

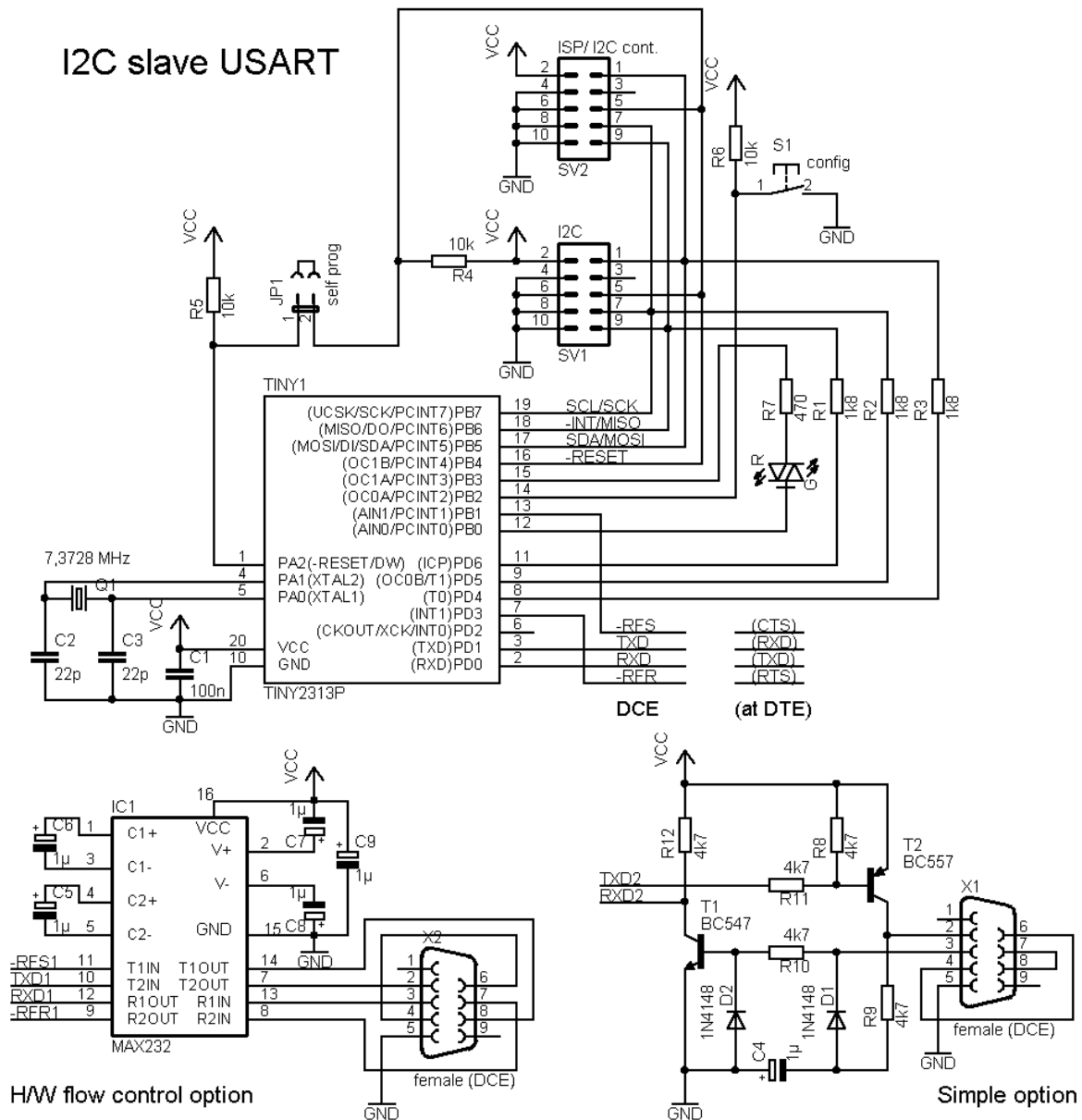
I2C-slave USART

von Klaus2m5

Ein AVR ATTINY2313 als I2C nach RS232 Konverter. Benutzt USART und USI Hardware des AVR, programmiert in Assembler. Erweitert einen Master um jeweils einen USART, begrenzt nur durch die Anzahl der verfügbaren I2C-Adressen und der Bandbreite des I2C-Bus.

