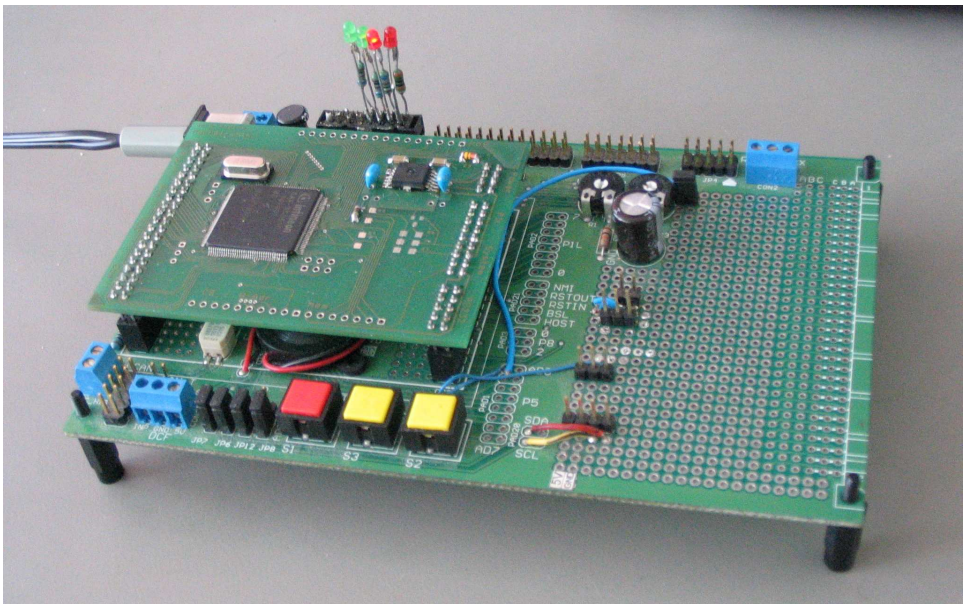
Mikrorechnerentwurf

mit

Kontroller mit XC161



Jens Räthe

Grundgedanke des Layouts

Ich selber besitze eine so genannte „C-Controll II“ von Conrad. In dieser ist ein C164CI von Infineon verarbeitet. Des Weitern sind noch Flash - ROM und RAM integriert, weil der C164CI noch keinen eigenen hatte. Nun bestand die Möglichkeit auf den Entwurf eines Kontrollers mit einen XC161. Dieser gehört, wie man dem Namen schon entnehmen kann zu der gleichen Familie. Der XC161 hat jetzt unter anderem einen integrierten Flash. Somit konnte auf den Einsatz eines externen Speichers verzichtet werden.
Es wurde nun versucht einen Kontroller zu entwerfen der zu der „C-Controll II“ pinkompatibel ist und dadurch auf dem „Applikationsboard“ benutzt werden kann. Der Kontroller ist aber trotzdem, genauso wie auch die „C-Controll II“ ohne dem Board voll funktionsfähig.
Da der Kontroller XC161 nur in dem TQFP 144 Gehäuse mit einem Rastermaß von 0,5mm lieferbar ist, ist er auch das Element welches den meisten Aufwand verursacht. Aus diesem Grund wurde sich entschieden die andern Bauteile auch in SMD – Technik auszuführen. Dadurch steht fast die gesamte Ober- bzw. Unterseite zum Routen zur Verfügung, was bei realisierten 66 IO – Lines recht hilfreich ist.

