIC kontrolliert. Wird diese Leitung auf Masse gelegt, so wird ein Reset durchgeführt. Liegt diese Leitung auf Vcc (Betriebsspannung, im Bereich von +3,3V bis +5V), ist der PIC im normalen Betriebszustand. Auf Vpp (Programmierspannung, +12V) gelegt, befindet sich der Microcontroller im Programmiermodus. So kann sein Speicher beschrieben, aber auch ausgelesen werden.

Die Leitungen SDATA und SCLK bilden zusammen einen seriellen Bus, über den die Daten geschrieben und gelesen werden können. SCLK steht dabei für Serial Clock und SDATA für Serial Data. Die Leitung SDATA ist Bidirektional, das heißt Daten können über denselben Pin gelesen und geschrieben werden. SCLK, die Taktleitung, hingegen ist ein reiner Eingang, das heißt, das komplette Bustiming wird sowohl beim Lesen als auch beim Schreiben von außen gesteuert.

Daten werden über diesen Bus jeweils Blockweise in 16-Bit Blöcken, auch genannt „Data Payloads“, übertragen, vor die jeweils noch ein 4-Bit Kommandoblock vorangestellt wird. Somit werden Daten immer in 20-Bit Blöcken über den seriellen Bus geschoben. Bei jeder Low-Flanke auf der SCLK-Leitung erfolgt die Datenübernahme eines Bits.

Es gibt insgesamt 10 verschiedene 4-Bit Kommandos, die wie folgt lauten:

Kommando:	Name:
0000	Kernbefehl, 16 Bit Daten übergeben
0010	Inhalt von TABLAT Register ausgeben
1000	Lese
1001	Lese, erhöhe danach Adresspointer
1010	Lese, erniedrige danach Adresspointer
1011	Lese, erhöhe davor Adresspointer
1100	Schreibe
1101	Schreibe, erhöhe danach Adresspointer um 2
1110	Schreibe, erniedrige danach Adresspointer um 2
1111	Schreibe, starte Programmiervorgang

Tabelle 1: 4-Bit Befehle des PIC18F452

In der Abb. X ist ein Timing-Diagramm zu sehen, aus dem abgelesen werden kann, wie der serielle Datentransfer abläuft. Sehr wichtig ist hierbei die Reihenfolge, in der die Bits des 4-Bit Kommandos und des 16-Bit Datenblocks ausgegeben werden: Beim Kommando, welches als erstes übertragen wird, das LSb (Least Significant bit) zuerst. Beim Datenblock wird mit der Ausgabe des Low-Bytes begonnen, bei dem wieder zuerst das LSb ausgegeben wird, gefolgt vom High-Byte, bei dem ebenfalls das LSb zuerst ausgegeben wird.

