PIC-Grundlagen Teil 4

Dieser Teil der Artikelserie beschreibt die Erstellung eines Beispielprogramms bis hin zur Programmierung des PICs mit dem PICSTART-PLUS-Programmiergerät.

Entwicklungsumgebung

Die Installation der Windows-Entwicklungsumgebung und der Software für den PICSTART-PLUS wurde bereits im vorangegangenen Artikel ausführlich beschrieben.

Nach der erfolgreichen Installation befinden sich im Programm-Ordner unter Windows 95 und im „MPLAB"-Ordner unter Windows 3.x zwei Programmicons mit der Bezeichnung „MPLAB" und „MPASM for Windows".

Bei dem Programm „MPASM for Windows" handelt es sich um den Assembler, der einen Quellcode in Maschinencode umwandelt, der dann in den PIC programmiert werden kann.

Bei dem Programm „MPLAB" handelt es sich um die eigentliche Entwicklungsumgebung, mit der Quellcodes erstellt, simuliert, assembliert und letztendlich das PICSTART-PLUS-Programmiergerät angesteuert wird.

Die Entwicklungsumgebung wird durch einen Doppelklick auf das „MPLAB"-Icon gestartet.

Projekterstellung

Zur Erstellung eines neuen Projektes ist zuerst der gewünschte Prozessortyp einzu-

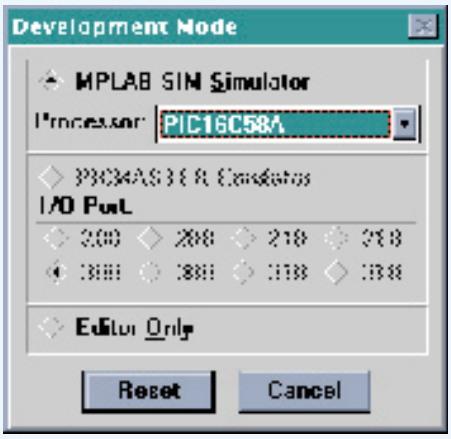

Bild 15: Ansicht des Fensters zur Auswahl des PIC-Typs

stellen, wozu im Menü „Options" der Punkt „Development Mode" auszuwählen ist, worauf das Fenster (Abbildung 15) erscheint. Im oberen Teil des Fensters ist dann der Prozessortyp aus einer Liste auszuwählen. Für unser Beispielprogramm ist

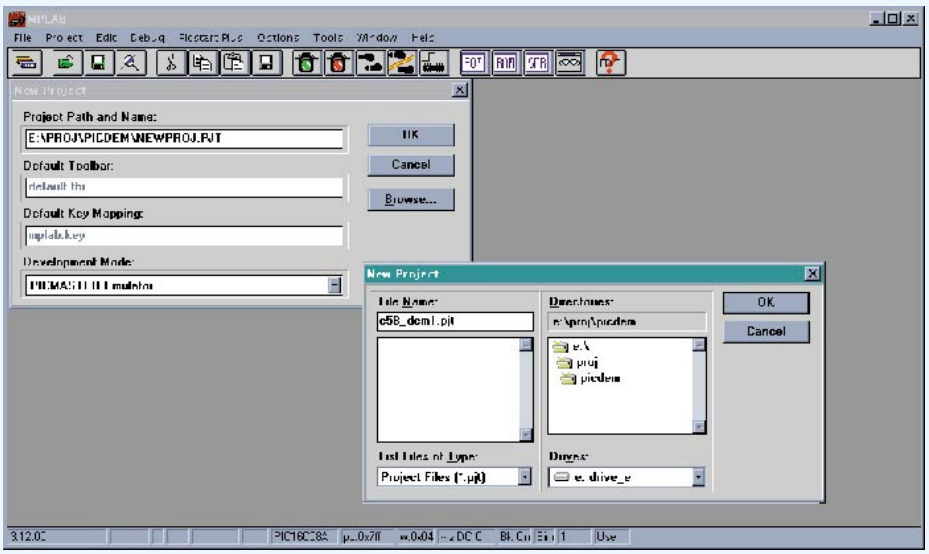

Bild 16: Ansicht des Fensters zur Erstellung eines neuen Projektes

hier der PIC16C58A zu wählen. In diesem Fenster erfolgen ebenfalls die Einstellungen für den eventuell angeschlossenen Emulator sowie die Einstellung, ob das Programm lediglich als Editor verwendet werden soll. Die Einstellung wird mit dem Button „Reset" verlassen, woraufhin alle Einstellungen für den ausgewählten Prozessortyp vorgenommen werden.

