Die BASIC-Tiger®-PIO

Gunther Zielosko, Heiko Grimm

1. Grundlagen

Sie kennen das Problem – Sie wollen mit dem BASIC-Tiger® ein Gerät bauen, das leider wieder einmal mehr I/O-Pins benötigt, als dieser zur Verfügung stellen kann. Also muß ein Erweiterungsbaustein her, es gibt ja im Sortiment der Firma Wilke Technology eine ganze Reihe davon mit vielen zusätzlichen Ausgängen, Eingängen und auch kombiniert Eingänge und Ausgänge in einem Baustein. Nur – diese Bausteine sind bezüglich der Anwendung der I/O-Pins festgelegt, ganz im Gegensatz zu den „echten“ I/Os des BASIC-Tigers®. Dort können Sie wählen, sowohl die Befehle IN und OUT als auch die Befehle DIR_PIN und DIR_PORT erlauben die variable Nutzung der Port-Anschlüsse. Was halten Sie davon, wenn wir mit Hilfe eines Xilinx CoolRunner®-Bausteines erweiterte Ports (EPorts) zur Verfügung stellen, die mehr oder weniger wahlfrei als Eingänge oder Ausgänge (pin-, tetraden- und portweise) benutzt werden können. Im folgenden Applikationsbericht werden wir ein solches Projekt realisieren, wobei wir zunächst 2 neue Ports anbieten wollen. Wählt man einen größeren CoolRunner®-Baustein aus, können erheblich mehr I/O-Ports bereitgestellt werden, das ist aber wegen der sehr komplexen Gehäuse und der damit notwendigen professionellen Leiterplattenverdrahtung kein Objekt mehr für den Eigenbau, hier sind industrielle Lösungen erforderlich.

2. Realisierung eines „kleinen“ Modells mit XCR3064XL-10 PC44

Wie schon erwähnt, sind die größeren CoolRunner®-Bausteine für Laboruntersuchungen sehr schwer zu handhaben. Deshalb hier ein Lösungsvorschlag mit Modellcharakter für 2 Ports zu je 8 Bits. Wegen der begrenzten Register- und Logikkapazität dieses kleinen Bausteines mußten auch Einschränkungen in der Programmierfreiheit hingenommen werden. Mit größeren Bausteinen lassen sich z.B. 8 völlig frei programmierbare Ports zu je 8 Bits realisieren. Jetzt stehen uns ein Port, dessen Pins bitweise, und ein Port, dessen Pins nur tetradenweise (also gemeinsam Bits 0...3 und Bits 4...7) als Ein- oder Ausgänge programmiert werden können, zur Verfügung. Dies ist bei Standard-PIO-Bausteinen ebenfalls üblich und stellt keine dramatische Einschränkung dar.
Die Adressierung erfolgt mit den Befehlen XIN und XOUT ähnlich dem EPort-System des BASIC-Tigers®, das wir auch bei früheren CoolRunner®-Experimenten schon genutzt haben.

2.1. Hardware

Die Ankopplung des XCR3064XL an den BASIC-Tiger® ist denkbar einfach. Wir brauchen lediglich die 8 Pins des Port 6 (die üblichen Datenleitungen für Erweiterungsbausteine) sowie drei Signale ACLK, DCLK und INE. Diese Signale sind aber diesmal nicht die üblichen Pins

L33, L34 und L35, sondern die Pins L80, L81 und L82. Dies ist wegen sonst möglicher Konflikte z.B. mit der Tastatur notwendig. Bild 1 zeigt die Anschlußbelegung unseres XCR3064XL und Bild 2 dessen Anbindung an den BASIC-Tiger®.