Eine spezielle Anwendung besteht darin, einem beliebigen AVR eine temporäre RS232 Schnittstelle zur Verfügung zu stellen, ohne dass dieser einen Baudratenquartz und Schnittstellenhardware benötigt. Mittels des vorhandenen ISP-Steckers kann I2C-Master Software den USART steuern.

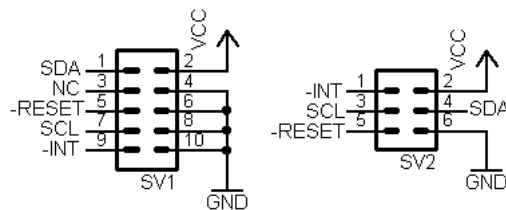
Die Schaltung



Ein Platinenlayout existiert nicht, da man diese Schaltung in vielen Variationen, bis hin zur Integration mit einem I2C-Master und mehreren I2C-Slaves, aufbauen kann.

I2C-Schnittstelle

Wie schon in der Einführung erwähnt, kann die ISP-Schnittstelle auch gleichzeitig als I2C-Schnittstelle verwendet werden. Beim 2313 liegen auch gleich die USI Signale SCL und SDA auf den gleichen Pins. Über das bei I2C nicht benötigte MISO wird das Interrupt Signal -INT ausgegeben (Daten im Empfänger-Puffer). Die Überwachung des -Reset Pins der ISP Schnittstelle erlaubt den gleichzeitigen Anschluss eines ISP-Programmers. Wenn der Programmer aktiv ist, duckt sich die I2C Schnittstelle nebst der schaltbaren Pullups, so dass der Master weiterhin programmiert werden kann.



Optional kann die I2C Schnittstelle auch mit nur einem Stecker und einem T-förmigen Kabel aufgebaut oder direkt auf derselben Platine mit dem Master verbunden werden. Der 10-Pin Stecker kann durch einen 6-Pin Stecker unter Beibehaltung der ISP-Pinbelegung ersetzt werden. Die Pullups sind nur einmal am I2C-Bus notwendig.

RS232-Schnittstelle

Die RS232-Schnittstelle kann in zwei Varianten realisiert werden. Die erste Variante besteht aus einem MAX232 IC nebst externer Beschaltung und erlaubt eine Hardware-Flusskontrolle, da die Signale -RFR/-RFS (ready for sending/receiving, besser bekannt als RTS/CTS) bereitgestellt werden. Die zweite Variante ist ein einfacher Pegelumsetzer, wie er zum Beispiel aus Klaus Leidingers AVR-Programmer bekannt ist. Eine dritte Möglichkeit wäre, einen USB zu TTL-RS232 Konverter zu verbauen.

Sonstiges

Natürlich muss bei Benutzung der USART Schnittstelle ein Baudratenquarz eingesetzt werden. Die Stromversorgung des AVR erfolgt über den ISP/I2C Stecker. Weitere optionale Elemente sind:

- Ein Jumper zum Programmieren des 2313 (nur ein AVR am Programmer!)
- Eine Duo-LED zur Anzeige des Betriebszustands
- Ein Taster zum Umschalten in den Debug-/Konfigurationsmodus

AVR programmieren

Empfohlene Fuse-Einstellungen (Ext=0xFF, High=0x9B, Low=0xFD):

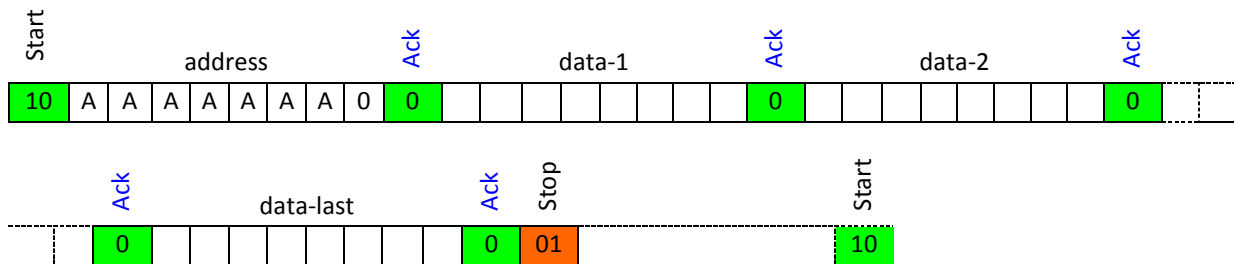
- EEPROM protected during Chip Erase
- Brown-out detection enabled at 2,7V
- External Crystal Osc., Frequency 3-8 MHz (Quartz=7,3728 MHz) , Start-up 14CK+65ms
- Divide Clock by 8 (CKDIV8) off!!!

