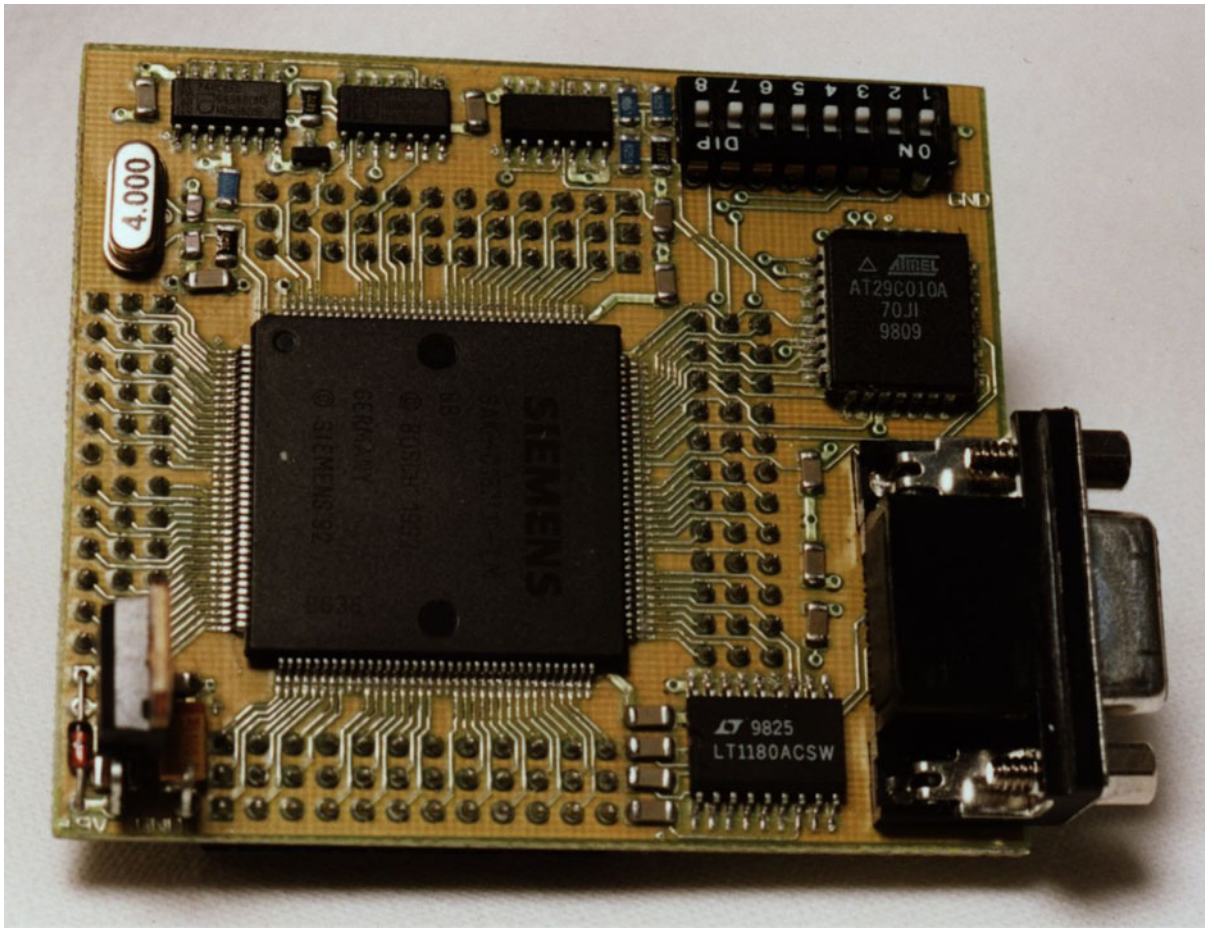
Ein Entwicklungskern für den Microcontroller C167CR – LM

Ass. Prof. Dipl.-Ing. Dr. Robert Röhrer
Institut für Elektronik, TU Graz

Graz, 2001-09-29

Für den Einsatz in Lehre, Forschung und industrieller Geräteentwicklung wird hier eine Entwicklungsplatine vorgestellt, die folgende Eigenschaften besitzt:

- Laden von Applikationsprogrammen im Standard Intel Hex Format mit jedem gängigen Terminalprogramm als Textdateien, dadurch bedingte Unabhängigkeit von Rechnerplattformen und Betriebssystemen
- Keine Modifikation der Adressen oder Interruptvektoren bei der Programmentwicklung
- Kein Verlust von Speicherresourcen durch „Verstecken“ des Laders sowie Schutz des Laders gegen Überschreiben
- Erstmaliges Einbringen des Laders über den Bootstrap Lader in die fertig bestückte Leiterplatte
- Direkter Programmstart des fertig entwickelten Applikationsprogrammes durch Umlegen eines DIP Schalters
- Einsteckbarkeit in die Leiterplatte der Applikation



