Zusätzliche I/O Ports mit PCF8475P

Gunther Zielosko

1. Einführung

Jeder BASIC-Tiger-Anwender kennt das Problem – obwohl der kleine Steuercomputer eigentlich gut mit frei programmierbaren I/O-Pins bestückt ist, fehlen bei vielen Projekten zum Schluß ein paar logische Ein- oder Ausgänge. Das liegt einmal daran, daß eine Anzahl von I/O-Leitungen des im BASIC-Tiger eingebauten Mikroprozessors bereits für innere Aufgaben (Adress-, Daten- und Steuerbus) gebraucht werden und zum anderen daran, daß das Konzept des BASIC-Tigers je nach Anwendung schon einige Ports oder Einzelpins „konfisziert“ (Datenbus Port 6, Serielle Schnittstellen Port 9, Steuerleitungen Port 4 usw.). So muß man entweder mit den vorhandenen I/O-Pins haushalten oder sich zur Erweiterung etwas einfallen lassen. Wie könnten solche Erweiterungen aussehen?

Die erste Möglichkeit bietet die Firma Wilke selbst an, nämlich erweiterte Eingänge oder erweiterte Ausgänge (wie sie z.B. im Plug-and-Play-Lab vorhanden sind). Die Schaltung- und Programmiertechniken werden ausführlich im Handbuch beschrieben. Auf derselben Basis werden sogar fertige Erweiterungsbausteine mit verschiedenen Kombinationen oder Ausführungen zusätzlicher Ein- und Ausgänge angeboten. Nachteil dieser Lösung ist – die so gewonnenen Pins sind hard- und softwareseitig in ihrer Funktion festgelegt. Eingänge bleiben für alle Zeiten Eingänge und Ausgänge immer Ausgänge! Außerdem ist der Hardwareaufwand zur Gewinnung weniger zusätzlicher Pins erheblich (mindestens drei IC’s!), wenn eine größere Anzahl von Ports gebraucht wird, lohnt sich das schon eher.

Eine zweite Möglichkeit wäre, fertige parallele I/O-Schaltkreise (PIO) aus Mikroprozessor-Systemen zu verwenden, wie z.B. den Z80-PIO mit 2 kompletten Ports oder den 8255 mit sogar 3 Ports. Eine solche Lösung erfordert zwar einigen Anpassungsaufwand, hat aber auch Vorteile. Die Pins sind i.A. wirklich frei programmierbar und das Ganze geht sehr schnell. Allerdings sind beide PIO’s große 40-polige Bausteine und ihr Einsatz schon deshalb nicht überall möglich.

Wir wollen in diesem Applikationsbericht eine dritte Möglichkeit ausprobieren. Mit dem Schaltkreis PCF8574 wird ein programmierbarer 8-Bit-Bus zur Verfügung gestellt, der über nur 2 Leitungen (I^2C-Schnittstelle) gesteuert wird. Und das Schönste ist, diese beiden Leitungen können problemlos 7 weitere solche Bausteine programmieren, das ergibt die stattliche Anzahl von 64 neuen I/O’s. Nimmt man den PCF8574A dazu, einen „Bruder“ des PCF8574, werden es durch andere Adressierung sogar 128. Ein Nachteil soll nicht verschwiegen werden, wegen der seriellen Ansteuerung dauert es natürlich länger, bis der Baustein (oder gar alle 16) in die gewünschte Funktion versetzt werden. Jedesmal muß ihm ja „bitweise“ mitgeteilt werden, ob er Ein- oder Ausgang sein und was er z.B. als Ausgang für ein logisches Datenwort ausgeben soll. Dennoch wird diese Lösung für viele Anwendungen,

bei denen es nicht so sehr auf die Geschwindigkeit dieser zusätzlichen I/O-Pins ankommt, nützlich sein.

