Nachnutzung

Drucken

Details
Geschrieben von Ralf Kästner
Hauptkategorie: Projekte
Kategorie: KCNET
Veröffentlicht: 28. August 2008
Zugriffe: 2228

Auch für Nicht-KC85-Anwender ist eine Nachnutzung des Projektes interessant. Sowohl das Hardware-Interface als auch die Software sollte mit jedem CP/M 2.x kompatiblen Z80-System problemlos funktionieren. Was man dafür tun muss, kann hier nachgelesen werden.

Nach der Offenlegung der innersten "Geheimnisse" des KCNET im letzten Artikel soll es im folgenden Beitrag um die Nachnutzung und die notwendigen Voraussetzungen dafür gehen.

Die KC85 Anwender werden es dort logischerweise am einfachsten haben. Hier ist ein Hardwaremodul als Bausatz geplant, welches man natürlich aufbauen muss. Anschliessend ist man aber fein raus, in einen Modulschacht stecken, KC-Programm laden und loslegen - zumindest im idealen Fall, wenn alles sofort funktioniert.

Alle anderen Interessenten müssen sich für ihr System selbst Gedanken machen. Auch die Anpassung der KC-Software ist ja normalerweise notwendig. Die bisherigen Quellen liegen komplett als Assemblerquelltexte für den Microsoft-Assembler M80 vor. Damit sollte man, zur Not auch im Emulator auf einem PC oder ähnlich, umgehen können.

Ich kann dort zwar begrenzt behilflich sein, grössere Aktionen kommen aber von meiner Seite nicht in Frage, da ich ohne die Hardware der Zielplattform nichts testen kann und das wird spätestens dann zum Problem, wenn man alles aufgebaut und angeschlossen hat und nichts funktioniert.

Ich will an dieser Stelle lediglich davor warnen, etwas anzufangen, was man nicht beherrscht. Der Nachbau ist durch den vergleichsweise sehr geringen Hardwareaufwand, die Verwendung eines Z80 Standard Schaltkreises als Schnittstelle und den eingebauten Interface-Debugger ziemlich sicher und eine Fehlersuche im Problemfall relativ einfach.

Hardware

Die notwendige Hardware setzt sich aus den beiden Grundbestandteilen Z80 PIO-Schnittstelle und Interface-Hardware zusammen.

Der Schaltplan für die Interface-Hardware befindet sich im Downloadbereich, ebenso wie die Stückliste. Das muss man auf einer geeigneten Leiterplatte aufbauen. Leider hat das Netzwerkmodul Stiftleisten im 2 mm Raster, was nicht kompatibel mit den 2,54 mm einer Lochrasterplatine ist. Eine Möglichkeit besteht zum Beispiel darin, geeignete Aussparungen für 2 mm Buchsen auszufräsen, die Buchsen dort einzukleben und dann per Fädeldraht oder auch Lötverbindung in die Schaltung zu gehen.

Alle Bauteile, welche an den Pins 10 (RxD) und 11 (TxD) des ATmega angeschlossen sind, kann man mutigerweise komplett weglassen. Das ist der Pegelumsetzer für die RS232 Schnittstelle eines PC o.ä. Dort bekommt man Zugang zum Debugger der Interface-Firmware, welchen man nur benötigt, wenn die Inbetriebnahme scheitert. Für die normale Nutzung als Netzwerkschnittstelle ist dieser Schaltungsbestandteil nicht notwendig.

Wenn man das Interface in sein System integriert, hat man auch die 5V Betriebsspannung bereits zur Verfügung. Dann können BU1, D1-D4, C1, C2 und IC7 noch weggelassen werden - der 3,3V Spannungsregler IC6 (Pin 3) muss dann natürlich auch an 5V angeschlossen werden.

Das Netzwerkmodul kann in Deutschland beispielsweise über www.dacomwest.de oder www.watterott.com bezogen werden. Alle anderen Bauteile bekommt man bei den gängigen Elektronik-Händlern. Der Schaltplan an sich ist getestet bis auf die RESET-Schaltung des Controllers. Im Easy-TCP war dort ein ZSM560C im Einsatz, welcher nicht mehr verfügbar ist.

Im Prinzip kann man dort auch andere IC's einsetzen oder den RESET mit einem RC-Glied sicherstellen. Wir hatten mit einer einfachen RC-Schaltung aber im KC schon einmal Probleme, welche erst nach dem Einbau eines richtigen RESET-Generators beseitigt waren. Das ist also unbedingt zu empfehlen, bisher hatte ich mit dem ZSM560C noch nicht einen "Hänger" beim Einschalten des Systems. So wie der Plan im Moment vorliegt, wird er auch in das KC-Modul übernommen werden.