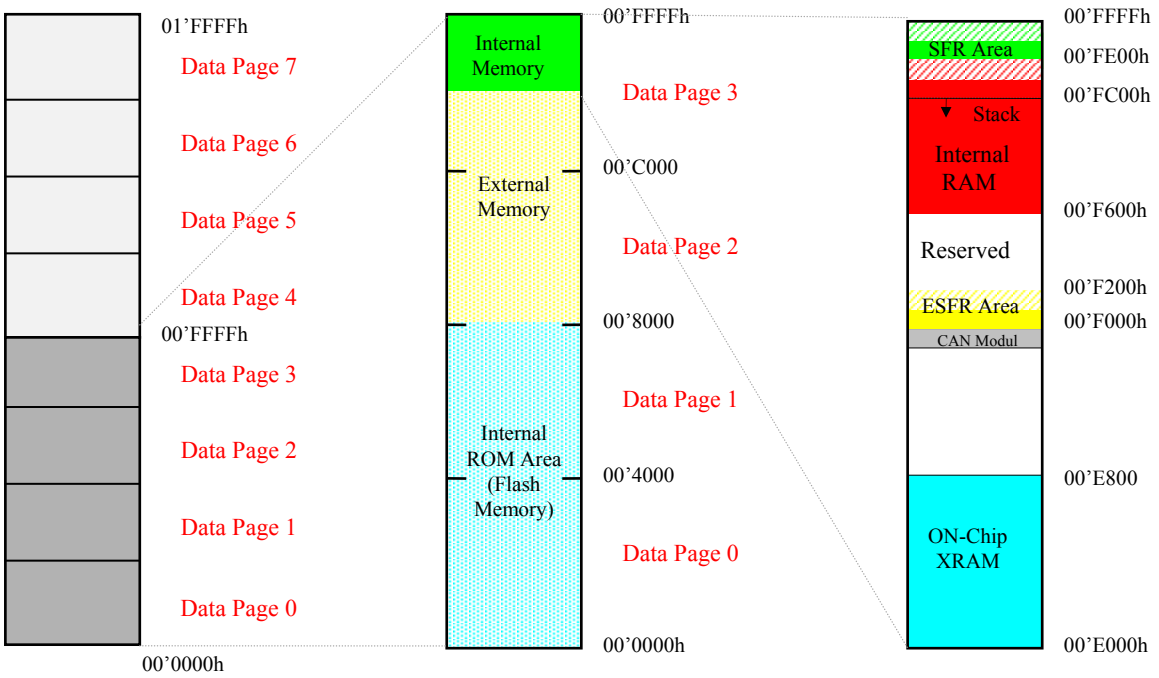
1 Allgemeine Beschreibung des C167CR – LM

Die wichtigsten Merkmale sind:

- 16 Bit Zentraleinheit
- 2 kByte internes RAM
- 2 kByte On-Chip XRAM
- umschaltbare Registerbänke
- 111 E/A Leitungen
- 16 MByte linear adressierbarer Daten- und Programmspeicher
- einstellbare Buskonfiguration
- 8 bit oder 16 bit Datenbusbreite
- 56 Interrupt Quellen mit 16 Prioritäten
- 8 Kanal Peripheral Event Controller (PEC)
- 2 General Purpose Timer Units (fünf 16 bit Zeitgeber/Zähler mit 200ns bzw. 400ns Auflösung bei 20 Mhz CPU Frequenz)
- 16 Kanal 10-bit ADU mit programmierbarer Umsetzzeit
- 2 Zeitgeber mit 16 Kanal Capture/Compare Einheiten
- 4 8-bit pulsweitenmodulierte Ausgänge
- serielle Schnittstelle
- Bootstrap Loader
- On Chip CAN Modul

1.1 Speicheraufbau des 167CR-LM

Der Speicher des C167 ist als ‘von Neuman’ Architektur aufgebaut, d.h. Befehle und Daten sind im gleichen linearen Addressraum angeordnet. Der C167 kann insgesamt bis zu 16 Mbytes addressieren, die in 256 Segmente zu je 64 Kbyte aufgeteilt werden.
Jedes Segment ist noch in vier Datenseiten zu 16 Kbyte unterteilt.



Segment 0 (64kB)
Segment 1 (64kB)
Die oberen 256 Byte des SFR-, ESFR- und des internen RAM-Bereiches sind bitadressierbar.

Special Function Registers: Die Funktionsweise der CPU, des Businterfaces, der E/A-Leitungen usw. kann mit den SFR´s festgelegt werden. Sie sind in 2 Registerblöcken zu je 512 Bytes angeordnet. Der erste Block (SFR) liegt über dem internen RAM (00'FFFFh ...00'FE00h), der zweite Block (Extended SFR) liegt in den 512 Bytes unter dem internen RAM (00'F1FFh .. 00'F000h)

System Stack: Die Größe wird vom Bitfeld STKSZ im Register SYSCON festgelegt. Er wächst von höheren zu niederen Adressen hin (Default: 00'FBFEh-00'FA00h). Ein Stack Über-Unterlauf kann durch die beiden Register STKOV und STKUN festgestellt werden.

General Purpose Registers (GPR): Ein Block von 16 Word Registern (R0-R15). Der Context Pointer (CP) zeigt auf die Basisadresse der aktuellen Registerbank. Eine Registerbank belegt maximal 32 Byte. Jedes Bit in diesen Registern kann einzeln angesprochen werden. Im Gegensatz zum Stack wächst die Registerbank von niederen zu höheren Adressen hin.

PEC Source und Destination Pointers: Diese Word Zeiger (SRCPx, DSTPx) sind zwischen bitadressierbaren RAM (00'FCFEh) und dem Stack (00'FCE0h) angesiedelt.

On Chip XRAM : Die 2 kByte XRAM liegen im untersten Bereich des internen Speichers (00'E7FFh ... 00'E000h). Es kann über das Bit XPEN im Register SYSCON deaktiviert werden. Ist dieses Bit gelöscht, so werden alle Zugriffe auf das XRAM in den externen Speicher umgeleitet. Das XRAM kann zur Programm- und Datenspeicherung verwendet werden, jedoch nicht für den Stack und die Registerbänke (GPR's). Im Gegensatz zu anderen Bereichen des internen Speichers ist es nicht bitadressierbar.