2. Der PCF8574 von Philips

Den PCF8574 gibt es in verschiedenen Gehäusevarianten und Ausführungen, die sich in der Programmierung unterscheiden. Ein Datenblatt bekommt man über:

http://www-us6.semiconductors.com/acrobat/datasheets/PCF8574_2.pdf

In unserem Applikationsbericht behandeln wir nur die 16-polige DIP-Bauform PCF8574P, es ist bei der Bestellung sehr genau darauf zu achten, daß nicht der PCF8574AP, der eine andere Adressierung braucht (hier passen die Beispielprogramme im Anhang nicht!) oder die 20-polige SSOP20-Bauform gewählt wird (hier stimmen die Pin-Bezeichnungen unserer Zeichnungen nicht). Allerdings bietet der PCF8574AP durch seine andere Typadresse bei sonst gleicher Funktion die Chance, noch einmal 64 I/O-Pins zusätzlich zu realisieren!

Zur Funktion des PCF8574 hier nur soviel:

Die Ausgänge sind gelatcht, d.h. sie behalten ihr Signal solange, bis ein neues Kommando erfolgt und sind so konzipiert, daß sie direkt LED's ansteuern können (leider nur im Low-Zustand des Ausganges).

Es gibt noch einen Interrupt-Ausgang, den wir aber hier nicht weiter beachten (offenlassen!).

Jeder Baustein hat 3 Adressleitungen, seine individuelle Adresse A0...A2, die in der Schaltung fest verdrahtet werden (siehe Bilder 1 und 2). Weitere intern 4 festgelegte Adressbits (Typ-Adresse PCF8574P 0100, PCF8574AP 0111) dienen zur Unterscheidung von anderen I^2C-Bausteintypen (z.B. EEPROM's), die vielleicht auf demselben I^2C-Bus arbeiten sollen.
Mit den I^2C-Befehlen wird dieser Baustein zunächst adressiert und dann programmiert. Durch das Schreib- und Lesekommando R/W werden schließlich Aus- oder Eingaben vorbereitet:

Schreiben → R/W = 0
Lesen → R/W = 1

Die Bitfolge der ersten Bits sieht dann so aus (Tabelle 1):

		Slave-Adresse des PCF8574									
	Start	**Typ-Adresse**				**Individuelle Adresse**			**R/W**	**Antwort**	**Daten →**
PCF8574	**S**	**0**	**1**	**0**	**0**	**A2**	**A1**	**A0**	**0/1**	**A**	
PCF8574A	**S**	**0**	**1**	**1**	**1**	**A2**	**A1**	**A0**	**0/1**	**A**	

Tab. 1 Die Slave-Adressierung von PCF8574 und PCF8574

Verbindet man also alle Adressen A0...A2 eines PCF8574 mit GDN, ergibt sich für diesen Baustein die Adresse 0100000x, also 40h bzw. 41h. (Schreiben bzw. Lesen). Ein zweiter Baustein könnte mit A0 an VCC, mit A1 und A2 an GND beschaltet werden, was die Adresse 42h bzw. 43h ergibt usw. In unseren Softwarebeispielen sind dies die Bausteine PCF8574a und PCF8574b. Die Startkondition, die Adressenbildung, die Schreib- Leseauswahl, die Abfrage der Bausteinantwort sowie die Datenübergabe werden von fertigen BASIC-Tiger-Befehlen organisiert.

Wie das im Einzelnen beim Schreiben und Lesen funktioniert, wird später behandelt.

Eine Besonderheit gibt es noch beim PCF8574, man kann mit einigen Tricks sowohl Eingaben als auch Ausgaben quasi bidirektional in einem 8-Bit-Port realisieren. Damit kommen die Eigenschaften der neuen I/O's nahe an die Fähigkeiten der echten Tiger-I/O-Pins, die das ja auch können.

3. Die Schaltung

Die hardwareseitige Kopplung von BASIC-Tiger und PCF8574 ist wirklich einfach. Bild 1 zeigt die Zusammenschaltung von BASIC-Tiger und bis zu 8 PCF8574. Genauso werden bei Bedarf PCF8574A hinzugefügt.