Zum zweiten Teil - der PIO im Z80 Rechner - kann ich nicht viel sagen, da das vom eigenen System abhängt. Auf alle Fälle benötigt man aber die komplette PIO mit beiden Ports und allen 4 Handshakeleitungen ARDY, /ASTROBE und BRDY, /BSTROBE, da die bidirektionale Betriebsart der PIO von der Software genutzt wird.

Ab Version 1.2 des Interface lässt sich die PIO auch im Polling betreiben, man benötigt also den vorher unbedingt erforderlichen Interrupt Mode IM2 des Z80 nicht mehr zwingend, es könnte für später aber von Vorteil sein, wenn man die PIO im IM2 auch betreiben kann. Im Moment läuft die komplette Software ausschliesslich im Polling.

Wenn man über die Schaltpläne zum System verfügt, kann man u.U. den Anschluss einer bereits vorhandenen PIO einfach übernehmen, man muss natürlich per Adressdekoder für eine andere I/O-Adresse bei der neuen PIO sorgen, wie gesagt, ist so etwas immer vom vorhandenen Z80 System abhängig, mit dem man sich ein wenig auskennen sollte.

Falls man in der glücklichen Lage ist und eine freie PIO in seinem System hat oder, wie beim KC, fertige Hardwaremodule benutzen kann, beschränkt sich der Nachbau auf die Interface-Hardware.

Unser Hardware Spezialist Enrico Grämer (siehe Kontakt), welcher auch die Platine für das KC-Modul layoutet und fertigen lässt, plant neben diesem Modul mindestens noch eine zweite universelle Platine für die ehemaligen Z80 CP/M-Rechner von Robotron & Co. Dort soll die I/O-Adresse der PIO im System per DIP-Schalter frei eingestellt werden können und auch die Interface-Hardware ihren Platz finden. Dann muss man nur noch die Verbindung zum Z80 Systembus herstellen und eine freie I/O-Adresse kennen. Falls an so einer Variante Interesse besteht, setzt Euch bitte mit ihm in Verbindung, ich werde in Richtung Hardware keine Aktivitäten unternehmen.

Firmware

Die Software, welche im ATmega 162 läuft, liegt nun in der Version 1.2 vor. Sie besteht aus 2 Teilen, einer HEX-Datei (xxxx.HEX) für den Flash-Programmspeicher und einer HEX-Datei (xxxx.EEP) für den EEPROM. Die Quellen dieser Firmware sind Closed Source und werden von mir nicht veröffentlicht werden.

Beim Brennen sollte man beachten, dass manche Programmiertools HEX-Dateien nur richtig laden, wenn der Dateityp auch *.hex heisst. In diesem Fall muss man die EEPROM-Datei also vor dem Brennen entsprechend umbenennen, bei der Nutzung des Brennprogrammes von BASCOM ist das beispielsweise der Fall.

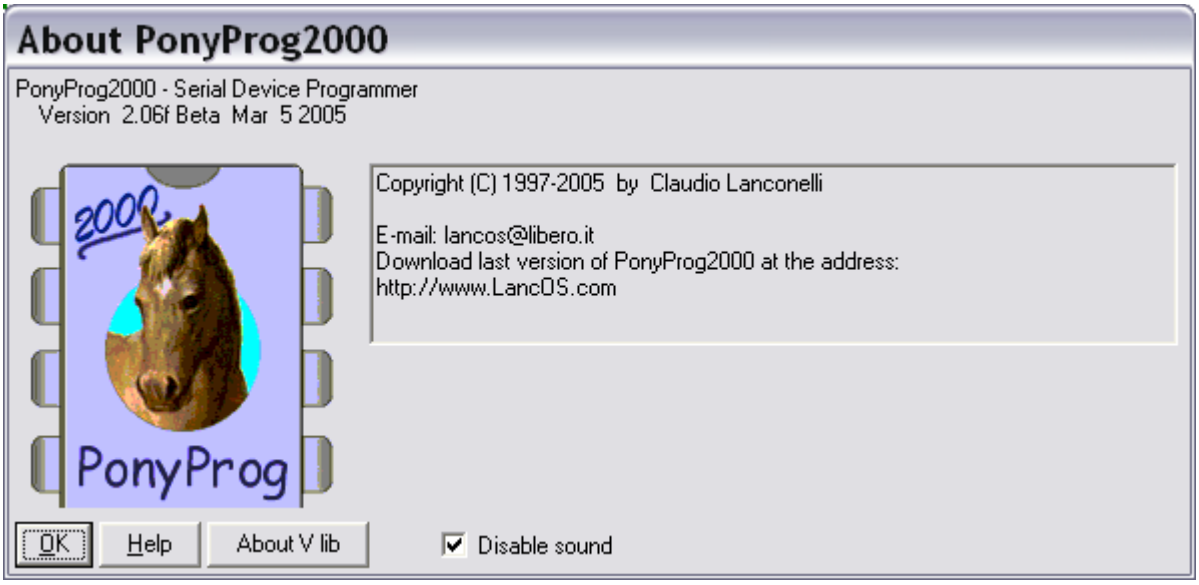
Am gesamten Interface sind keine Änderungen mehr geplant. Nach den letzten Anpassungen in der Version 1.2 sollte einer störungsfreien Nachnutzung nichts mehr im Wege stehen. Die Firmware lässt sich ja auch bei Notwendigkeit im Gegensatz zur Hardware leicht ändern und neu in den Controller programmieren.