1.2 Der System Reset

Er kann im Prinzip durch 3 Arten ausgelöst werden:

- Hardware
- Software Befehl
- Watchdog Timer

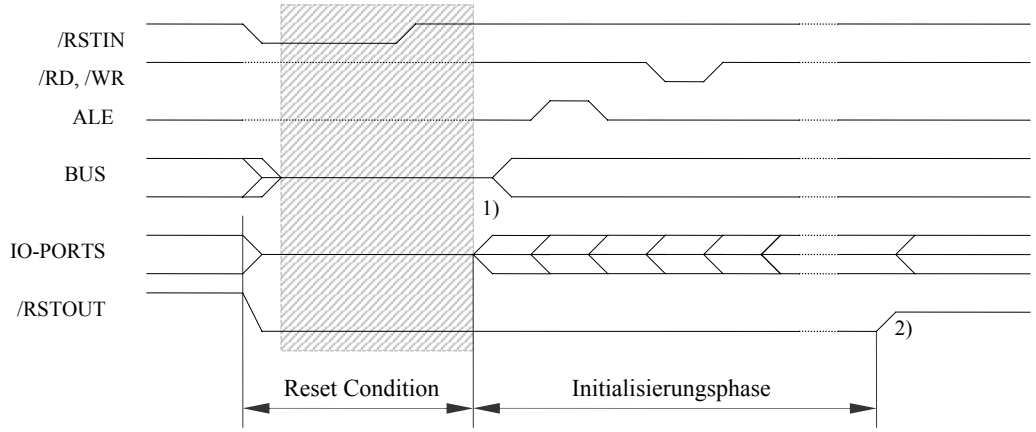
Mit Hilfe der /EA- Leitung wird festgelegt, ob der C167 von 'außen' konfiguriert werden kann, oder ob er mit der Standardkonfiguration startet. Zusätzlich legt es auch noch den Wert des Bits ROMEN (ROM enable) im Register SYSCON fest.

Pin /EA	**verwendeter Speicher**
1	internes ROM (On Chip)
0	externer Speicher

Im vorhandenen Prozessorkern ist sie fest auf LOW-Pegel gelegt (externe Konfiguration).

1.2.1 Ein/Ausgangssignale des C167 während des Resetvorganges

Wird ein Reset ausgelöst, so wird eine sogenannte Reset-Sequenz gestartet, die genau 516 CPU Zyklen (bzw. solange der Eingang /RSTIN auf LOW-Pegel liegt) dauert.
Nach der Reset-Sequenz werden der Adreß- bzw. Datenbus und die entsprechenden Kontrollsignale aktiviert. Alle anderen E/A-Ports bleiben als Eingänge geschalten und der C167 startet die Programmausführung bei Addresse 00'0000h.
Hier steht der Einsprungvektor für die Initialisierungsroutine.



1) Ende der Reset-Sequenz. Der C167 startet mit der Programmausführung
2) Ende der Initialisierungsphase (Ausführung des EINIT-Befehls).

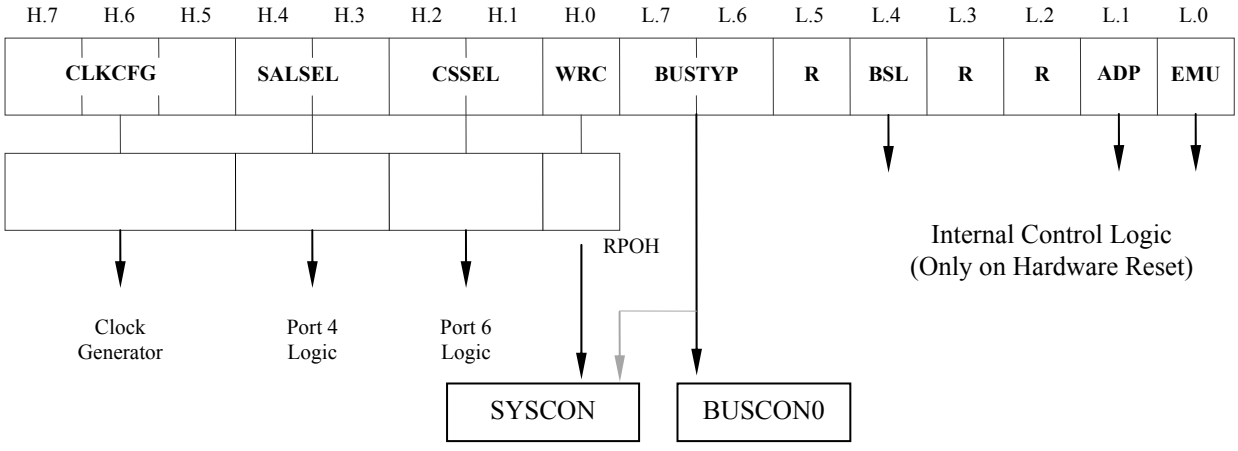
DAS /RSTOUT Signal bleibt solange auf LOW Pegel, bis die Initialisierungsphase mittels Softwarebefehl (EINIT) beendet wird.

Watchdog Timer: Fängt nach der Reset-Sequenz mit einem Rückladewert von 00h an. Wurde der Reset durch den Watchdog Timer ausgelöst, so ist das WDTR Bit (Watchdog Timer Reset Indication) im Konfigurationsregister WDTCON gesetzt. Er kann über das Bit DISWDT (Disable Watchdog Timer) ausgeschalten werden.

Register : Der Speicherinhalt des On-Chip RAM's wird nur durch Spannungsausfall verändert, ansonsten bleibt er unverändert.

Ports und ext. Bus während der Reset-Sequenz: Während der Reset-Sequenz sind alle Ports als Eingänge geschalten, während die Augangstreiber hochohmig sind. Das Signal ALE wird über einen internen Pull-down Widerstand auf LOW-Pegel gehalten, während die Signale /RD und /WR über einen internen pull up Widerstand auf HIGH-Pegel gehalten werden.[1] Liegt am EA Pin LOW-Pegel an, wird über Port P0 am Ende der Reset-Sequenz die System-Konfigurationen eingelesen.