Bezeichnung der Ausgänge

P1H.0	P1H	P1H.1
P1H.2		P1H.3
P1H.4		P1H.5
P1H.6		P1H.7
GND		GND
GND	PLM	PLM0
Beep		PLM1
FRG1		FRQ0
GND		GND
ADC0	ADC	ADC1
ADC2		ADC3

Power in	Power	GND
2,5V out		GND
5V out		GND
5V out		GND

P1L.7	P1L	P1L.6
P1L.5		P1L.4
P1L.3		P1L2
P1L.1		P1L.0
GND		GND
RSTOUT	SV	NMI

ADC4		ADC5
ADC6		ADC7
GND		GND
I2C SCL	Com	I2C SDA
n.c.		n.c.
CAN_L		CAN_H
TTL TxD0		TTL RxD0
Max CTS		TTL TxD1
Max RTS		TTL RxD1

RSTIN		GND
BOOT		GND
HOST		GND

GND	RS232	n.c.
n.c.		n.c.
TxD0		CTS
RxD0		RTS
n.c. (Pin 1)		n.c.

Tabelle 1: Pinbelegung

Die Ports P1L, P1H und ADC sind mit den entsprechenden Ports am XC161 direkt verbunden. Des Weiteren trifft das auch auf die Pins TTL TxD0, TTL RxD0, TTL TxD1, TTL RxD1, RSTIN, RSTOUT, NMI zu.
Es wurde die ASC0 (TxD0/RxD0) mit einem RS232 Treiber versehen zur Verfügung gestellt. Die zweite Serielle Schnittstelle, ASC1 kann per Jumper als CTS bzw. RTS Signal genutzt werden. Oder bei entsprechendem Kabel können auch beide UARTs über den RS232 – Treiber mit der Umwelt kommunizieren. Da beide Schnittstellen auch auf TTL – Niveau erreichbar sind, kann auch die Hardware - IrDA – Unterstützung der Schnittstellen genutzt werden.
Die beiden I2C Pins sind auf SDA2 und SCL2 geführt und sind so direkt mit der Hardwareeinheit steuerbar, bzw. nutzbar.
Um das CAN Interface zu nutzen ist es noch notwendig einen passenden Treiber aufzulöten. Die zur Zeit bekannten Treiber im gleichen Gehäuse SO8 sind alle kompatibel. Es ist also egal ob es ein Phillips PCA82C250/251, ein TI SN75LBC031 oder ein anderer ist.
Die PLM Anschlüsse sowie der FRQ0 sind mit dem Port 9 verbunden. Sie sind somit für die Benutzung mit der CapCom2 Einheit gedacht.
Eine spezifische Benutzung der jeweiligen Pins als Eingang oder Ausgang ist nicht zwingend notwendig. So ist es zum Beispiel möglich mit zwei CAN Treibern am Port 9 die komplette TwinCAN Einheit zu nutzen (auch eine Mischung mit dem vorbereiten Treiber ist möglich).

Der FRQ1 ist auf den Pin P3.6/T3IN geführt, der Host auf P3.4T3EUD. Somit ist über den gedachten Sinn hinaus mit diesen Beiden eine Hardwarelösung für die Auswertung eines Inkrementalgebers möglich.
Mit RSTIN ist ein Hardware Reset möglich, wenn dieser Pin auf Low gezogen wird. Der dazu nötige Pullup Widerstand wird durch den Spannungsregler bereitgestellt. Hier ist durch den Spannungsregler ein Soft-Hardware Reset realisiert, bei dem die Versorgungsspannungen nicht unterbrochen werden. So ähnlich verhält es sich mit dem BOOT Signal. Wird dieser Pin bei einem Hardware Reset auf Low gezogen, so geht der Kontroller in den Bootmodus und erwartet eine Übertragung über die serielle Schnittstelle. Dies ist per RS232, speziell der ASC0 möglich. Denkbar wäre aber auch eine Übertragung von Kontroller zu Kontroller über die TTL Strecke. Hierzu sind dann TTL RxD0 und TTL TxD0 zu verwenden. Der XC161 unterstützt noch einen Bootmodus über CAN, hierzu ist der Widerstand R2 aufzulöten. Aber dann wird auch der Alternative Start ab der Adresse $41'0000_h$ ausgeführt.