Ohne diese Software bleibt die Hardware dumm, nur mit einem programmierten Controller lässt sich das KCNET in Betrieb nehmen. Die Firmware kann kostenlos von mir bezogen werden, allerdings sind an den Bezug einige wenige Bedingungen geknüpft, siehe nächster Abschnitt.

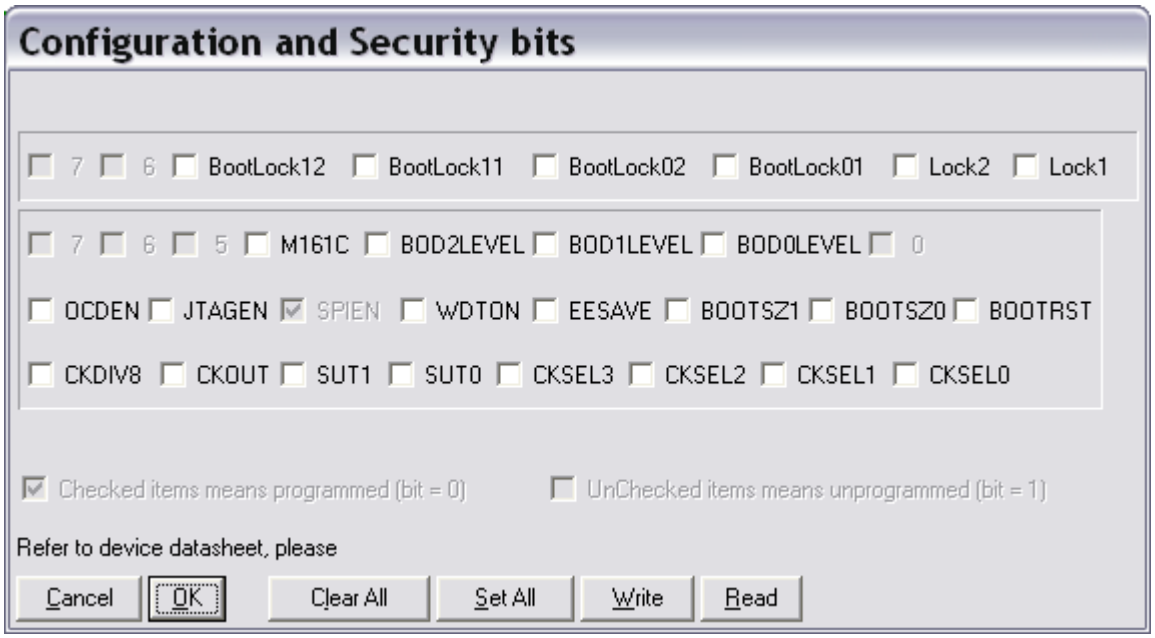
Für jedes Interface-Exemplar wird ein zusammengehöriges eigenes Paar der o.g. beiden Dateien benötigt, welche nicht verwechselt, nicht verändert (CRC Test beim Booten) und auch nicht doppelt oder mehrmals verwendet werden dürfen, ansonsten kann im schlimmsten Fall das Netzwerk lahmgelegt werden.

Prinzipiell gibt es nun 2 Möglichkeiten den Controller fit zu machen, entweder man bezieht einen fertig programmierten Controller inklusive Firmware oder man programmiert den Controller selbst mit dem von mir per E-Mail zur Verfügung gestellten Exemplar seiner Firmware.