System Konfiguration: Während der Reset-Sequenz sind im Port P0 interne Pull up Widerstände aktiv, wodurch der Eingangspegel ohne äußere Beschaltung HIGH ist. Das Highbyte von P0 wird direkt in RP0H geschrieben.



[1] siehe auch Diagramm oben

Bit ***EMU*** (P0L.0): Emulationsmodus wenn ‘0’
Bit ***ADP*** (P0L.1): Adaptermodus wenn ‘0’
Bit ***BSL*** (P0L.4): Bootstrap Loader wenn ‘0’
Bits ***BUSTYP*** (PL0.6,P0L.7): Konfiguration des externen Busses

Btyp	Datenbusbreite	Adressbustyp
0 0	8 Bit	demultiplexter Adreßbus
0 1	8 Bit	gemultiplexter Adreßbus
1 0	16 Bit	demultiplexter Adreßbus
1 1	16 Bit	gemultiplexter Adreßbus

Bit ***WRC*** (P0H.0): Alternativfunktion der Signale /WR, /BHE
Bits ***CSSEL*** (P0H.1, P0H.2): Legt die Anzahl der /CS Signale fest

CSSEL	Anzahl der /CS-Signale
1 1	/CS4 - /CS0
1 0	keine
0 1	/CS1 - /CS0
0 0	/CS2 - /CS0

ACHTUNG! Die nachfolgenden Konfigurationen können nicht mehr geändert werden (werden in RP0H gespeichert).

Bits ***SALSEL*** (P0H.3, P0H.4): Legt Adreßsegment Leitungen fest

SALSEL	Addreßleitungen	Speichergröße
1 1	A17 - A16	256 kByte
1 0	A23 - A16	16 MByte
0 1	----------	64 kByte
0 0	A19 - A16	1 Mbyte

Bits ***CLKCFG*** (P0H.5 - P0H.7): Legt die CPU Frequenz fest

CLKCFG	CPU Frequenz
1 1 1	F_{XTAL} * 4 = 16MHz
1 1 0	F_{XTAL} * 3 = 12MHz
1 0 1	F_{XTAL} * 2 = 8MHz
1 0 0	F_{XTAL} * 5 = 20MHz
0 X X	F_{XTAL} * 1 = 4MHz

1.3 Der Bootstrap Loader

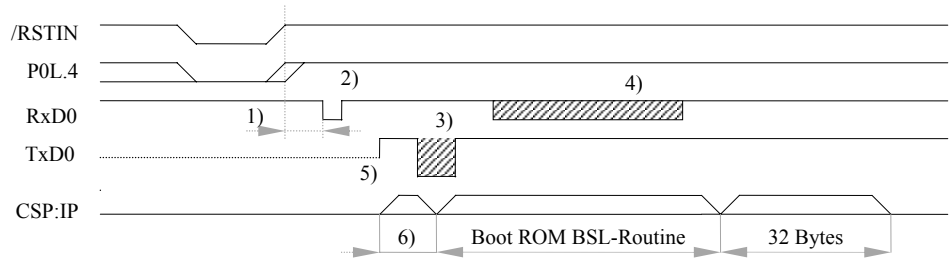
Der Bootstrap Loader (BSL) ist ein Mechanismus, über welchen ein Start(Lade)programm über die serielle Schnittstelle geladen und anschließend ausgeführt werden kann. Dazu wird kein externer Speicher oder internes ROM gebraucht. Der Bootstrap Loader lädt Kode/Daten in das interne RAM. Es ist auch möglich Daten in vorhandenes externes RAM zu laden, indem man eine zweite Laderoutine benützt.

Der Bootstrap Loader Mode wird durch einen LOW-Pegel Pin P0L.4 aktiviert. Das Bootstrap Programm ist in einem speziellen On-Chip Boot ROM gespeichert. Nach dem Start des Bootstrap Loaders wird über RxD0 (Pin P3H.4) ein Zero Byte (1 Startbit, 8 Datenbits, 1Stopbit) abgewartet. Dieses Null Byte dient zur Berechnung der nötigen Baudrate. Danach wird über TxD0 ein Identifikationsbyte (C5h für C167) zurückgesendet.

Jetzt werden 32 Byte über die serielle Schnittstelle empfangen und in den Speicherstellen 00'FA40h bis 00'FA5Fh im internen RAM abgelegt. Danach springt der BSL an die Speicherstelle 00'FA40h wo die erste Instruktion steht und führt diese aus. Diese 32 Byte können nun eine Laderoutine enthalten, welche die schon initialisierte serielle Schnittstelle nützt, und Daten in zuvor festgelegte Speicherbereiche schreibt.
Dieser nun geladene Kode kann nun der endgültige Programmkode oder eine zweite Laderoutine sein, die ein Übertragungsprotokoll zur Verfügung stellt bzw. empfangene Daten im externen Speicher ablegt.
Der BSL Mode kann nur über einen Software Reset (der Pegel an P0L.4 wird dieses mal ignoriert!) oder über einen Hardware Reset (P0L.4 muß nun HIGH-Pegel führen !) verlassen werden. Nach dem Reset wird die Programmausführung bei Adresse 00'0000h im internen ROM oder im externen Speicher (abhängig von Pin /EA) gestartet.