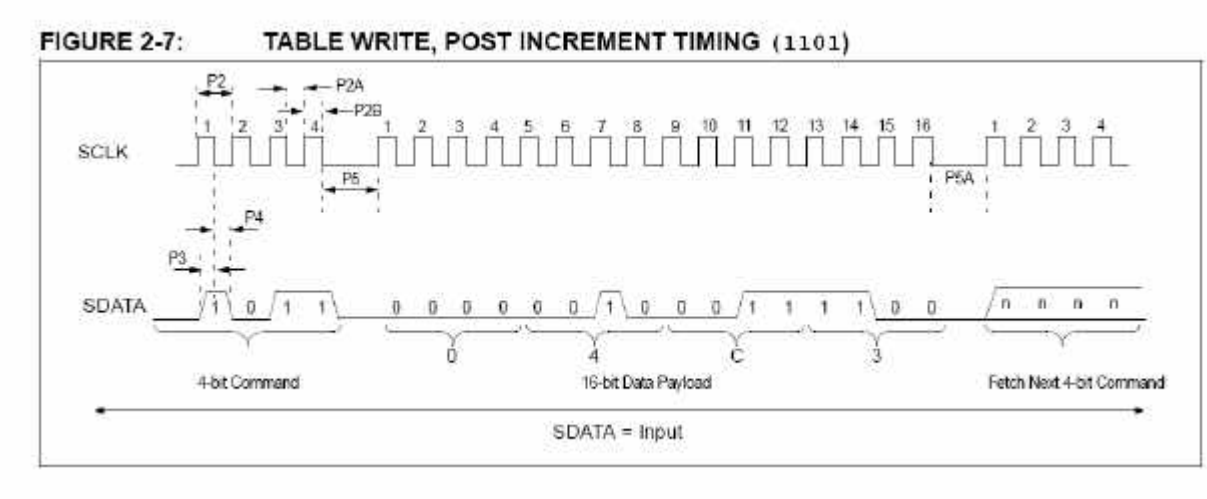


Abb. 1: Timing-Diagramm zum Senden eines 20-Bit Blocks (aus [3], Seite 6)

Der Flash-Speicher eines PICs ist unterteilt in vier Blocks: Boot-Block, Programm-Block, Config-Block und ID-Block; außerdem ist noch der EEPROM-Datenspeicher vorhanden. Der Programmblock ist noch mal unterteilt in 4 Panels. Jeder dieser Blöcke muss auf eine andere Art und Weise geschrieben werden und das Beschreiben der Blöcke muss in einer bestimmten Reihenfolge erfolgen. Diese Reihenfolge ist in der Abb. 2 dargestellt.

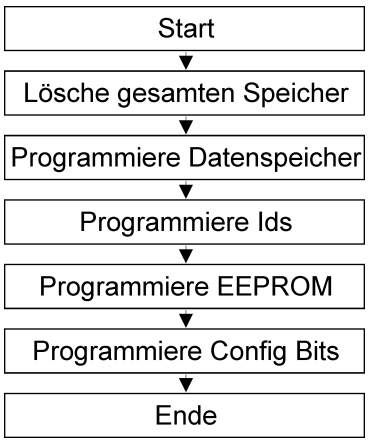


Abb. 2: Flussdiagramm Programmierung

Jede Speicherzelle, egal in welchem Block, hat eine fest zugewiesene Adresse. Bevor nun eine Speicherzelle gelesen oder beschrieben werden kann, muss erst der Adresspointer an diese Stelle gesetzt werden. Das passiert durch eine bestimmte Folge von 20-Bit Blöcken, bestehend aus Kommando und Adress-Bytes.

Die Art und Weise der Programmierung der einzelnen Blöcke ist eindeutig im Datenblatt [3] festgelegt.

3.1.1. Anforderungen an die Programmer – Hardware

Die Anforderungen, die an die Programmer – Hardware gestellt werden, sind relativ niedrig, da auf Hardware – Ebene keine „Intelligenz“ gefordert wird.

Falls von einem PC aus programmiert werden soll, wird nichts anderes benötigt als ein Bustreiber wie zum Beispiel der 74HCT541, der zwischen der SDATA- und SCLK – Leitung und dem Port des PCs geschaltet wird. Falls von einem PIC aus programmiert wird, können diese sogar entfallen. Außerdem wird eine Schaltung benötigt, welche digital über weitere Portleitungen gesteuert den MCLR – Pin auf Masse, Vcc oder Vpp legen kann. Auch dieser Teil der Schaltung ist nicht sehr kompliziert, sodass insgesamt die Schaltung des Programmers relativ einfach und übersichtlich ist.

3.2.2. Anforderungen an die Programmer - Software

Die Anforderungen, die an die Software auf dem PC oder im PIC gestellt werden, sind wesentlich höher als die, die an die Hardware gestellt werden. Das liegt daran, dass die gesamte Intelligenz des Programmers in die Software implementiert werden muss. Es muss zum Beispiel eine komplette Portsteuerung für den PIC-Programmierbus mit serieller Datenausgabe entwickelt werden. Auch müssen verschiedenste Adress-Zählerschleifen, die teilweise noch ineinander verschachtelt sind, geschrieben werden.

Insgesamt muss sehr hardwarenah programmiert werden, was in der heutigen Zeit der Hochsprachen – Programmierung nicht immer sehr einfach ist. Somit ist die Software, die hinter einem Programmer steht, durchaus als komplex anzusehen.