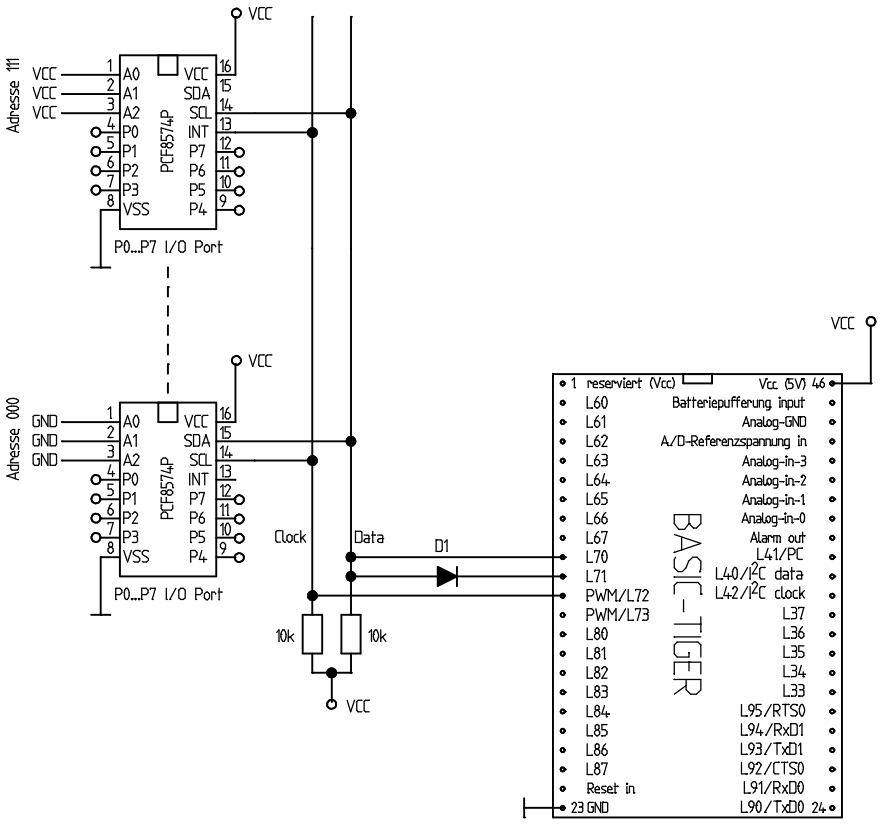

Bild 1 Anschaltung mehrerer PCF8574P an den BASIC-Tiger

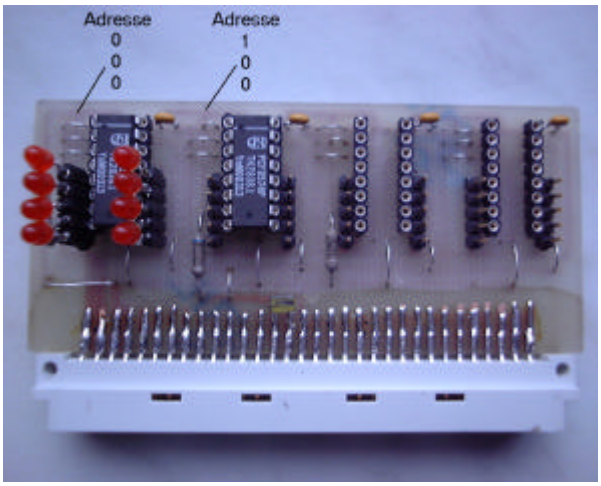

Bild 2 Laboraufbau mit 2 PCF8574P und 8 LED's gegen Vcc

In dem Laboraufbau nach Bild 2 wurden die individuellen Bausteinadressen 000 und 001 gewählt, die komplette Slave-Adresse ist also für Baustein 1 die 0100000x (40h) und für Baustein 2 die 0100001x (42h).
Zusätzlich zu den gezeigten Komponenten sollte man jedem PCF8574 mindestens einen 100 nF Abblock-Kondensator spendieren. Das ist alles, mit einer passenden Software werden wir zunächst Ausgänge, dann Eingänge und zum Schluß kombiniert Ein- und Ausgänge an einem PCF8574 realisieren.