Konfiguration nach dem Start des Bootstrap Loaders:

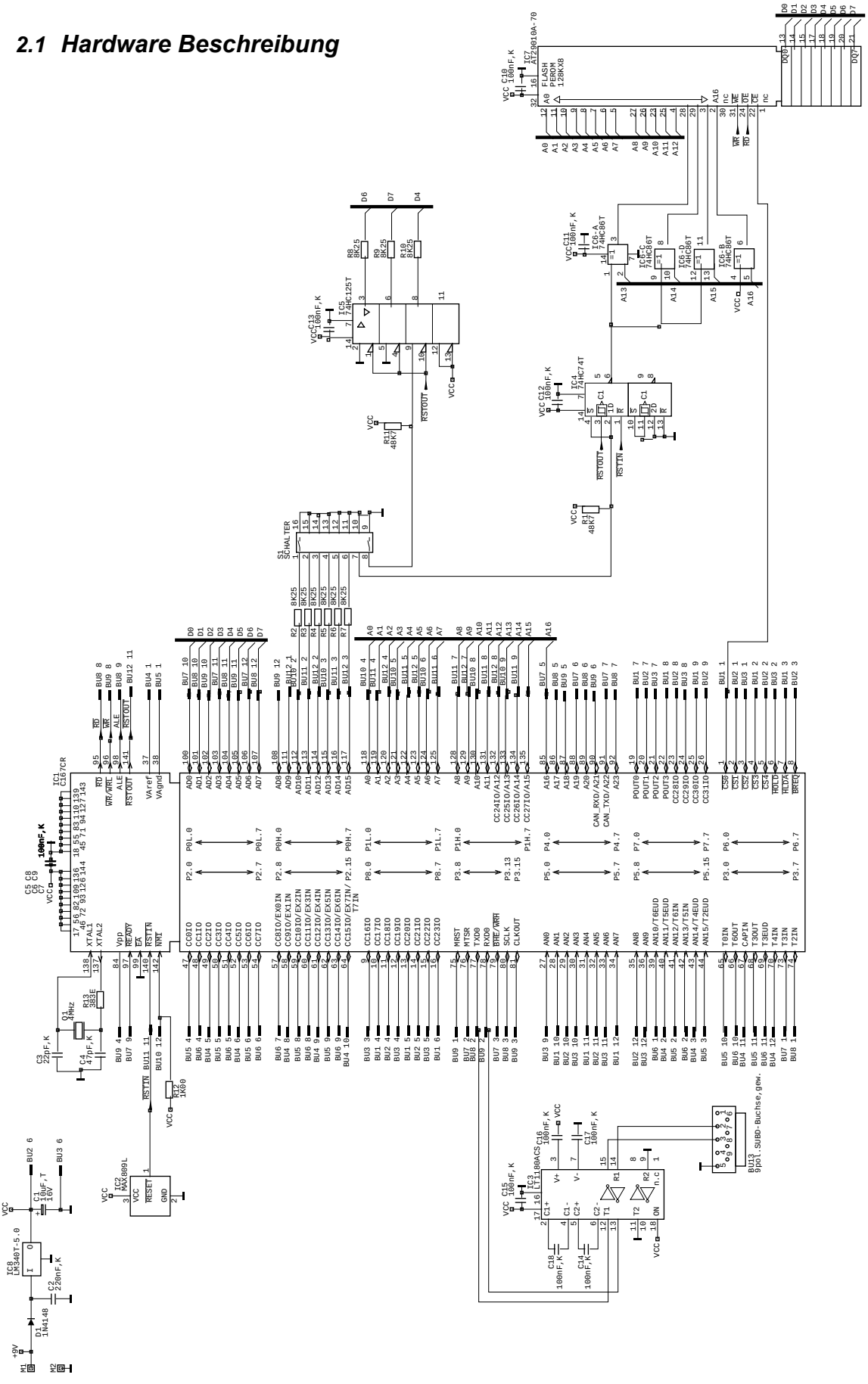
Watchdog Timer:	Disabled	Register SYSCON: 0E000H	
Context Pointer CP:	FA00H	Register STKUN: FA40H	
Stack Pointer SP:	FA40H	Register STKOV: FA0CH 0<->C	
Register S0CON:	8011H	Register BUSCON0: acc. to startup config.	
Register S0BG: acc. to '00' byte		P3.10 / TXD0: ' 1'	
		DP3.10: ' 1'	



1) BSL Initialisierungszeit, > 2µs
2) Zero Byte (1 Startbit, 8 '0' Datenbits, 1 Stopbit)
3) Identifikationsbyte
4) 32 Byte Kode/Daten
5) TxD0 wird nach einer bestimmten Zeit nach Empfang des Zero Bytes gesendet (2,5 µs)
6) Internes Boot ROM

2 Beschreibung des Entwicklungskerns

2.1 Hardware Beschreibung

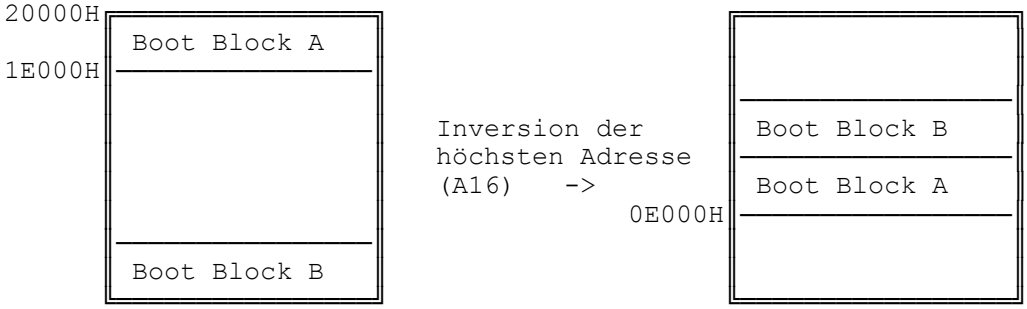


Die Schaltung benötigt als Stromversorgung eine Gleichspannung >8 Volt, die von einem Fixspannungsregler auf 5 Volt stabilisiert wird. Damit werden alle auf dem Print vorhandenen integrierten Schaltungen versorgt; eine Ausnahme bildet der Anschluß für den AD Umsetzer auf dem Mikrocontroller C167, dessen Anschlüsse bei Bedarf extern beschaltet werden müssen. Als Restbaustein wird der IC MAX 809 verwendet. Der Quartzoszillator ist mit einem 4 MHz Quartz beschalten, dadurch ergibt sich intern ein CPU Takt von wahlweise 4, 8, 12, 16 oder 20 Mhz. Als Buskonfiguration ist ein 8 bit Datenbus, Adressbus nicht multiplex gewählt.