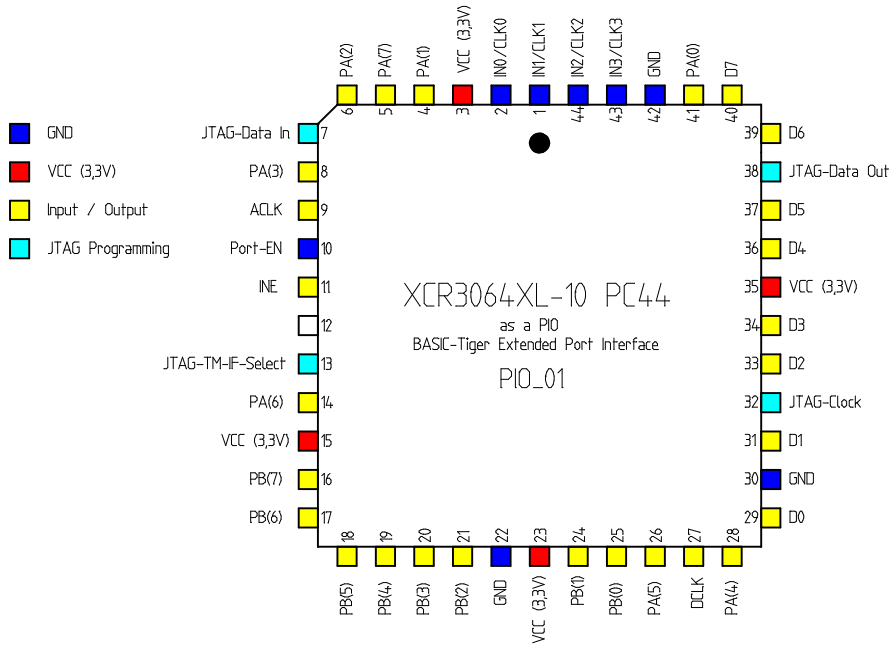

Bild 1 Pinbelegung der Modell-PIO

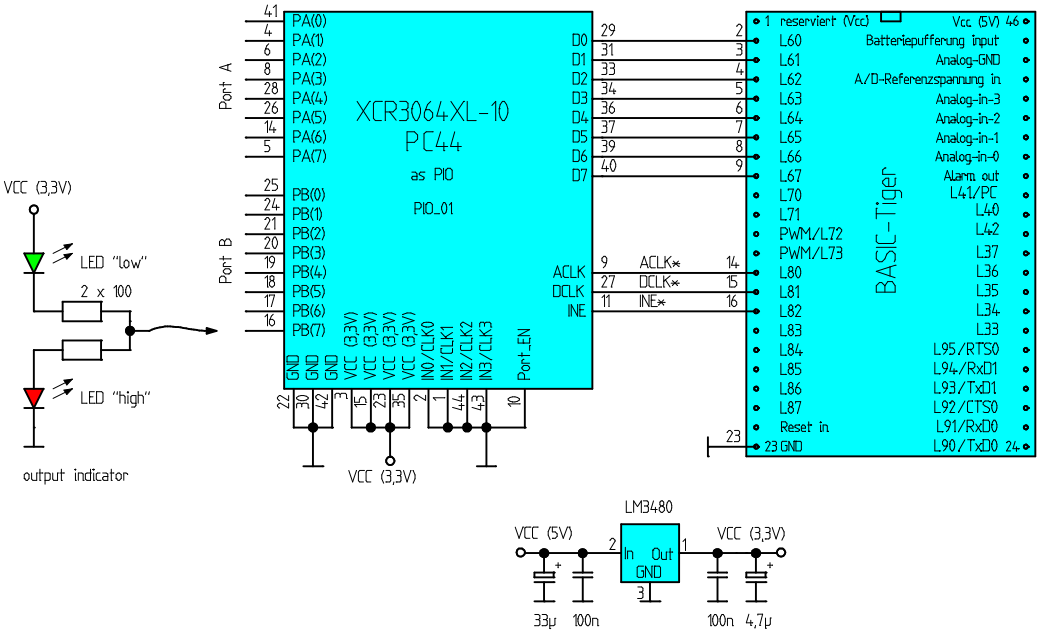

Bild 2 Anschaltung an den BASIC-Tiger® mit einfachem Logik-Indikator für Ausgänge

Denken Sie daran, daß auch hier wieder alle VCC-Anschlüsse des CoolRunners® an 3,3 V zu legen sind und jeder dieser Anschlüsse einen eigenen kurz angelöteten 100 nF – Abblockkondensator gegen Masse braucht.

Beachten Sie auch, daß Ausgänge des CoolRunner®-Systems nur Pegel bis 3,3 V liefern können. Werden die Ausgänge nur gering belastet, sollte dies kein Problem für die gängigen Logiktypen, wie LS, HCT oder HC sein.