Für eine Erweiterung mit zusätzlichen Speichern wie Flash und/oder RAM ist der Port P0 und die je ersten 4 Chipselect sowie die ersten 4 höheren Adresssignale zur Verfügung gestellt. Dazu kommen noch die nötigen Signale von P20, wie RD, WR, BHE, READY, ALE und EA, diese sind unter Mem im Layout zusammen gefasst. An dem Kontaktbereich werden auch beide Spannungen und die Masse zur Verfügung gestellt. Die Erweiterung kann zum Beispiel durch eine Aufsteckplatine geschehen.

CS				H-Adr				2,5V	5V		Mem					
CS3	CS2	CS1	CS0	A19	A18	A17	A16	V DDI	V DDP	GND	BHE	RD	WR	READY	ALE	EA

Tabelle 2: Belegung der Speichererweiterungsmöglichkeit (unten)

Benutzung ohne Applikationsboard

Die Spannungsversorgung befindet ist auf dem Layout mit „Power“ gekennzeichnet. Hier sind laut der Tabelle 1 an „Power in“ mindestens 6V bereitzustellen.

Für Reset oder Boot steht jeweils eine Masse zur Verfügung um die Signale mit Jumper zu generieren. Die serielle Schnittstelle hat den Pin 1 ganz unten. Weiteres ist für den Betrieb und für die Programmierung nicht nötig. Die Beschaltung ist in der folgenden Abbildung 1 zu sehen. Hier sind auch die Jumper für RTS und DTS gesteckt.