Zur Verbindung mit dem Host-Computer ist eine RS232 Schnittstelle ohne Handshakeleitungen über den Baustein LT1080 vorgesehen, der die kompatiblen Spannungspegel erzeugt. Als externer Speicher dient ein FLASH AT29C010 mit 128 kByte, dessen Adreßleitungen speziell beschalten sind. A16 ist immer invertiert, das heißt, dass der obere und der untere 64 KByte Block immer vertauscht angesprochen wird.
Das FLASH besitzt zwei Bootblöcke mit Schreibschutz und durch diese Vertauschung kann ein Bootblock als Speicher für einen Lader verwendet werden, der dann bei Bedarf mit einem nicht mehr aufzuhebenden Schreibschutz versehen werden kann. Die Adressleitungen A15, A14 und A13 können über EXOR Gatter 74HC86 wahlweise invertiert werden. Dadurch ist es möglich, den Speicherbereich von 0E000H bis 0FFFFH auf 00000H bis 01FFFH zu mappen. Das ist jener Speicherbereich des FLASH, der normalerweise nicht angesprochen werden kann, da er sich mit dem internen Speicher des Controllers überlappt.

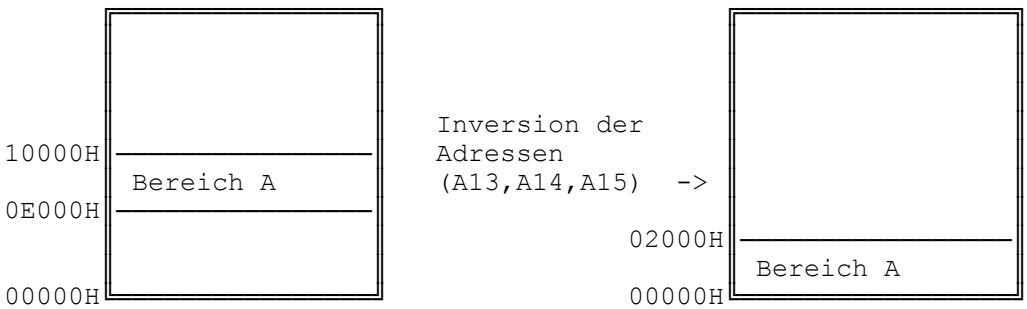
a) Boot-Block Verschiebung

Das FLASH-PEROM der Firma Atmel 29C010 bietet die Möglichkeit 2 Bereiche im Speicher mit einem Schreibschutz zu versehen. Diese beiden Bereiche befinden sich am oberen bzw. unteren Ende des Speichers. Durch geeignete Modifikation der Adressen kann dieser Bereich verschoben werden.



b) Verschieben von Speicherbereichen

Um wahlweise einen physikalischen Speicherbereich auf einer anderen logischen Adresse einzublenden müssen durch Hardware-Beschaltung die Adreßleitungen des Speicherbausteines modifiziert werden.



Die Umschaltung der Adreßbereiche wird vom Ausgang /Q des Flip-Flops 74HC74 vorgenommen. Am Eingang des Flip Flops liegen folgende Signale:
.
Der Reset des Resetbausteins am asynchronen /CLEAR
Der Resetout des C167 am CLOCK
Ein DIP Schalter auf dem D Eingang und dem asynchronen /SET

Dadurch sind zwei Betriebszustände möglich:

Der DIP Schalter liegt auf LOW; die Adressen sind nie invertiert (SET Dominant!)
Dies ist die normale Stellung, um ein Applikationsprogramm, das sich bereits im FLASH befindet, ab der Programmstelle 0 zu exekutieren.

Der DIP Schalter liegt auf HIGH; die Adressen sind zu Beginn durch das /CLEAR des Bausteins invertiert, der Lader wird exekutiert, er kopiert sich in das interne Ram des Controllers und wird dort weiter exekutiert. Dabei wird erstmals der Befehl EINIT (End of initialisation) ausgeführt. Dadurch kommt es zu eine positiven Flanke am CLOCK, das HIGH des DIP Schalters erscheint als LOW am /Q des Flip Flops, die Adreßinversion wird aufgehoben und ein Applikationsprogramm kann in das FLASH geladen werden. Bei einem nachfolgenden Software_Reset Befehl wird dieses dann gestartet.

Auf der Schaltung befinden sich weitere 7 DIP Schalter, die auf die Systemeinstellung Einfluß nehmen.

DIP- Schalter Zuordnung:

Chip Select Leitungen

SW 1	CHIP SELECT LEITUNGEN
Aus	Fünf: /CS4 .. /CS0
Ein	Zwei: /CS1 .. /CS0

Segment Adreßleitungen

Sw 3	Sw 2	Segment Adreßleitungen	verfügbarer Adreßbereich
Aus	Aus	Zwei: A17 .. A16	256 kByte
Aus	Ein	Acht: A23 .. A16	16MByte
Ein	Aus	Keine	64KByte
Ein	Ein	Vier: A19 .. A16	1 MByte

Takt Generierung

Sw 6	Sw 5	Sw 4	CPU Frequenz
Aus	Aus	Aus	f_{XTAL} * 4
Aus	Aus	Ein	f_{XTAL} * 3
Aus	Ein	Aus	f_{XTAL} * 2
Aus	Ein	Ein	f_{XTAL} * 5
Ein	X	X	f_{XTAL} * 1