Im nächsten Arbeitsgang erfolgt die Erstellung eines Projektes, wozu der Menüpunkt „New Projekt" aus dem Menü „Projekt" auszuwählen ist. Daraufhin erscheint das Fenster, das in Abbildung 16 dargestellt ist. In der oberen Zeile des Fensters erscheint das aktuelle Verzeichnis und der Name des Projektes, der durch den Button „Browse" verändert werden kann.

In unserem Beispiel haben wir das Verzeichnis „e:\proj\picdem\" gewählt und das Projekt mit „c58_dem1.pjt" bezeichnet. Nach dem Bestätigen der Einstellungen erscheint das Fenster, das in Abbildung 17 dargestellt ist. Hier werden alle Quellcode-Dateien, die sich im angegebenen Verzeichnis befinden, auf der rechten Seite angezeigt. Die Dateien, die zum Projekt gehören, sind aus dieser Liste auszuwählen und über den Button „Add" auf die linke Seite zu kopieren. Da wir jedoch noch keine Quelldatei erstellt haben, sind beide Seiten leer, und das Fenster kann geschlossen werden.

Erstellung eines Programmlistings

Im nächsten Schritt wird die Quelldatei erzeugt, wozu im Menü „File" der Punkt „New Source" auszuwählen ist. Es öffnet sich ein Fenster, in dem das Beispiel-Programmlisting eingegeben werden kann (Abbildung 18).

Bei der Erstellung eines Programms sollte dabei nicht vergessen werden, die Datei in regelmäßigen Abständen abzuspeichern, damit bei einem eventuellen Absturz des Rechners die mühsam erstellen Daten nicht verloren gehen.