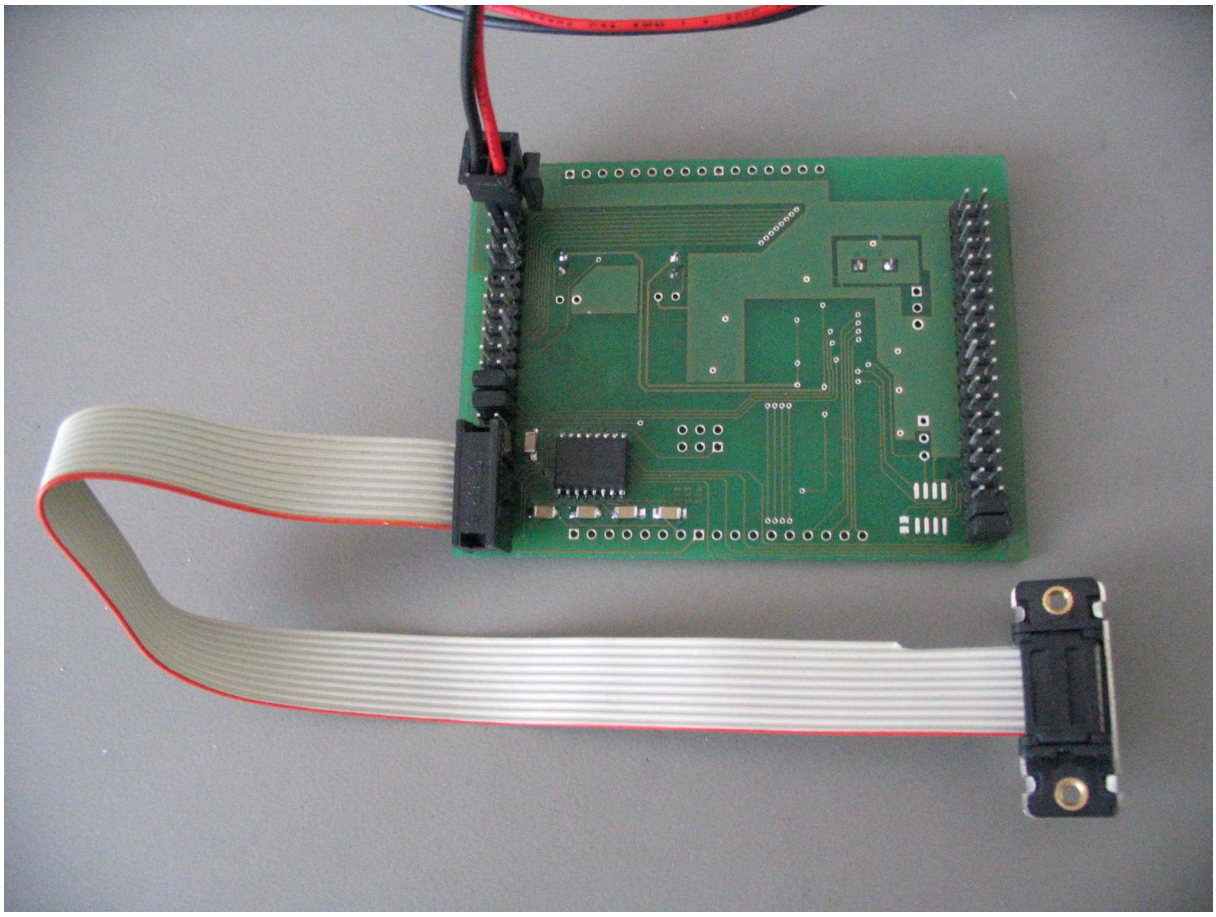


Abbildung 1: Betrieb ohne Applikationsboard

Verwendete Elemente und ihre Aufgaben

Der Kontroller ist sehr übersichtlich, weil sich neben dem XC161 nur noch der Spannungsregler sowie der RS232 Treiber auf dem Board befinden. Auch wenn noch der CAN Treiber hinzukommen sollte, ist der meiste Platz für die Leiterbahnen genutzt worden.

Der XC161 braucht ein paar Widerstände, für die Startkonfiguration. Hierzu dienen die Widerstände welche am Port P20 angeschlossen sind, siehe Tabelle 3.

Bezeichnung	Pin	Funktion
R1	RSTOUT	Pullup für das RESET OUT Signal
R2*	ALE=1	Alternate Start/Boot
R3	EA=1	Internal Start
R4*	WR=0	RSTOUT als P20.12 (purpose IO)
R6		Vorwiderstand für Status – LED
R7	NMI	Pullup um unbeabsichtigtes Auslösen zu verhindern
*	Sind nur nötig/erlaubt wenn ein anderer Modus gewünscht ist	

Tabelle 3: Widerstandsfunktion

Da der Entwurf für einen „internen“ Start ausgelegt ist befinden sich keine Widerstände am Port0. Eine Konfiguration zur Nutzung von externen Speicher kann während der Laufzeit erfolgen oder nach entfernen des Widerstandes R3 auch auf der Speichererweiterung ergänzt werden.
Als Spannungsregler kommt der von Infineon empfohlene TLE7469GV52 zum Einsatz. Dieser stellt beide benötigten Spannungen 5V sowie 2,5V zur Verfügung. Darüber hinaus liefert er noch ein Hardwareresetsignal sobald die Spannungen im zulässigen Bereich liegen. Man könnte noch die Watchdog Funktion nutzen, dies wird aber hier nicht ausgeführt. Die Glättungskondensatoren sind symmetrisch um den Regler herum angeordnet. I im vorliegenden Aufbau sind die Kondensatoren auf das Nötigste reduziert. Es besteht aber die Möglichkeit zum vollständigen empfohlen Aufbau mit polar und bipolar Kondensatoren.

Der RS232 Treiber hat seinen Platz auf der Unterseite gefunden. Direkt daneben befinden sich die Pumpkondensatoren.