Operations-Modus Auswahl

Sw8	Sw7	Operations Modus
Aus	Aus	Hex-Lader Modus
Aus	Ein	Applikations Modus
Ein	Aus	Nicht zulässig
Ein	Ein	Bootstrap Lader

2.2 Zeitverhalten des externen Bus Interfaces

Die programmierbaren Parameter für das Bus Zeitverhalten des C167 sind **ALE Control**, **Memory Cycle Time**, **Memory Tri State Time, Read/Write Delay** und **READY Control.** ALE und Ready werden bei der Kern Hardware ohne Erweiterung nicht verwendet. (8 bit non multiplex Busmode für CS0)

2.2.1 Controller		**External Memory** (FLASH PEROM – AT29C010A-12JC)	
Address to valid Data In	4*TCL +2ta+ t_C-30 [ns]	Address to Output Delay	120 [ns]
\CS low to valid Data In	3 TCL+2ta + t_C-20 [ns]	\CE to Output Delay	120 [ns]
\RD to valid Data In (no R/W delay) \RD to valid Data In (with R/W delay)	3TCL + tc - 20 [ns] 2 TCL + t_C - 20 [ns]	\OE to Output Delay	50 [ns]
Data float after \RD	2 TCL + t_F – 15 [ns]	\CE or \OE to Output Float	30 [ns]

t_C = 2*TCL * <MCTC>
t_F = 2*TCL * <MTTC>
ta = 0, da kein ALE verwendet wird (non multiplex**)**

2*TCL = Periode des CPU Clock (50ns at CPU clock=20MHz)
<MCTC> = Anzahl der Waitstates während des Speicherzugriffs (0..15)
<MTTC> = Anzahl der memory Tri-State waitstates (0 oder 1)

Das System arbeitet korrekt, wenn die Zeiten des Controllers größer sind als die des Speichers.

„Address to Output Delay“ des externen Speichers muß um die Verzögerungszeit der Gatter der Adresseninvertierung erhöht werden, (propagation delay 74HC86 = 10ns), ergibt 130ns.

Daher ergibt die Kalkulation für den C167 Kern bei 20Mhz::

Address to valid Data In: 100ns – 30ns = 70ns + 2 Waitstates(MCTC) 100ns = 170ns > 130ns

\CS low to valid Data In: 75ns – 20ns = 55ns + 2 Waitstates(MCTC) 100ns = 155ns > 120ns

\RD to valid Data In: (with R/W delay) 50ns – 20ns = 30ns + 2 Waitstates(MCTC) 100ns = 130ns > 50ns

Data float after \RD: 50ns – 15ns = 35ns > 30ns (MTTC nicht nötig)

Zusätzlich muß noch die Mindestlänge von \RD und \WR berücksichtigt werden.

3*TCL + tc – 10ns = 75ns –10ns = 65ns + 2 Waitstates(MCTC) 100ns = 165 ns > 90ns

3 Beschreibung der Operationsmodi

3.1 APPLICATION MODE

Er dient zum Ausführen eines fertigen Applikationsprogrammes mit dem Programmstart auf Adresse 00000H

Hardware: DIP Schalter #7 ON

Die Adreßleitungen A13 bis A15 sind nicht invertiert, eine geladene Applikation sieht nicht die Programme

PRELOAD_FLASH gespeichert von 0E000H bis 0E07FH im FLASH und
HEX_LOADER gespeichert von 0F700H bis 0FDFFH im FLASH,

da sie sich im Speicher Mapping mit internen RAM, Peripherie und XRAM überlappen.

Das Speicher Mapping wird durch das Applikationsprogramm bestimmt.

Die Soft- und Hardware ist jetzt im APPLICATION STATE

3.2 DEVELOPMENT MODE

Er dient zur Programmentwicklung

Hardware: DIP Schalter #7 OFF

Beim Start sind die Adreßleitungen A13 bis A15 invertiert

Das Programm

PRELOAD_FLASH erscheint von 00000H bis 0007FH und
HEX_LOADER erscheint von 01700H bis 01DFFH

PRELOAD_FLASH wird ausgeführt und legt das Speicher Mapping folgendermaßen fest:

SYSCON	auf 0E200H	User defined system stack, BHE disabled
REGISTERBANK	von 0F600H bis 0F61FH	Eine Registerbank
USER STACK	von 0F620H bis 0F680H	
SYSTEM STACK	von 0F68CH bis 0F6FFH	

PRELOAD_FLASH ladet dann den HEX_LOADER in das interne RAM

KODE	von 0F700H bis 0FD28H
DATA	von 0FD2AH bis 0FD7FH
128 Byte Puffer	von 0FD80H bis 0FDFFH

Als nächstes wird ein Sprungbefehl auf den HEX_LOADER auf 0F700H durchgeführt, der dann mit dem Befehl EINIT startet und dadurch die Adreßinvertierung aufhebt

Es ergibt sich das Speicher Mapping des FLASH wie im Application mode.

Die Soft- und Hardware befindet sich jetzt im HEX_LOADER STATE.

HEX_LOADER STATE:

Der HEX_LOADER versteht jetzt drei Kommandos:

1.) Eine standard INTEL HEX Datei, die mit dem Zeichen ':' beginnt, wird über die serielle Schnittstelle empfangen und ihre Daten werden in das FLASH programmiert. Die Daten müssen in aufsteigender Ordnung sein und die Löcher müssen aufgefüllt sein
Einstellung in der Keil Entwicklungsumgebunge:
TARGET: Output: Create Hex File: FLASH Fill Byte 0xFF

2.) Kommando daaaaa wird über die serielle Schnittstelle empfangen:
Ein Block des FLASH (128 Byte) wird ab der Adresse aaaaaH zum Host gesendet und kann dort angezeigt werden.

Beispiel: d0E000

Wenn das Kommando d<cr> ist wird der folgende Block zum Host gesendet.