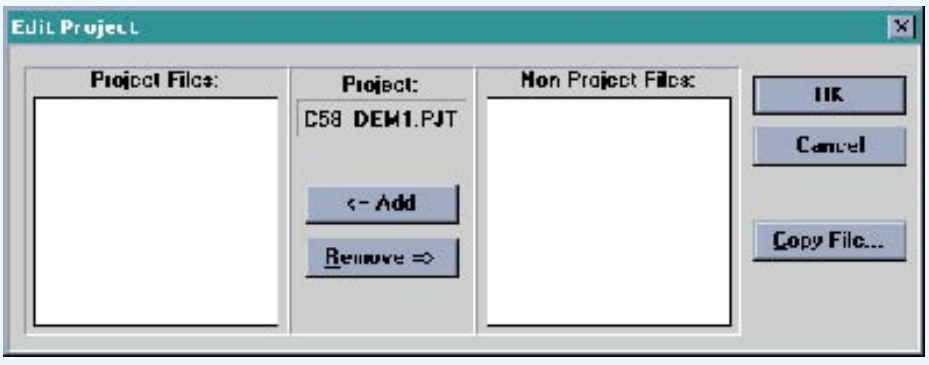

Bild 17: Auswahl der Quelldateien, die zum Projekt gehören

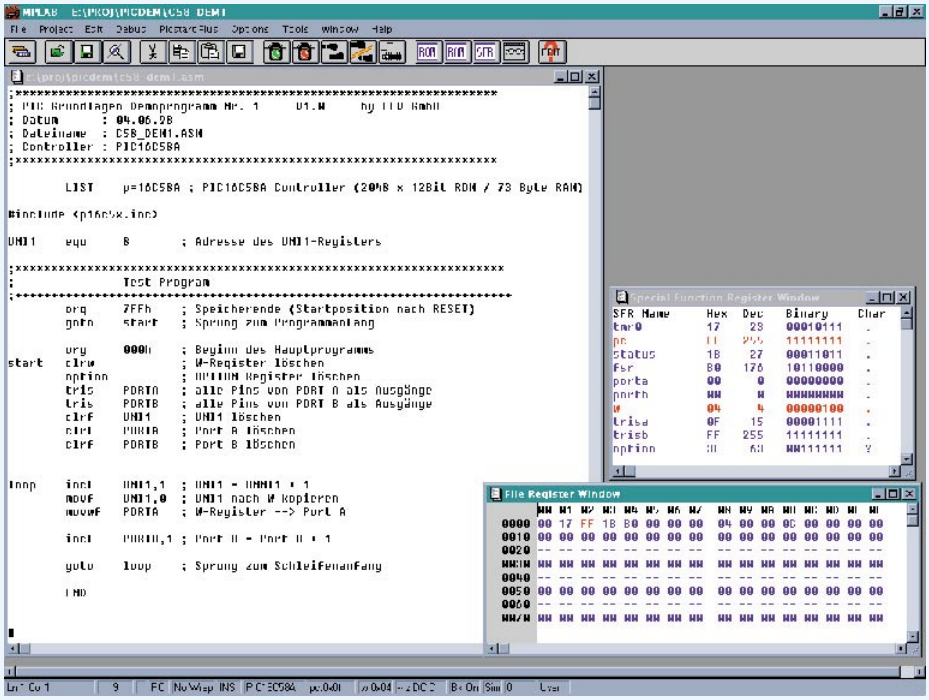

Bild 18: Ansicht des Listings und des Inhalts der File- und Funktions-Register

Befindet sich in einer Programmzeile ein Semikolon, so wird der darauffolgende Text bis zum Ende der Zeile nicht als Programmcode interpretiert. Auf diese Weise können Hinweise mit in den Programmcode eingebracht werden, indem zum Beispiel hinter einem Befehl ein Semikolon mit einer Beschreibung der Funktion erfolgt. So bilden auch die ersten 6 Zeilen des Beispielprogramms den Programmkopf, indem alle wichtigen Informationen zum Programm untergebracht sind.

Die Befehle, gefolgt von den Parametern, werden zeilenweise untereinander eingegeben, wobei darauf zu achten ist, daß kein Befehl an der 1. Stelle einer Zeile beginnen darf. Es empfiehlt sich, am Zeilenanfang einen Tabulatorsprung einzufügen, den Befehl einzugeben und nach einem weiteren Tabulatorsprung den Parameter einzugeben. Somit ergibt sich eine übersichtliche Anordnung der Befehle, die alle in einer Reihe untereinander stehen.

Um im Programm zu einer bestimmten Position zu springen und den Programmablauf an dieser Position fortzusetzen, werden Sprungbefehle eingesetzt. Dabei muß das Sprungziel in Form eines Labels angegeben werden. Als Label werden Begriffe bezeichnet, die an der 1. Position einer Befehlszeile beginnen und den Punkt im Programm bezeichnen. So wird zum Beispiel der Anfang der Programmschleife in unserem Beispiel mit „loop" bezeichnet, zu dem am Ende des Programms mit dem Befehl „goto loop" gesprungen wird.

Als Bezeichnung für die Label sollten Ausdrücke verwendet werden, die gleichzeitig die Funktion beschreiben, wodurch die Übersichtlichkeit im Programm erhöht wird.

In der 8. Zeile des Beispiels erfolgt die Übergabe von Parametern an den Assembler. Nach dem Wort „LIST" können verschiedene Parameter angegeben werden, wobei im Beispiel der Parameter „p=" den PIC-Typ angibt. Weitere Parameter sind in der Regel nicht notwendig, da der Assembler bereits mit Defaultwerten arbeitet.