Es ist die Debugschnittstelle zugänglich gemacht, wobei auf eine Sortierung verzichtet worden ist. Die JTAG Kontakte für TMS, TDO, TDI, TCK und TRST sind mit einem 5V – Anschluss auf den sechspoligen Port rechtsunten vom XC geführt. Die XC161 spezifische Erweiterung BRKIN und BRKOUT sind zusammen mit einem

Masseanschluss auf den dreipoligen Port linksoben geführt. Es wird im Users Manual des XC161 empfohlen den Pegel von TRST bei Nichtbenutzung auf High zu legen, hierfür kann ein Jumper aufgesteckt werden. Es ist erst nach Erhalt der Platinen aufgefallen das die Anschlüsse für BRKIN, BRKOUT und Masse (sowie der für die ADC Referenzspannung) nicht im Rastermaß sind, so ist es nicht direkt möglich einen Debug – Adapter auf einer Lochkarte zurealisieren.

Auf der Unterseite ist der CAN Treiber vorgesehen. Dieser braucht keine weitere Beschaltung. Direkt daneben ist eine Lötbrücke mit der sich bei den Treiber die Flankensteilheit und somit das EMV – Verhalten anpassen lässt.

Die Beschaltung des Analog – Digital Wandlers ist auf ein brauchbares aber erweiterungsfähiges Maß reduziert. Die analoge Masse V AGND ist mit der digitalen Masse verbunden. Die Referenzspannung V_{AREF} ist noch nicht mit V_{DDP} verbunden. Stattdessen ist sie auf einen Pin links neben dem XC geführt. Parallel zu diesem Pin stehen V_{DDP} und die digitale Masse, bzw. genau der Punkt an welchem die analoge Masse mit der digitalen verbunden ist, zur Verfügung.

Installiertes Demoprogramm

Es wurde ein einfaches Assemblerprogramm geschrieben, welches an P1L und P1H nachvollziehbare Reaktionen hervorruft. Der Quelltext ist für Tasking geschrieben und kann im kompletten im Anhang noch einmal nachgelesen werden. Im Folgenden gehe ich etwas genauer auf den Inhalt ein.

Grundlegender Ablauf

Zu Beginn wird als erstes der OnChip – Watchdog abgeschalten.
Darauf hin wird CapCom2 Einheit durch Bearbeitung des SYSCON3 – Registers aktiviert[1)].

```
diswdt
extr  #1
mov   R1,SYSCON3
bclr  R1.7
extr  #1
```

Mit <einit> wird die interne Initialisierung abgeschlossen. Danach sind keine Zugriffe auf bestimmte Register, wie z.B. SYSCON3 mehr erlaubt. Außerdem wird der /RSTOUT Pin (in der Standardkonfiguration) auf High gezogen.	mov SYSCON3,R1 einit

Als erstes wird die Bearbeitungszeit durch Veränderung des PLL – Konfigurationsregister erhöht, weil bei einem internen Start mit halber Oszillatorfrequenz begonnen wird.
Daraufhin werden die Ports P1L und P1H als Ausgänge geschalten. Es wird der Pin P1H.6 mit der alternativen Funktion verknüpft, und der Timer T7 mit der längst möglichen Laufzeit konfiguriert.
	extr #4
	mov PLLCON,#7354h
	mov DP1L,#0ffh
	mov DP1H,#0ffh
	bset ALTSEL0P1H.6
	mov T78CON,#0007h

P1H.6 oder auch intern CC26 genannt wird als Compare – Channel auf Timer T7 eingestellt und mit einem Wert versehen.
Danach wird der Timer T7 gestartet.
	mov CC26,#7fffh
	mov CC2_M6,#0500h
	bset T7R

Als Hauptprogramm läuft eine einfache Endlosschleife, welche den Pin P1H.5 un einem Verhältnis von ca. 1:3 toggled.
Parallel dazu wird das Highbyte des Timers T7 am Port P1L dargestellt.
	noloop:
	mov R1,CC2_T7
	bset P1H.5
	nop (mehrmals)
	bclr P1H.5
	nop (mehrmals)
	mov P1L,RH1
	jmpr noloop

Der Return – Befehl ist nur vorhanden,	ret

um eine Assemblerwarnung zu unterdrücken.