Ich verwende z.Z. die freie Version von Ponyprog 2000 in der folgenden Version, welche man auch benutzen sollte, die früheren Versionen können Probleme mit dem ATmega 162 machen. Ab Windows Vista aufwärts sollte man bei Brennfehlern die Hilfe-FAQ im Kapitel 5 mit den AVR Questions beachten. Gleich das erste Q/A-Pärchen beschreibt die Lösung für Timingprobleme :



Zum Programmieren verwende ich die KCNET-Platine oder ein STK200 mit einem Programmieradapter für die parallele Schnittstelle des PC. Für die Selbstprogrammierer ist das Einstellen der Fusebits des Controllers wichtig - die müssen in jedem Fall geändert werden, da ein fabrikneuer ATmega 162 mit anderen Einstellungen ausgeliefert wird !!! - nachfolgend die Einstellungen für Ponyprog 2000:



Wer andere Programmiertools verwendet, sollte sich vorher mit der undurchsichtigen Logik von gesetzten oder nichtgesetzten Fusebits und ihren Auswirkungen auf den Betrieb des Controllers aus dem Datenblatt von Atmel und der Variante von Ponyprog auseinandersetzen. Jede Software interpretiert das "besser" als Atmel, wo das aber auch nicht gerade einfach verständlich erläutert wird.

Beim Programmieren nicht vergessen, dass die Firmware aus zwei Dateien besteht, welche auch beide in den Controller gebracht werden müssen - ein Verify nach der Programmierung kann auch nie schaden.

Im Gegensatz zur Hardware einer PIO, welche fest an den Takt des Z80-Systems gekoppelt ist, kann man die Firmware des Controllers flexibel anpassen. Ein entscheidender Punkt ist hierbei das Hardware-Handshake der PIO für die Daten Ein- und Ausgabe zur Peripherie, welches man sich im Zilog-Datenblatt anschauen kann.

Mit unterschiedlichen Taktfrequenzen des Z80-Systems ändern sich zwangsweise auch die Impulsbreiten der Handshake-Signale. Um hier kein Risiko einzugehen, wurde die Firmware des Controllers, welcher als Peripheriegerät für die /STROBE-Impulse verantwortlich ist, von Anfang an so ausgelegt, dass sich die Breite dieses Impulses automatisch nach der Hostfrequenz richtet. Sie ist mindestens so lang, wie ein Takt des Z80-Systems dauert aber auch nicht viel länger, um die Übertragungsgeschwindigkeit im optimalen Bereich zu halten.

Das wird durch den Assembler während der Übersetzung der Firmware angepasst, so dass die Impulsbreite bei langsamen Systemen entsprechend grösser als bei schnelleren Z80 Systemen eingestellt wird. Im Moment wird ein Bereich von 500 kHz bis 8 MHz PIO-Taktfrequenz abgedeckt, was für alle gängigen Systeme passen sollte. Damit sollte das Interface sowohl an hoch als auch niedrig getakteten Systemen sicher funktionieren.

Voraussetzungen für den Bezug der Firmware

Jeder Teilnehmer eines Ethernet-Netzwerkes hat sich an einige Regeln zu halten, damit sich das Netzwerk physisch und logisch störungsfrei betreiben lässt - dieser Sachverhalt trifft genauso auf das KCNET-Interface zu. Da ich im Falle des KCNET auch Sorge dafür tragen muss, das ein Netzwerk, an dem ein solches Interface angesteckt wird, nicht in die Knie geht, gibt es einige Bedingungen für den Bezug der kostenlosen Firmware.

Eine leidige aber sehr wichtige Geschichte ist die MAC-Adresse für das Netzwerkinterface. Sie wird im Ethernet Layer für das Funktionieren des ARP-Protokolls benötigt und muss sich bei jedem Teilnehmer eines zusammengehörigen Netzwerksegmentes (Teilnetzwerkes) unterscheiden. Sie muss also einmalig sein, um sich an beliebige Netzwerke anschliessen zu können, ohne die anderen (unbekannten) Teilnehmer zu stören.

Leider kommt der WIZnet Chip ohne MAC-Adresse daher, so dass sich die Firmware im Controller um diese Angelegenheit kümmern muss. Normalerweise werden MAC-Adressen von den Herstellern der Netzwerkkarten im Paket erworben, was wie üblich nur durch die Zahlung entsprechender Gebühren möglich ist.

Nach langem Überlegen habe ich mich nun für die folgende Variante entschieden. Wer ein Exemplar der Firmware haben möchte, ist für die MAC-Adresse selbst verantwortlich. Die naheliegendste und auch sicherste Variante besteht darin, einfach die MAC einer anderen Ethernet-Netzwerkkarte, z.B. vom PC, zu übernehmen und diese Netzwerkkarte aus dem Verkehr zu ziehen bzw. zu entsorgen.