In der 10. Zeile wird das Include-File zum aktuellen Mikrocontroller eingebunden. Hierbei handelt es sich um einen Quellcode, bei dem die einzelnen Registernamen den Registern zugeordnet werden. So kann zum Beispiel mit der Bezeichnung „PORTB" auf die Portpins RB0 bis RB7 zugegriffen werden, deren Register sich auf Adresse 06h befindet.

Ebenso können auch andere Register oder Konstanten einem Namen zugeordnet werden, um die Verständlichkeit des Programms zu erleichtern. Dies geschieht zum Beispiel in der 12. Zeile. Hier wird dem Namen „UNI1" der Wert 8 zugewiesen. Da zu der Zahl kein Zahlensystem angegeben ist, wird diese in hexadezimaler Schreibweise übernommen, d. h. der Wert 12 wird zum Beispiel als 12h (= dezimal 18) übernommen. Der Assembler kann über einen Parameter auch dazu veranlaßt werden, Zahlen zum Beispiel als Dezimalzahl zu verarbeiten. Es ist jedoch davon abzuraten, diese Einstellung zu ändern, da sonst ein Quellcode auf einem anderen Rechner mit anders konfiguriertem Assembler zu einem fehlerhaften Programmcode führt.

Liegt ein Wert in einem anderen Zahlensystem vor, so kann dieser auch direkt eingegeben werden durch Eingabe in einem bestimmten Format. So kann zum Beispiel die Dezimalzahl 123 als „D'123" eingegeben werden. Die verschiedenen Formate sind jeweils mit einem Beispiel in der Tabelle 4 zusammengefaßt.

In der 17. Zeile des Beispiels beginnt dann das eigentliche Programm. Wie bereits in den vorherigen Artikeln beschrieben, beginnen die PIC-Controller der PIC 16CX-Famile den Programmablauf nach dem Reset an der höchsten Adresse des Programmspeichers. An diesem Punkt muß somit zuerst ein Sprungbefehl zum Programmanfang stehen. Dies geschieht in den Zeilen 17 und 18, wobei mit „org 7ffh" die Adresse vorgegeben wird, ab der die nachfolgenden Befehle im Programmspeicher abgelegt werden. So wird der Sprungbefehl zum Programmanfang (Adresse „start") an der Adresse 7FFh abgelegt.

In der 20. Zeile folgt dann die Vorgabe, daß die folgenden Befehle ab dem Speicheranfang abgelegt werden sollen. Das Label „start" entspricht dabei der Adresse 000h.

In den Programmzeilen 21 bis 27 erfolgt dann die Konfiguration der I/O-Ports, des Timers und des Watch-Dogs. Zuerst wird das W-Register gelöscht und dessen Inhalt dann in das OPTION-Register kopiert, das zur Steuerung des Timers und Watch-Dogs dient. Alsdann werden die Portpins A und B als Ausgänge konfiguriert durch Beschreiben der Tristate-Register mit 0. Zum Ende der Konfiguration erfolgt die Initialisierung der Variablen UNI1 und der Ports A und B.

Die Programmschleife beginnt in der 30. Zeile, die mit dem Label „loop" bezeichnet ist und den Anfang der Programmschleife kennzeichnet. Hier wird das Register „UNI1" um eins erhöht und anschließend in das W-Regsiter kopiert, daß daraufhin auf dem Port A ausgegeben wird.

Da die Ports direkt im Adreßbereich des Speichers angeordnet sind, kann auf diese auch direkt mit den Registerbefehlen zugegriffen werden, welches in Zeile 34 erfolgt. Hier wird der Registerinhalt um eins erhöht, der gleichzeitig an den Pins RB0 bis RB7 erscheint.

In der 36. Zeile folgt dann der Sprungbefehl zum Anfang der Programmschleife.

Um das Ende einer Quelldatei zu kennzeichnen, ist in der letzten Zeile des Programms ein „END" einzutragen.