Zur Kontrolle der Ausgänge des fertigen Moduls zeigen wir links im Schaltbild einen einfachen Indikator, der sich auch für andere Logikausgänge eignet. Liegt kein logisches Signal an (Pin ist Eingang oder hochohmig), leuchten beide Dioden. Wird logisch eine „1“ (high) ausgegeben, leuchtet die rote, wird logisch „0“ (low) ausgegeben, die grüne LED allein. Da die LED’s in diesem Fall allein vom Ausgang getrieben werden, kann man automatisch auch dessen Treiberfähigkeit abschätzen. Der CoolRunner® leistet da einiges...

2.2. Adressierung

Bevor wir zur BASIC-Tiger®-Programmierung kommen, zunächst ein paar Worte zur Adressierung des PIO-Bausteines. Wir haben zwei Ports, die natürlich jeweils ihre eigene Adresse haben müssen. Im Gegensatz zu den bisherigen Erweiterungsbausteinen des BASIC-Tiger®-Systems wollen wir mit derselben Adresse Inputs bzw. Outputs ansprechen. Im EPort-System des BASIC-Tigers® ist solch ein Fall nicht vorgesehen, es gibt einen Adreßraum für erweiterte Outputs und einen anderen für erweiterte Inputs. Aus diesem Grunde verwenden wir die relativ neuen Befehle XOUT und XIN, bei denen es diese Beschränkung nicht gibt. Außerdem kollidieren unsere Aus- und Eingaben nicht mit denen der (leider!) unvollständig auskodierten Adressen der Tastatur, der DIL-Switches und der erweiterten Ports. Voraussetzung für einen sicheren Betrieb ist, daß wir auch unabhängig von den drei Standard-Steuersignalen ACLK (L33), DCLK (L34) und INE (L35) werden. Der Befehl XSETUP ermöglicht die weitgehend freie Verwendung normaler I/O-Leitungen des BASIC-Tigers® zu diesem Zweck, wir haben:

L80 zum ACLK*-Signal,
L81 zum DCLK*-Signal und
L82 zum INE*-Signal gemacht (* steht dafür, daß die Original-Signale für Tastatur usw. weiterhin unabhängig existieren. Für unsere PIO haben wir damit eigene Signale, die durch andere Zugriffe nicht gestört werden)

Für die Festlegung der Datenrichtung unserer beiden neuen PIO-Ports benötigen wir weitere zwei Adressen. Obwohl wir nur zwei Ports adressieren müssen, haben die Adressen der Ports und der dazugehörigen Richtungsregister mit einem Abstand von 8 Adressen festgelegt, um frei für spätere Entwicklungen mit mehr I/O-Ports zu bleiben. Der Adressraum sieht damit wie so aus, wie es die Tabelle 1 zeigt. Wie man sieht, gibt es keinen Unterschied mehr zwischen physikalischen und logischen Adressen, wie das bisher Praxis bei den EPorts war.

Die Adressen werden allein aus den Daten des Ports 6 gebildet, einen Offset gibt es nicht mehr.

Nach dem Power On Reset sind alle Pins der Ports A und B Eingänge. Alle Ausgaben müssen so programmiert werden, daß mit einem XOUT-Befehl zunächst die Richtung einmal festgelegt wird. In weiteren XOUT- oder XIN-Befehl werden dann die Aus- bzw- Eingaben durchgeführt. Dabei ist es möglich, per XIN-Befehl auch die Zustände der als Ausgang programmierten Pins „zurückzulesen“.

Physikalische Adresse	Register-Adresse im XCR3064XL PC44 (PIO_01)	Funktion	Kommentar
08h	08h	Adresse Port A	OUT oder IN
09h	09h	Adresse Port B	OUT oder IN
0Ah	0Ah		Reserviert
0Bh	0Bh		Reserviert
0Ch	0Ch		Reserviert
0Dh	0Dh		Reserviert
0Eh	0Eh		Reserviert
0Fh	0Fh		Reserviert
10h	10h	Richtung Port A	1 Input, 0 Output bitweise*
11h	11h	Richtung Port B	1 Input, 0 Output tetradenweise**
12h	12h		Reserviert
13h	13h		Reserviert
14h	14h		Reserviert
15h	15h		Reserviert
16h	16h		Reserviert
17h	17h		Reserviert