Erstens stellt man damit sicher, dass die MAC echt ist und zweitens auch nicht doppelt existieren kann, da die Hersteller der Karten das sicherzustellen haben. Ideal sind alte ISA-Karten, welche man i.d.R. sowieso nicht mehr einsetzen kann, da dieser Bus nahezu ausgestorben ist.

Allerdings muss man die MAC-Adresse auch erst mal herausbekommen. oft ist sie auf der Rückseite der Karte irgendwo aufgedruckt oder steht auf einem Aufkleber. Im schlimmsten Fall muss man die Karte in einen PC einbauen und per Programm oder durch das Betriebssystem auslesen.

Windows XP zeigt im MSDOS-Fenster nach der Eingabe von "ipconfig /all" hinter "Physikalische Adresse.....:" die folgende Ziffernfolge an: 00-0E-05-5E-A3-15 - Das ist eine MAC-Adresse, welche aus 48 Bit besteht und immer byteweise meist in hexadezimaler Form getrennt durch Bindestrich bzw. Doppelpunkt angegeben wird.

Die vom Besteller der Firmware gelieferte MAC-Adresse wird fest in den Flash geschrieben und kann nachträglich auch nicht mehr geändert werden. Deshalb darf man die Firmware-Dateien auch nicht verwechseln oder doppelt verwenden - jedes Interface benötigt eine eigene Version der Dateien, damit sich die MAC-Adressen unterscheiden!

Damit wir nun speziell zu den KC-Treffen kein Desaster erleben, werde ich zusätzlich eine interne Liste führen, wo ich die Namen der Besteller, die Seriennummer für das Interface und die zugehörige MAC-Adresse notieren werde. Eine Nutzung von gelieferten MAC-Adressen kann dann natürlich auch nur erfolgen, wenn es die MAC in der Liste noch nicht gibt!

Neben der wichtigen MAC-Adresse benötige ich folgende Angaben von Interessenten für jedes einzelne Exemplar der Firmware in getrennter Form:

1. Name, Vorname des Bestellers
2. E-Mail Adresse des Bestellers
3. gewünschte MAC-Adresse für das Interface (z.B. von anderer Ethernet-Karte)
4. Bezeichnung für das Z80-System (z.B. KC85 bei unserem System)
5. Taktfrequenz der PIO in MHz (z.B. 1750000 beim KC85, i.d.R. identisch mit dem Systemtakt)

Für jedes Exemplar der Firmware wird eine laufende Seriennummer vergeben, welche sich neben den Punkten 3, 4 und 5 per Debugger anzeigen lässt, so kann und sollte man die

individuellen Anpassungen für seine Bestellung kontrollieren, perfekt ist keiner und manchmal passieren Fehler bei den einfachen und selbstverständlichen Dingen. Die Punkte 1 und 2 sind für mich intern und werden natürlich Dritten nicht zugänglich gemacht, ich werde Euch auch nicht mit Werbung belästigen - versprochen :-)

Zusammenfassung

Abschliessend in Kurzform noch mal eine Checkliste für die Nachnutzung in allgemeiner Form, wenn man alle aufgeführten Punkte selbst oder mit Hilfe anderer Personen erfüllen kann, ist eine Nachnutzung möglich und sinnvoll:

- Z80-System vorhanden
- Taktfrequenz bekannt
- Zugang zum Systembus vorhanden oder herstellbar
- einmalige MAC-Adresse vorhanden oder beschaffbar
- ein Aufbau der PIO-Schnittstellenhardware und Interface-Hardware eventuell auch über Dritte ist sichergestellt
- für eine eventuelle Fehlersuche in der Hardware sind Ansprechpartner vorhanden
- für die Nutzung der RS232-Diagnoseschnittstelle ist ein PC o.ä. mit einem Terminalprogramm vorhanden und es kann auch bedient werden
- falls kein KC85 oder CP/M kompatibles System genutzt wird, ist eine Anpassung und Übersetzung der M80-Assemblerquellen möglich, die Programme können anschliessend auch auf das Zielsystem übertragen werden

Der Lohn der Mühen sollte anschliessend ein funktionierendes Ethernet-Netzwerk Interface sein, welches den TCP/IP-Stack bereits eingebaut hat und unkompliziert genutzt werden kann. Ich denke mittlerweile gar nicht mehr darüber nach, wieviel Arbeit das gemacht hat, sondern nutze ganz selbstverständlich die Netzwerkfunktionalität des KC85/5 unter CP/M oder CAOS um Dateien von ein System auf das andere zu bringen und das ist erst der Anfang ...