Nach dem Erstellen und Speichern der

Tabelle 4: Format für verschiedene Zahlensysteme

Zahlentyp	Syntax	Beispiel
dezimal	D'Zahl'	D'123'
hexadezimal	H'Zahl'	H'9f'
octal	O'Zahl'	O'777'
binär	B'Zahl'	B'00111001'
ASCII	'Zeichen'	'C'

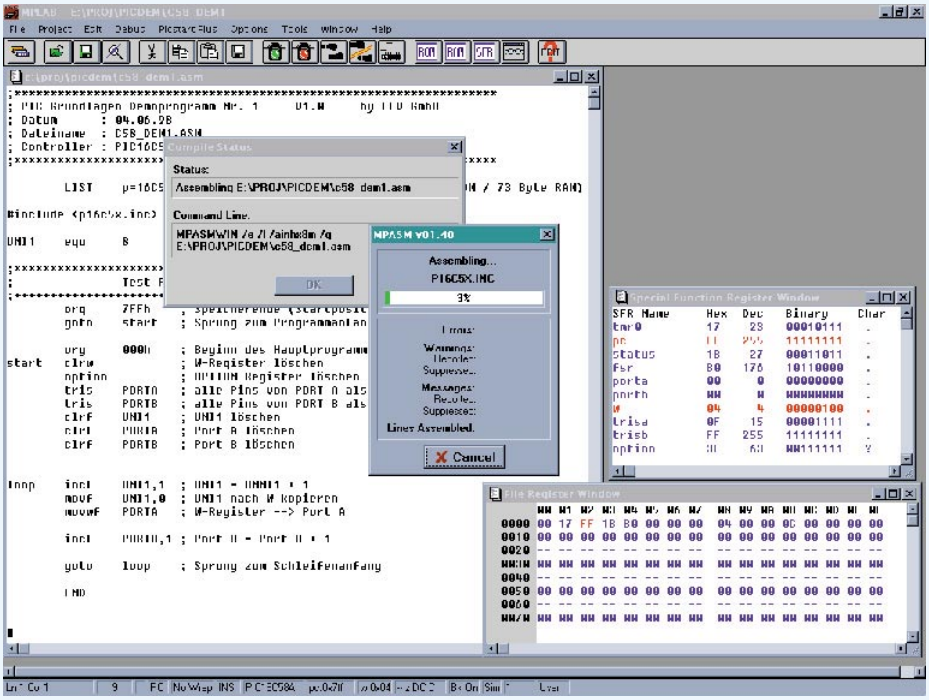

Bild 19: Aufruf des Assemblers

Quelldatei ist diese noch zum Projekt hinzuzufügen. Dazu wird der Punkt „Edit Projekt" im „Projekt"-Menü ausgewählt, woraufhin sich wieder das Fenster (Abbildung 17) öffnet. Die erstellte Datei erscheint nun auf der rechten Seite. Sie wird ausgewählt und mit dem Button „Add" in das linke Fenster kopiert.

Im nächste Schritt folgt das Assemblieren des erstellten Programms, welches durch den Punkt „Make Projekt" im „Projekt"-Menü erfolgt. Der Editor ruft dann automatisch den Assembler auf und übergibt ihm den Quellcode. Der Fortlauf der Übersetzung wird dabei auf dem Bildschirm angezeigt (siehe Abbildung 19). Treten bei der Übersetzung Fehler auf, so öffnet sich ein zusätzliches Fenster, in dem die Fehlermeldungen mit der entsprechenden Zeilennummer im Quellcode aufgelistet sind. Nachdem alle Fehler behoben sind, wird der Assembler erneut gestartet, bis dieser fehlerfrei durchläuft.

Bild 20: Buttons zur Simulation

Simulation

Um die Funktionsweise des erstellten Programms zu überprüfen, bietet die Entwicklungsumgebung eine komfortable Möglichkeit, den Programmablauf zu simulieren.

Um bei der Simulation den Inhalt der Register sowie die aktuellen Portzustände darzustellen, können im Menü „Window" die Punkte „File Register" und "Special Function Registers" ausgewählt werden. Es öffnen sich zwei Fenster, die auf eine freie Fläche des Bildschirms geschoben werden können.