4. Der BASIC-Tiger und seine I²C-Befehle

Die Befehle des BASIC-Tigers zum Schreiben und Lesen eines I²C-Bausteines werden im Handbuch nicht sehr ausführlich behandelt, zudem gibt es einige Widersprüche. Insbesondere mit dem I²C-Sonderling PCF8574 treten zusätzliche Probleme auf, die im folgenden etwas näher beleuchtet werden sollen.

Fangen wir mit dem Schreibbefehl **I2C_WRITE** an. Die in Klammern angefügten Parameter (Chip, Adresse, Anzahl, Variable) haben folgende Bedeutung:

Chip	ist der IC-Typ und seine individuelle Adresse, der angesprochen werden soll (übergeben wird in unserem Fall einfach die komplette Adresse, z.B. 40h)
Adresse	hier ist nicht der IC-Typ und seine Adresse gemeint, sondern eine innere Adresse, wie sie beispielsweise bei EEPROM's benötigt wird. Eine solche Adresse ist beim PCF8574 nicht vorhanden.
Anzahl	Anzahl der zu übergebenen Bytes (bis 31)
Variable	die zu schreibenden Bytes in Stringform

Was bei anderen I²C-Bausteinen sinnvoll funktioniert, macht beim PCF8574 Probleme. Der I2C_WRITE-Befehl des Tiger-BASIC besteht auf der Übergabe einer inneren Adresse, die

dieser IC gar nicht hat. Wenn man hier beispielsweise eine 0 einträgt, könnte ein Beispielprogramm so aussehen, wie es auch von Marcel Hendrickx in der Tiger-Mailing-Liste vorgestellt wurde:

```
#define PCF8574  40h
     TASK MAIN
       STRING A$

       I2C_SETUP  (  7,   0,    7,    1,     2,     0)
       A$ = "01010101"B
       I2C_WRITE ( PCF8574, 0, 1, A$)

     END
```

Es funktioniert sogar, hat aber einen Schönheitsfehler. Die scheinbar unwirksame innere Adresse „0“ wird vom PCF8574 als Datenbyte erkannt und ausgegeben. Erst dann erfolgt die Ausgabe der richtigen Datenbytes, d.h. vor jeder gewollten Ausgabe wird das Adreßbyte als kurzes Datenbyte ungewollt ausgegeben, eine Katastrophe z.B. für nachgeschaltete Flip-Flops. Danach kann man aber bis zu 31 Bytes als String „richtig“ übertragen.
Wilke Technology empfiehlt einen anderen Weg aus dem Dilemma, anstelle mit dem I2C_WRITE-Befehl soll mit dem I2C_READ-Befehl auf den PCF8574 „geschrieben“ werden. Die Bedeutung der in Klammern geschriebenen Parameter (Chip, Adresse, Anzahl, Variable) ist dieselbe wie beim I2C_WRITE-Befehl. Der vorgeschlagene Trick ist nun, mit dem I2C_READ-Befehl so zu tun, als wolle man lesen, die eigentlich wichtige Aktion ist aber das Einbringen der inneren Adresse. Dies bewirkt genau wie beim Schreiben eine Ausgabe dieses Adreßbytes auf die Ausgänge des PCF8574. Das „unnötige“ Adreßbyte ist also das auszugebende Byte, das mit dem I2C_READ-Befehl eingelesene Byte wird ignoriert.
Das funktioniert auch, hat aber den Nachteil, daß nicht viele Bytes in einem Zuge wie mit dem I2C_WRITE-Befehl übertragen werden können, sondern immer nur eins. Trotzdem wurde dieser Weg im Beispielprogramm **8574_OUT.TIG** gewählt.