[1)]Siehe hierzu im Kapitel Probleme bei der Inbetriebnahme

Wenn das Programm mit Tasking gebaut wird und mit FlashIt geladen werden solll, so ist es erforderlich im Hex File den Programmcode hinter die Vektortabbele zu schieben und die zwei Zeilen mit der Speicherplatzbeschreibung zu löschen.

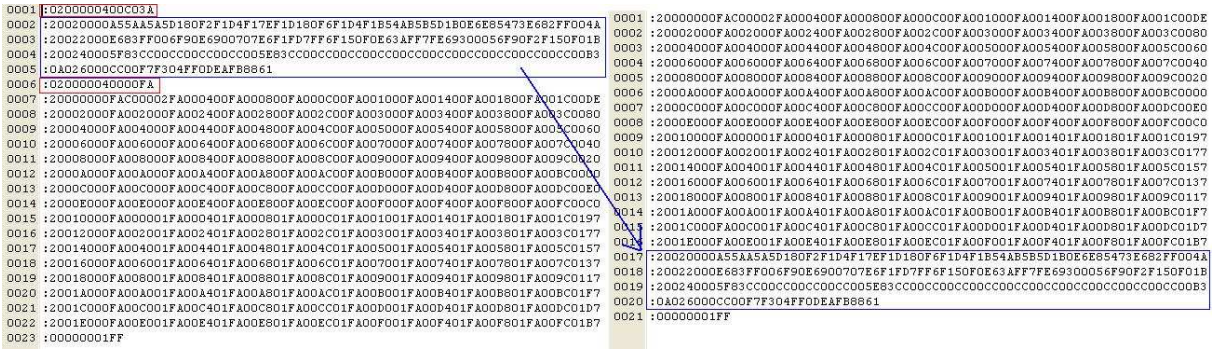


Abbildung 2: Änderung am Hex File

Probleme bei der Inbetriebnahme

Der erste große Fehler der mir unterlaufen ist, war es den Kontrollerchip um 90 Grad verdreht auf zu löten. Das kommt daher, weil im Datenblatt der Kontroller im Kapitel „2.2. Pin Configuration and Definition“ so abgebildet ist, dass Pin 1 rechtsoben ist. Und im Kapitel „5.1. Packaging“ wird aufgeführt das Pin 1 mit einem Kreis dargestellt wird. Dieser ist hier mit Pin 1 linksunten. Das Design des Layout hat sich am einfachsten gestaltet mit dem Pin 1 rechtsoben. Nun habe ich den Chip beim ersten Anlauf blind nach der Orientierung der Beschriftung aufgelötet. Diese ist nun aber in der Ausrichtung, welche beim Packaging genutzt wird aufgebracht, und somit der Pin 1 linksunten.

Des Weiteren habe ich beim Durcharbeiten der Datenblätter, beim Spannungsregler einen internen Pulldown Widerstand als Pullup gelesen, was erst bei der Inbetriebnahme aufgefallen ist. In diesem Fall ist der Spannungsregler nicht eingeschalten worden. Der Fehler ist durch einen 100K Widerstand zwischen Pin 1 und 2 korrigiert worden.

Bei der Programmierung bin ich über einen Fehler im Manual gestoßen, oder es muss irgend wo anders noch erwähnt sein das ein Großteil der Peripherie beim Start nicht eingeschalten ist. Im Manual wird auf Seite 6-48 als Resetwert des SYSCON3 0x0000 angegeben. Der festgestellte Resetwert ist aber 0xFCC4 ist. Wobei das gesetzte Bit 4 das Merkwürdigste ist, weil diese Stelle gar nicht definiert ist.
Genauso ein nicht definiertes, aber gesetztes Bit befindet sich an der Stelle SYSSTAT.4.