* `10100101` Bits 0, 2, 5 und 7 sind Eingänge
Bits 1, 3, 4 und 6 sind Ausgänge

** `XXXXXX01` Bits 0, 1, 2 und 3 (untere Tetrade) sind Eingänge
Bits 4, 5, 6 und 7 (obere Tetrade) sind Ausgänge

Tabelle 1 Adressierung der PIO

2.3. BASIC-Tiger®-Programme PIO_01.TIG und PIO_02.TIG

Das BASIC-Tiger®-Programm PIO_01.TIG demonstriert die neuen Möglichkeiten mit den 16 zusätzlichen wahlfreien I/O-Pins. Ist die notwendige Hardware vorhanden, können wir sofort loslegen.
Zunächst wollen wir einen kleinen Ausflug in die Welt der Befehlsgruppe XOUT,XIN usw. machen, die mit der Version 5.0 des Tiger-Basic neu hinzugekommen sind, im Handbuch aber (leider) nicht sehr gut kommentiert wurden. Diese Befehle sind schneller, können ganze Gruppen von Ein- und Ausgaben erledigen und sind außerdem sehr flexibel einzusetzen. Hier brauchen wir zunächst die Befehle XOUT, XIN und XSETUP. Fangen wir hinten an,

XSETUP erlaubt uns die Auswahl verschiedener Pin-Kombinationen zum Ansteuern externer Baugruppen. Dabei kann man sowohl die ursprünglichen Pins für externe Bausteine benutzen oder, wenn dies durch schlechte Auskodierung und störende Zugriffe bereits vorhandener Komponenten unmöglich ist, ganz neue. In unserem Beispielprogramm benutzen wir folgendes:

FLAG=XSETUP(6,8,0,1,2,4,0)

Port 6 ist der Datenbus	wie bei Tastatur, Display, Erweiterungsbausteinen usw.
Port 8 ist der Steuerbus	bei den bisherigen Komponenten ist dies Port 3!
L80 ist ALCL	bei den bisherigen Komponenten L33
L81 ist DCLK	bei den bisherigen Komponenten L34
L82 ist INE	bei den bisherigen Komponenten L35
Port 4 ist Steuerbus 2	muß zwar mit eingegeben werden, wird aber hier nicht gebraucht
L40 ist CE-Pin	muß zwar mit eingegeben werden, wird aber hier nicht gebraucht

Vor der Verwendung von XSETUP müssen alle benötigten Pins des **neuen** Steuerbus mit DIR_PIN- oder DIR_PORT-Befehlen als Ausgang definiert werden. Das ist noch nicht alles, die Ruhepegel der benötigten Steuersignale müssen auch vorher definiert werden – mit einem normalen OUT-Befehl setzt man die Bits der neuen Steuerpins so, wie sie in Ruhe sein sollen. Das ist anders, als bei den gleichnamigen Signalen ACLK, DCLK und INE, die nicht frei eingestellt werden können! Eigentlich eine gute Sache, man kann damit sehr flexibel auf die unterschiedlichste Hardware reagieren. Schlecht ist, wenn das nirgends steht! Unser PIO-Baustein ist so programmiert, daß

ACLK*	high-aktiv,	(Ruhepegel low)
DCLK*	high-aktiv,	(Ruhepegel low)
INE*	low-aktiv ist.	(Ruhepegel high)

Ist das geschafft, geht es mit der Programmierung weiter. Wir brauchen drei Arten von Befehlen, deren Eigenarten im Folgenden etwas näher beschrieben werden sollen.

Der Befehl zur Definition der Datenrichtung (anstelle DIR_PORT bzw. DIR_PIN):

Grundsätzlich gilt: 1 = Eingang
0 = Ausgang

Beispiel:

XOUT (10h, 10100101b) am erste 8-Bit Port (08h) wird
Bit 0 Eingang
Bit 1 Ausgang
Bit 2 Eingang
Bit 3 Ausgang

Bit 4 Ausgang
Bit 5 Eingang
Bit 6 Ausgang
Bit 7 Eingang

Bitte beachten!

- Der Richtungsbefehl hat immer eine um 8 höhere Adresse als der zugehörige Port.
- Für die vorliegende Modell-PIO wird der zweite Port tetradenweise in der Richtung umgeschaltet. Dazu dienen die beiden niederwertigsten Bits (Bit 0 = niederwertige Tetrade, Bit 1 = höherwertige Tetrade)

Der Befehl zur Datenausgabe (wie das normale XOUT):

Beispiel:

XOUT (08h, 10011101b) Das Datenbyte 10011101 wird am ersten Port 08h ausgegeben Nur die vorher als Output definierten Pins geben eine Information aus.