Die Steuerung der Simulation erfolgt hauptsächlich durch die 5 Buttons, die in Abbildung 20 dargestellt sind. Durch den rechten Button (oder F6) wird das Programm zurückgesetzt, damit die Simulation am Programmanfang beginnen kann. Durch einen Druck auf den mittleren Button (oder F7) wird dann ein Befehl des Programms ausgeführt. Der aktuelle Zustand der File-Register und der Ports ist dabei in den Fenstern dargestellt. Die Register, die durch den letzten Befehl verändert wurden, sind in rot dargestellt.

Die aktuelle Position im Programmcode ist durch einen schwarzen Balken gekennzeichnet, der den nächsten auszuführenden Befehl zeigt.

PIC programmieren

Nachdem das Programm fehlerfrei übersetzt wurde, kann es in einen PIC programmiert werden. Dazu ist der PIC-Programmer zuerst an eine freie serielle Schnittstelle des PCs anzuschließen und mit dem Netzteil zu verbinden. Vor der ersten Inbetriebnahme des Programmers muß zuerst die serielle Schnittstelle eingestellt werden, welches unter „Programmer Options" im Menü „Options" erfolgt.

Das Programmiergerät muß zuerst initialisiert werden, indem „Enable Programmer" im „Picstart Plus"-Menü ausgewählt wird. Daraufhin öffnet sich das Konfigurationsfenster, das in Abbildung 21 dargestellt ist. Hier sind der Typ, der Oszillatormode und andere Einstellungen für den PIC vorzugeben.

Danach sind die verschiedenen Optionen für das Programmieren, Auslesen, Prüfen usw. im Menü „Picstart Plus" sichtbar (Abbildung 22). Durch die Auswahl der Option „Program/Verify" kann das aktuelle Projekt in den PIC programmiert werden.

Damit ist die Beschreibung zur Erstellung eines Programms für die PIC-Microcontroller abgeschlossen. Im nächsten Teil der Artikelserie wird eine kleine Testplatine vorgestellt, die alle Portpins des PICs auf einer Stiftleiste zur Verfügung stellt und den einfachen Test eines erstellten Programms ermöglicht. ELV

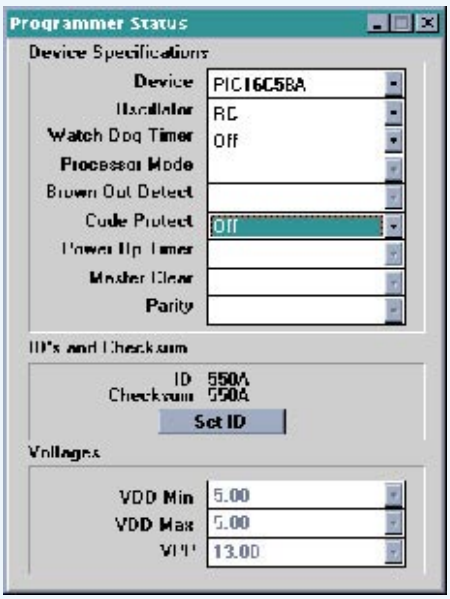

Bild 21: Konfigurationsfenster des PIC-Programmers

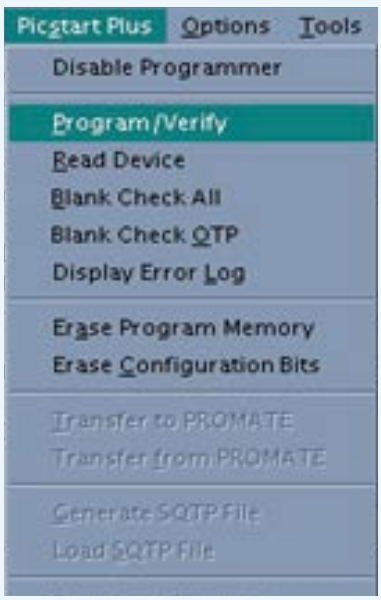

Bild 22: Funktionen des PIC-Programmers