Auch nicht ganz ohne Klippen ist das eigentliche Lesen mit dem Befehl **I2C_READ**. Wie wir schon beim „Schreiben“ mit diesem Befehl gesehen haben, erfolgt auch beim Lesen immer eine ungewollte kurze Ausgabe der (inneren) Adresse an die Portpins des PCF8574. Dies bedeutet für angeschlossene Datenlieferanten (z.B. andere IC’s) eine unschöne Vergewaltigung, wenn die Pegel entgegengesetzt liegen. Nun gibt es einen Umstand beim PCF8574, den wir ausnutzen können. Der Low-Pegel bei einem als Ausgang geschalteten Pin ist sehr kräftig, er kann sogar, wie schon erwähnt, LED’s ohne Treiber direkt ansteuern. Der High-Pegel dagegen ist sehr „hochohmig“, er kann von jedem äußerem Logikgatter einfach überschrieben werden. Das kommt uns sehr entgegen, wir müssen die Ausgänge des PCF8574 vor jeder Eingabe lediglich auf High schalten, damit die von außen angelegten Pegel unverfälscht bleiben. Damit ist klar, welche (innere) Adresse wir im Befehl I2C_READ beim Einlesen eingeben müssen, nämlich 255. Damit werden bei der kurzen ungewollten Ausgabe dieser Adresse alle Ausgänge auf High und damit relativ hochohmig geschaltet. Das Programm **8574_IN.TIG** geht diesen Weg und demonstriert die Funktion von 8 neuen Inputs.

Jetzt sehen wir auch bereits die Lösung unseres letzten Problems, der Realisierung einer bitweisen Wahl von Ein- und Ausgängen am PCF8574. Alle Bits werden mit der (inneren) Adresse des Befehls I2C_READ auf High maskiert, wenn sie Inputs sein sollen. Dann werden die äußerlich angelegten Daten unverfälscht „hereingelassen“. Die als Output verwendeten Bits werden entweder ebenfalls auf High gesetzt, wenn sie high sein sollen oder auf Low, wenn sie low sein sollen. Das Programm **8574_MIX.TIG** gibt an den vier niederwertigen Bits eine Zählvariable aus und liest gleichzeitig an den höherwertigen Bits Daten ein.

Die Funktion der Ausgänge kann, wie schon erwähnt, ganz einfach mit Leuchtdioden überprüft werden. Schalten Sie einfach LED‘s mit ihrer Anode auf VCC, mit ihrer Kathode auf den jeweiligen Ausgang. Sie werden dann bei Low (0) am Ausgang leuchten und bei High (1) verlöschen. Beachten Sie aber bitte, daß mit LED’s als Indikator die Low-Spannung am Ausgang nicht nahe 0V liegt, sondern etwa 1,5V beträgt (Brennspannung der LED’s) und dadurch beim Lesen ein High „vorgespiegelt“ wird. Beim Programm 8574_MIX.TIG wird dann das Einlesen verfälscht, nach dem Abklemmen der LED’s werden echte Werte eingelesen.
Sind die Ports Eingänge, können sie mit Brücken zu VCC oder Masse auf High oder Low getestet werden, besser und sicherer sind Widerstände (z.B. 1kΩ). Genauso können natürlich Daten von anderen IC’s eingelesen werden, so z.B. vom zweiten Baustein PCF8574b...

Eigentlich ist der PCF8574 ein idealer Kandidat für viele Lösungen mit dem BASIC-Tiger, bei denen die I/O-Ports knapp werden. Trotz virtuoser Programmier-Tricks sind die gezeigten Lösungen nicht sehr elegant und passen nur notdürftig zu dem vielseitigen Baustein.
Der sauberste Weg ist eine eigene I^2C-Befehlsgruppe für den PCF8574 ohne die dargestellten Kompromisse, das könnte aber nur von Wilke Technology kommen. Ein Traum wäre es, wenn Wilke Technology 8 oder sogar 16 IC’s (PCF8574 und PCF8574A) in ihrer kleinsten Bauform in einen Erweiterungsbaustein stecken würde und mit leicht modifizierten IN-, OUT-, und DIR-Befehlen bis zu 128 echte zusätzliche I/O’s zur Verfügung stellen könnte...

(bitte in die Wunschliste für die BASIC-Tiger-Fans aufnehmen!)