Der Befehl zum Dateneinlesen (wie das normale XIN):

Beispiel:

E = XIN (08h) Der Port 08h wird in die Variable E eingelesen

Dabei gibt es folgende Möglichkeiten:

Der Port wurde zuvor vollständig als Eingang definiert (XOUT 10h, 11111111b)

Die Daten aller Pins des Ports 08h werden so eingelesen, wie die Information außen anliegt.

Nur einzelne Bits wurden als Eingang definiert (XOUT 10h, 10011101b)

Die Daten der als Eingänge definierten Pins (1) des Ports 08h werden so eingelesen, wie die Information außen anliegt.

Die Daten der als Ausgänge definierten Pins (0) werden so eingelesen, wie es ein vorheriger OUT-Befehl an diesen Ausgängen festgelegt hat (Wiedereinlesen einer früheren OUT-Anweisung möglich!)

Damit steht einem BASIC-Tiger®-Programm nichts mehr im Wege. Das kleine Programm PIO_01.TIG demonstriert einige Möglichkeiten der wahlfreien Ports. Es erklärt sich weitgehend selbst.

Bild 3 zeigt einen „Screenshot“ aus dem Ablauf des Programms.

Bild 3 PIO_01.TIG in Aktion

Oben wird die Maske M angezeigt (auf Adressen 10h, 11h für die Richtungsbestimmung der Pins), darunter das gerade ausgegebene Datenwort D (Ausgabe auf die Adressen 08h, 09h), darunter die Daten O, die real an den als Ausgang bestimmten Pins ausgegeben werden und ganz unten die Werte I, die an den als Eingang festgelegten Pins eingelesen werden. Dabei ist die linke Gruppe der PIO-Kanal A und die rechte Gruppe der PIO-Kanal B. Wie man sieht, erfolgt die Auswahl der Richtung in Port A pinweise und bei Port B tetradenweise. Im Programm wurde die Richtungsauswahl M fest eingestellt, die Ausgabedaten werden langsam hochgezählt.

Das zweite Programm PIO_02.TIG macht genau dasselbe, nur werden hier in der letzten Zeile alle Bits beider Ports angezeigt, das demonstriert die Möglichkeit des Wiedereinlesens von vorhergegangenen Output-Daten in ein Eingabe-Byte.

3. Ausblick

Wir haben einen kleinen CoolRunner®-Baustein von Xilinx zu einem universellen PIO-Baustein für 2 Ports mit je 8 Bit gemacht. Das vorgestellte System ist aber wesentlich universeller. Ziel dieses modellhaften Experimentes ist es, zukünftig PIO-Bausteine mit z.B. 8 Ports zu je 8 Bit zur Verfügung zu stellen, die sich in das System der Erweiterungsbausteine für den BASIC-Tiger® der Firma Wilke Technology einfügen.
Solche Bausteine benötigen dann größere CoolRunner®-Bausteine und vor allem eine komplexe Leiterplatte sowie ein entsprechendes Gehäuse.
Dann wäre eine allgemeingültige Adressierung im verfügbaren Adressraum des EPort-Systems wünschenswert. Die bisherige Adressierung des EPort-Systems ist wegen der begrenzten Möglichkeiten der diskreten Adreßbildung mit 1-aus-8-Dekodern 74HC138 unvollständig und bildet daher Fehlerquellen. Beim CoolRunner®-System besteht dieses Problem nicht, hier kann eine vollständige Adreßdekodierung erfolgen. Mit geeigneter Adreßwahl können so z.B. Konflikte mit der Adressierung der EPorts des Plug-and-Play-Labs vermieden werden. Ebenso lassen sich die allgemeinen Steuerleitungen ACLK, DCLK und INE für die neuen Bausteine nur eingeschränkt verwenden.
Sehr interessant wären neue Befehle zur Richtungsauswahl (wie DIR_PORT und DIR_PIN) analog zu den Ports des BASIC-Tigers® auch für die neuen Erweiterungsbausteine (vielleicht XDIR_PORT und XDIR_PIN). Dann könnten für die Richtungswahl dieselben Adressen verwendet werden wie für die eigentlichen OUT- oder IN-Befehle.

Für eigene Experimente fügen wir die Datei PIO_01.JED bei, die CoolRunner®-Experten sofort zum „Brennen“ der kleinen PIO benutzen können. Für die komplexere Zielvariante müssen wir auf eine entsprechende Entwicklung der Firma Wilke Technology warten...

Wünschen wir uns viel Erfolg!