3.) Das Zeichen ’s’ wird über die serielle Schnittstelle empfangen:
Der Controller bekommt ein Software Reset und das geladene Programm startet ab 00000H.

Die Soft-und Hardware ist jetzt im APPLICATION STATE

3.3 BOOTSTRAP MODE

Er dient zur Erstinstallation des HEX_LOADER

Hardware: DIP Schalter #6 ON ; Direct drive zur Oszillatorüberprüfung!
DIP Schalter #7 ON
DIP Schalter #8 ON

Die Hardware startet im Bootstrap Loader mode. Die Adressleitungen sind nicht invertiert.

Der Host sendet ein 00H Byte und empfängt ein Quittierungsbyte (0C5H from the C167).
Dann sendet der Host die binäre Datei LOADER.bin.

Danch befindet sich die Soft- und Hardware im HEX_LOADER STATE.

Um diesen Vorgang durchzuführen kann man das Terminal Programm TeraTerm mit seinem Macro bootload.ttl am PC verwenden.

Genaue Beschreibung des Bootload Vorganges:

Die Datei Loader.bin besteht aus vier Teilen:

Drei Teile SECOND_LEVEL_LOADER
PRELOAD_SERIAL
PRELOAD_FLASH

stammen von der Assembler Datei Bootload.a51,

der vierte HEX_LOADER

stammt von der C Datei Load.c

Der Firmware Bootstrap loader ab 00000H wird ausgeführt und erwartet ein Start-Byte auf der seriellen Schnittstelle. Nach dem Empfang eines 00H Byte wird die Baudrate berechnet und das Byte 0C5H als Quittierung zurückgesendet.

Dann werden die ersten 32 Byte, der SECOND_LEVEL_LOADER empfangen und ab 0FA40H gespeichert. Das Speicher Mapping ist das vorgegebene Bootstrap Mapping, wie im User Manual. beschrieben.

SYSCON ist 0E000H und nicht 0E00H (Fehler im User Manual 2.0!)

Dann führt der Bootstrap Loader einen Sprung auf Adresse 0FA40H aus und der SECOND_LEVEL_LOADER wird ausgeführt.
Dieser ladet den Kode PRELOAD_SERIAL von 0F620H bis 0F6B1H.
Danach wird ein Sprungbefehl auf 0F620H durchgeführt und der Kode von PRELOAD_SERIAL exekutiert..

PRELOAD_SERIAL ändert die Speichereinstellungen auf die exakt gleichen wie oben beschrieben von PRELOAD_FLASH.

PRELOAD_SERIAL empfängt zuerst den Kode von PRELOAD_FLASH und programmiert ihn von Adresse 0E000H bis 0E07FH (eine Bank) in das FLASH.
Dann wird der Kode von HEX_LOADER über die serielle Schnittstelle empfangen und von 0F700H bis 0FDFFH in das FLASH programmiert. Gleichzeitig wird derselbe Kode auf die gleichen Adressen in das interne RAM des C167 geschrieben, damit er sofort startbereit ist.

Um das FLASH (blockweise) programmieren zu können ist ein 128 Byte Puffer im internen RAM nötig, der sich auf den RAM Adressen 0FD80H bis 0FDFFH befindet. (Derselbe Puffer wird nachher vom HEX_LOADER verwendet)
Da auf die FLASH Adressen 0E000H bis 0FFFFH nicht direkt zugegriffen werden kann (überlappt von den internen Resourcen des Controllers) wird das Programmieren mit einem Offset von zwei Kode-Segmenten (z. B. 2E000H anstatt 0E000H) bewerkstelligt.

Zum Schluß führt PRELOAD_SERIAL einen Sprung auf 0F700H durch, der HEX_LOADER wird ausgeführt und das Programm befindet sich im HEX_LOADER STATE.

4 Erstellung der Software mit der Keil Programmierumgebung

4.1 Verzeichnisstruktur

Nach dem Auspacken von Kern.zip in das Stammverzeichnis C:\ (Die Programme enthalten zahlreiche Pfade, die sonst alle editiert werden müssten) erhält man folgende Verzeichnisstruktur im Laufwerk C:

Kern
- Dokumentation
- Dump
- Erase
- Loader
- Lock
- Utilities
- Tterm23.zip

Tterm23.zip kann installiert werden und ist ein Freeware Terminal Programm, mit dem auch binär ohne Protokoll gelesen und geschrieben werden kann, auch ist es einfach, dafür Skrips zu schreiben. Für Dump, Bootload, Erase und Lock befinden sich die Skripts im Verzeichnis Utilities. Aufruf im Terminal Programm: Control>Macro>xxxxx.ttl

Loader ist der oben beschriebene Lader.

Erase löscht das gesamte FLASH (wenn es nicht durch Lock versperrt wurde!)

Lock sperrt im FLASH den Bereich des Laders. (Vorsicht: kann nicht rückgängig gemacht werden!)

Dump schickt den binären Inhalt des FLASH zum Host.
Zuerst in TeraTerm File>LOG>memory.bin in Verzeichnis c:\Dump. Option binary
Dann Control>Macro>Dump.ttl. Warten bis der Cursor steht.
Danach kann mit der Datei C:\Kern\Dump\Hexdump.bat eine Umwandlung in eine Intel Hex Format Datei gestartet werden, bei der auch das erste Zeichen, C5H (die Bootstrap Quittierung) entfernt wird.

Erase, Lock und Dump sind kurze Routinen, die mit dem Bootstrap Loader in das interne Ram eingebracht und danach dort exekutiert werden.
Hardware: DIP Schalter #7 ON
DIP Schalter #8 ON

In den Ordnern Dump, Erase, Loader und Lock befindet sich jeweils ein Projekt, das mit der Keil Entwicklungsumgebung (Demoversion genügt) die exekutierbaren Dateien erzeugt.