4. Beschreiben des PIC-Programmspeichers – In der Praxis

4.1. Prinzip meines Programmers

Mein Programmer funktioniert – im Gegensatz zu den herkömmlichen Programmern, weitestgehend ohne PC. Der Programmer muss nur einmal an den PC angeschlossen werden, um die Daten zu bekommen, die er programmieren soll. Danach arbeitet er völlig unabhängig vom PC. Er muss dann nur noch mit dem zu programmierenden PIC verbunden werden. Nach einem Tastendruck nimmt er dann seine Arbeit auf und beschreibt den Speicher des Ziel-PICs.

Dazu sind in meinem Gerät 2 Microcontroller vorhanden: Ein Controller, der als Datenspeicher benutzt wird und ein zweiter, der den Datenspeicher ausliest und den Ziel-Controller beschreibt.

Der Datenspeicher-PIC ist mit einem Steckverbinder ausgerüstet, über den er mit meinem Evaluation Board verbunden werden kann, über welches dann das Programm, das nachher in den Ziel-PIC übertragen werden soll, in den Datenspeicher-PIC geladen wird.

Angeschlossen an den Ziel-Controller, führt der Programmier-PIC eine Spiegelung des Datenspeicher-Controllers auf das Ziel durch. Das setzt voraus, dass der Datenspeicher-Baustein und der Ziel-Baustein vom selben Typ sind.

4.2. Aufbau der Hardware

Wie schon unter Punkt 3.2.1. beschrieben, sind die Anforderungen, die an die Programmer-Hardware gestellt werden, nicht allzu hoch. Deshalb ist auch die Schaltung als relativ übersichtlich zu bezeichnen. Ein Schaltplan der kompletten Schaltung mit allen drei PICs ist im Anhang A1 zu finden.

Die Schaltung habe ich auf einem Experimentier-Steckbrett aufgebaut, da es hier am einfachsten und schnellsten geht, falls eine Änderung an der Schaltung vorgenommen werden muss.

4.2.1. Beschaltung der PICs

Der Datenspeicher-PIC (im Schaltplan ganz links) benötigt keine externen Bauteile, um arbeiten zu können. Es sind, neben der Versorgungsspannung, nur 3 Leitungen an den PIC angeschlossen: SCLK (im Schaltplan als PGC bezeichnet), SDATA (im Schaltplan als PGD bezeichnet) sowie MCLR. Diese Leitungen gehen sowohl zur externen Anschlussbuchse als auch zu dem anderen PIC. Dadurch kann dieser Baustein entweder vom PC beschrieben werden oder vom Programmer-PIC (im Schaltplan der mittlere) ausgelesen werden. An diesem PIC wird kein Quarzgenerator oder ähnliches zur Takterzeugeung benötigt, da beim Schreiben oder Lesen des Controllers kein externer Takt benötigt wird.

Der Programmer-PIC benötigt noch etwas mehr Beschaltung: es sind 2 Status-LEDs sowie zwei Taster angeschlossen. Ein Taster löst bei Betätigung einen Hardware-Reset des PIC aus, der andere startet bei Betätigung den Programmiervorgang. An diesen PIC sind die SCLK

(PGC) - und SDATA (PGD) – Leitungen der anderen PICs angeschlossen. Außerdem werden über jeweils 2 Portpins und mithilfe einer kleinen Transistorschaltung an die MCLR-Pins die verschiedenen Pegel (GND, +5V, +12V), die für die verschiedenen Betriebszustände zuständig sind, angelegt. Außerdem ist noch ein Quarzgenerator angeschlossen, welcher dem PIC seine Taktfrequenz liefert, sodass dieser arbeiten kann.

Am Ziel-PIC (rechts im Schaltplan) sind zusätzlich zu einem Quarzgenerator und den Leitungen MCLR, SCLK (PGC) und SDATA (PGD) noch vier LEDs angeschlossen, die zu Diagnosezwecken verwendet werden.

4.2.2. Transistorschaltung zum Schalten der Betriebsspannung

Da ich direkt mit einer Portleitung keine Programmierspannung an einen PIC anlegen kann, musste ich noch eine kleine Schaltung aufbauen, die es mir ermöglicht, mithilfe zweier Portleitungen drei verschiedene Pegel an eine Leitung anzulegen.