Vorüberlegungen zum Applikationboard

Als Applikationsboard oder Motherboard kann das von Conrad in Frage kommen. Die Benutzung ist auf dem Titelbild zu sehen. Es ist hier ein Kondensator zu entfernen, weil hier der Platz für die Speicheradressleitungen benötigt wurde.
Es gibt noch einen privaten Anbieter, der die Lieferung von Komponenten für die C-Controll II fast komplett von Conrad übernommen hat. Dieser hat unter anderem auch zwei weiter Boards im Angebot. Da der Controller aber ein wenig größer geworden ist als die CCII, kommt nur das Reglerboard in Betracht. Bei dem Starterboard ist zu prüfen ob dieses verwendet werden kann.

Da als Grundgedanke die Steuerung eines Kipptisches stand, können, bzw. sollten folgende Aspekte realisiert werden:

- Stecker für Spannungsquelle auswählen
- Port P1H zusammen mit 5V und Masse 3reihige Stiftleise für PLM Nutzung
- Alle Bus-Schnittstellen für Kommunikation zu Verfügung stellen:
    - Sub-D für RS232
    - Möglichkeit für TwinCAN offen lassen, d.h. PLM0 und PLM1 offen lassen
    - I2C mit 5V und Masse auf 4pol Stiftleiste (PS2 – Realisierung)
    - Inkrementalgeberanschluss
    - SSC - Anschluss (nicht unbedingt nötig)
- Taster für Reset und Boot
- PullUp für FRQ0 um Messung zu ermöglichen

- Jumper für RTS und CTS wenn benötigt
- ADC Port vorbereiten

Anhang

Quelltext

```
$nonsegmented
$debug
$model(small)
$extend2
$nomod166
$stdnames(regxc161cj.def)

; !!!!!!!!!!!!!!!!!!
; This is a testprogram for the bootloader
;!!!!!!!!!!!!!!!!!!

SSKDEF 0
; ***************************************************************
test_c     SECTION CODE AT 0C00200h
assume    dpp3:system
REGDEF
; ***************************************************************

nix    proc task intno=0
diswdt
extr    #1
mov     R1,SYSCON3
bclr    R1.7
extr    #1
mov     SYSCON3,R1
einit
extr #4
mov     PLLCON,#7354h
mov     DP1L,#0ffh
mov     DP1H,#0ffh
bset    ALTSEL0P1H.6
mov     T78CON,#0707h
mov     R1,#7ffdh
mov     T7,R1
mov     CC26,#7fffh
mov     CC2_M6,#0500h
bset    T7R
noloop:
mov     R1,CC2_T7
bset    P1H.5
nop
nop
nop
nop
bclr    P1H.5
nop
nop
```

```
nop
nop
nop
nop
nop
nop
nop
nop
nop
mov  P1L,RH1
jmpr noloop

ret
nix endp
test_c ends

end
```

Bezugsquellen und Anlaufpunkte

Datenblätter, Manuals

Infineon.com	

Hardware

CC2Net.de	- Applikationsboard - Erweiterungen
Conrad	- Applikationsboard - Robotik (in Verbindung mit der CCII) - Bauteile
Jenaer Leiterplatten GmbH	- Leiterplatte ist in der CAM gespeichert

Software

www.CC2Net.de	- Bootloaderprogramm Demo (hierzu muss aber eine andere „Boot.hex“ Datei geschrieben werden, was ich zur Zeit noch Versuche)

	- Demo funktioniert bei zurück setzen des Datums
www.Hse-electronics.com	- Bootloaderprogramm Demo, wurde genutzt
Tasking (Altium), Keil, www.kpitcummins.com	- C/C++ Compiler - Entwicklungsumgebungen
www.Mikrocontroller.net	- Forum mit XC – Programmierern

CD Inhalt

- Datenblätter, Manuals
- Quelltext des Programms, fertiges Hex File
- Demos der Entwicklungswerkzeuge
- Schaltplan und Platinenlayout als Bild und OrCAD Format, Spannungsreglerbeschaltung korrigiert
- Dieses Dokument

Die CD ist als Multi-Session gebrannt und kann somit für die Sammlung weiterer interessanten Dokumenten genutzt werden.