Liegt an keiner der 2 Basen der Transistoren eine Spannung an, so liegen +5V am Ausgang an. Liegt an der Basis des in der Schaltung unteren Transistors (T3 bzw. T5 im Schaltplan) ein High - Pegel an, so wird der Ausgang auf Masse gezogen. Liegt an der Basis des anderen Transistors (T4 bzw. T6 im Schaltplan) ein High-Pegel an, wird über einen weiteren Transistor (Q2 bzw. Q1 im Schaltplan) der Ausgang auf +12V gelegt.

Diese Schaltung hat den Vorteil, dass sie gegen „Fehlbedienung“ geschützt ist, da falls fälschlicherweise an beiden Basen ein High-Pegel anliegt, die fließenden Ströme so gering sind, dass nichts beschädigt werden kann.

4.3. Meine Software im PIC

Meine Software, die ich für den Programmier-PIC geschrieben habe, ist relativ komplex. Sie besteht aus etwa 700 Zeilen Programmcode in der Programmiersprache C. Es würde den Rahmen dieser Arbeit sprengen, den Code an dieser Stelle komplett abzudrucken und ihn ausführlich zu beschreiben. Deshalb werde ich mich hier auf die Beschreibung des Funktionsprinzips des Programms sowie einiger wichtiger Prozeduren beschränken.

Um in einem so großen Programm nicht den Überblick zu verlieren, habe ich das Programm in Prozeduren aufgeteilt:

```
extern void SleepMS(int MilliSeconds);
void LEDTest (void);
void S0000 (unsigned char PL1, unsigned char PL2);
void S0010 (unsigned char PL1, unsigned char PL2);
void S1000 (unsigned char PL1, unsigned char PL2);
void S1001 (unsigned char PL1, unsigned char PL2);
void S1010 (unsigned char PL1, unsigned char PL2);
void S1011 (unsigned char PL1, unsigned char PL2);
void S1100 (unsigned char PL1, unsigned char PL2);
void S1101 (unsigned char PL1, unsigned char PL2);
void S1110 (unsigned char PL1, unsigned char PL2);
void S1111 (unsigned char PL1, unsigned char PL2);
void M0000 (unsigned char PL1, unsigned char PL2);
void M0010 (unsigned char PL1, unsigned char PL2);
void M1000 (unsigned char PL1, unsigned char PL2);
void M1001 (unsigned char PL1, unsigned char PL2);
void M1010 (unsigned char PL1, unsigned char PL2);
void M1011 (unsigned char PL1, unsigned char PL2);
void M1100 (unsigned char PL1, unsigned char PL2);
void M1101 (unsigned char PL1, unsigned char PL2);
void M1110 (unsigned char PL1, unsigned char PL2);
void M1111 (unsigned char PL1, unsigned char PL2);
void PL_out_S (unsigned char PL_out_S_1, unsigned char PL_out_S_2);
void PL_out_M (unsigned char PL_out_M_1, unsigned char PL_out_M_2);
unsigned char Data_read (void);
void Set_pointer_M (unsigned long Adr);
void Set_pointer_S (unsigned long Adr);
```

Dies ist der Teil des Programms, in dem ich die Prozeduren definiere.

Die Funktionen Sxxxx sind für die Ausgabe des jeweiligen 4-Bit Kommandos zum Ziel-PIC zuständig; die Funktionen mit den Namen Mxxxx tun dasselbe, jedoch erfolgt die Ausgabe an den Datenspeicher-PIC.

Die Funktionen „PL_out_S“ und „PL_out_M“ geben die 16-Bit Datenblöcke, die auf das 4-Bit Kommando folgen, seriell entweder an den Ziel-PIC („PL_out_S“) oder an den Datenspeicher-PIC („PL_out_M“) aus.

Die Funktion „Data_read“ liest, nach setzen des Adresspointers im Datenspeicher-PIC, den Inhalt einer Speicherzelle aus.

Mit den Funktionen „Set_pointer_M“ und „Set_pointer_S“ werden die Adresspointer im Datenspeicher- bzw. Ziel-PIC gesetzt.

Eine Routine zum Ausgeben eines 4-Bit Kommandos sieht so aus:

```
void S1011(unsigned char PL1, unsigned char PL2)
{
PORTA = 0b00011010;                                  // SCLK Clock high, SDATA = 1
PORTA = 0b00010010;                                  // SCLK Clock low, SDATA = 1
PORTA = 0b00011010;                                  // SCLK Clock high, SDATA = 1
PORTA = 0b00010010;                                  // SCLK Clock low, SDATA = 1
PORTA = 0b00001010;                                  // SCLK Clock high
PORTA = 0b00000010;                                  // SCLK Clock low
PORTA = 0b00011010;                                  // SCLK Clock high, SDATA = 1
PORTA = 0b00010010;                                  // SCLK Clock low, SDATA = 1
PL_out_S (PL1, PL2);
}
```

In dieser Routine wird durch explizites Setzen der Portleitungen das 4-Bit Kommando „1101“ ausgegeben, danach wird die Funktion „PL_out_s“ aufgerufen, welche die schon beim Aufruf der Funktion „S1011“ mit übergebenen 16 Bit Daten, abgelegt in den beiden 8-Bit Variablen „PL1“ und „PL2“, seriell ausgibt.

Die serielle Ausgabe der Daten möchte ich hier am Beispiel der Funktion „PL_out_S“ zeigen:

```
void PL_out_S (unsigned char PL_out_S_1, unsigned char PL_out_S_2)
{
unsigned char i = 1;
unsigned char BitCnt = 1;
for (i=1; i<=8; i++)
{
if ((PL_out_S_2 & BitCnt) != 0)
{
PORTA = 0b00011010;                  // SCLK Clock high, SDATA = 1
PORTA = 0b00010010;                  // SCLK Clock low
}
else
{
PORTA = 0b00001010;                  // SCLK Clock high
PORTA = 0b00000010;                  // SCLK Clock low
}
BitCnt = (BitCnt * 2);
}
i = 1;
BitCnt = 1;
for (i=1; i<=8; i++)
{
if ((PL_out_S_1 & BitCnt) != 0)
{
PORTA = 0b00011010;                  // SCLK Clock high, SDATA = 1
PORTA = 0b00010010;                  // SCLK Clock low
}
else
{
PORTA = 0b00001010;                  // SCLK Clock high
PORTA = 0b00000010;                  // SCLK Clock low
}
BitCnt = (BitCnt * 2);
}
}
```

Hier wird mithilfe von Bit-Maskierungen der Inhalt der zwei Variablen „PL_out_S_2“ und „PL_out_S_1“ seriell ausgegeben. Das ganze passiert in einer Schleife. Je nach dem, ob das durch die Variabel „BitCnt“ maskierte Bit null oder nicht null ist, wird eine null oder eine eins ausgegeben. Bei jedem Schleifendurchlauf wird der Wert der Variable „BitCnt“ verdoppelt, was bewirkt, dass ein Bit nach dem anderen maskiert und ausgegeben wird.

Auch das Setzen des Adress-Pointers in den Funktionen „Set_pointer_M“ und „Set_pointer_S“ funktioniert auf die gleiche Art. Auch beim Einlesen in der Prozedur „Data_read“ verwende ich eine ähnliche Technik.

Durch diese verschiedenen Prozeduren wird der Programmaufbau im Hauptprogramm sehr übersichtlich, was ich nun an einem Beispiel zeigen möchte:

```
S0000 (0x8E, 0xA6);
S0000(0x8C, 0xA6);
S0000 (0x0E, 0x3C);
S0000 (0x6E, 0xF8);
S0000 (0x0E, 0x00);
S0000 (0x6E, 0xF7);
S0000 (0x0E, 0x06);
S0000 (0x6E, 0xF6);
S1100 (0x00, 0x00);
S0000 (0x8E, 0xA6);
S0000 (0x9C, 0xA6);
Set_pointer_M (0x200000);
Set_pointer_S (0x200000);
S1101 (Data_read(), Data_read());
S1101 (Data_read(), Data_read());
S1101 (Data_read(), Data_read());
S1111 (Data_read(), Data_read());
PORTA = 0b00001010;                          // SCLK Clock high
PORTA = 0b00000010;                          // SCLK Clock low
PORTA = 0b00001010;                          // SCLK Clock high
PORTA = 0b00000010;                          // SCLK Clock low
PORTA = 0b00001010;                          // SCLK Clock high
PORTA = 0b00000010;                          // SCLK Clock low
PORTA = 0b00001010;                          // SCLK Clock high
SleepMS(2);                                  // Programmierzeit
PORTA = 0b00000010;                          // SCLK Clock low
SleepMS(1);                                  // "High voltage discharge" Zeit
```

Hier werden die acht ID-Bytes aus dem ID-Datenblock geschrieben. An diesem Beispiel kann man, unter Zuhilfenahme der Tabelle 1, auch das grundlegende Funktionsprinzip meiner Software erkennen: Es wird nicht mit Zwischenspeichern gearbeitet, sondern es wird ein Byte gelesen und danach direkt wieder geschrieben. Es findet dadurch eine 1 : 1 Spiegelung des Datenspeicher-PICs statt.

5. Systemtest

Zum Testen habe ich meine Schaltung auf einem Experimentier – Steckbrett aufgebaut und sichergestellt, dass alle Verbindungen und Bauteile richtig gesteckt wurden. Dann habe ich zunächst nur ein einfaches Programm in den Programmier – PIC geladen, welches nur den Speicher des Ziel – PIC löscht. Als das funktionierte, tastete ich mich Schritt für Schritt zu einem funktionierenden Programm heran, welches dann die gewünschte Funktion der Spiegelung des kompletten Speichers durchführte.

6. Probleme beim Schaltungsaufbau und bei der Softwareentwicklung

Beim Schaltungsaufbau gab es eigentlich keine grundlegenden Probleme. Ich hatte nur einmal ein Problem mit einem defekten internen Pull-Up-Widerstand in einem der Digitalausgänge

des PIC, sodass dieser Ausgang keinen High-Pegel mehr erreichen konnte. Nach Einsetzten eines externen Pull-Up-Widerstandes lief dann die Schaltung auch wieder.

Bei der Software hatte ich zu Anfang das Problem, dass ich die Bits der 20-Bit Blocks in der verkehrten Reihenfolge ausgegeben, das heißt, ich habe fälschlicherweise das MSb (Most Significant bit) anstatt des LSb (Least Significant bit) zuerst ausgegeben. Somit konnte natürlich nichts funktionieren. Aber auch dieses Problem war schnell durch eine Software-Änderung behoben.

7. Ausblick

Noch ist meine Schaltung ein Prototyp, der auf einem unhandlichen Steckbrett in nicht gerader robuster Art und Weise aufgebaut ist. Deshalb werde ich mich in der nächsten Zeit damit beschäftigen, meine Schaltung auf einer Platine in kompakter Form aufzubauen.

Außerdem dauert der Programmiervorgang im Moment noch relativ lange, was ich gerne zum Beispiel durch eine effizientere Softwaregestaltung ändern würde.

8. Quellen:

[1]: Binärpräfixe: http://physics.nist.gov/cuu/Units/binary.html

[2]: Elektor Ausgabe 2/2005, S. 34-40, Elektor-Verlag Aachen, 2005

[3]: PIC18FXX2/XX8 FLASH Microcontroller Programming Specification, Microchip Technology Inc., 2002.
Heruntergeladen von http://ww1.microchip.com/downloads/en/DeviceDoc/39576b.pdf

[4]: PIC18FXX2 Data Sheet, Microchip Technology Inc., 2002. Heruntergeladen von http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

[5]: www.microchip.com

9. Danksagung

Ich bedanke mich bei meinem Betreuungslehrer, Herrn StD Thomas Biedermann, für seine große Geduld und seinen großen Zeiteinsatz. Außerdem bedanke ich mich bei der gesamten Familie Biedermann, die mich an langen Forschungstagen ertragen und auch immer gut verpflegt hat. DANKE!!!

Anhang A1: Schaltplan