Anschließend I2C_to_RS232.hex in den Flash und I2C_to_RS232.eep ins EEPROM laden.

Grundfunktionen

- I2C-Slave nach RS232 Konverter
- Jeweils 48 Bytes Empfangs- und Sendepuffer
- Optionale Hardware und/oder Xon/Xoff Flusskontrolle
- Viele Optionen konfigurierbar, ohne neu zu flashen
- Bei I2C-Aktivität leuchtet die LED rot, ansonsten grün
- Wenn -RESET low ist, wird die I2C Schnittstelle inklusive der Pullups abgeschaltet (High-Z)

Senden



Der I2C-Master startet den Sendevorgang, indem er

- Die Startbedingung überträgt (SDA high to low bei SCL high)
- Die Schreibadresse des Konverters sendet (Default=0x0C)
- Das Acknowledge des Konverters abwartet

Danach können die zu sendenden Daten übertragen werden. Nach jedem Byte antwortet der Slave mit Ack. Wenn der Sendepuffer voll ist, hält der Slave den Master durch Clock-Stretching (Slave setzt SCL low) an. Sobald der Sendepuffer wieder Platz hat, wird das Clock-Stretching aufgehoben. Der Sendepuffer wird kontinuierlich und unter Berücksichtigung der Flusskontrolle geleert.

Wenn nach der Schreibadresse kein Acknowledge empfangen wird (Nak), ist der Slave im Debug- & Konfigurationsmonitor (I2C-Schnittstelle abgeschaltet) oder die Adresse stimmt nicht.

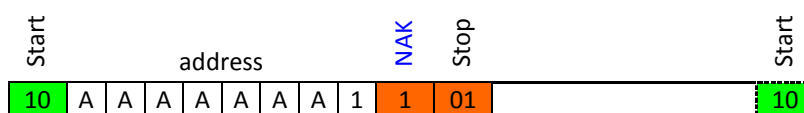
Der Master kann das Senden jederzeit durch eine Stop-Bedingung oder eine Repeated-Start-Bedingung beenden.

Empfangen

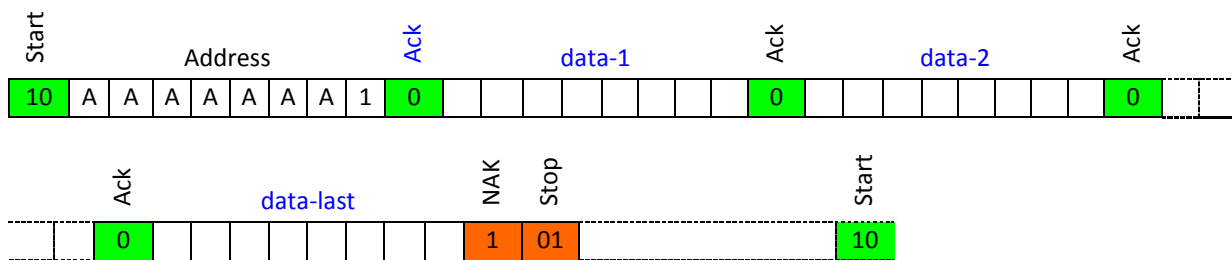
Zunächst muss der Master wissen, ob überhaupt Daten im Empfangspuffer sind. Darüber wird er durch das -INT Signal informiert. Wenn Daten empfangen wurden, setzt der Slave -INT auf low. Wenn der Puffer geleert wurde, Setzt der Slave -INT wieder auf high.

Alternativ und wenn mehrere Slaves das -INT Signal bedienen, kann der Master an der Ack/Nak Antwort, nachdem er die Leseadresse gesendet hat, erkennen, ob Empfangsdaten vorhanden sind.

Wenn keine Daten vorhanden sind:



Wenn Daten vorhanden sind:



Wenn der Master -INT=low sieht oder er den Slave auf Empfangsdaten abfragen möchte (Polling), startet er die Datenübertragung, indem er

- Die Startbedingung überträgt
- Die Leseadresse des Konverters überträgt (Default 0x0C+1)
- Das Ack/Nak des Slave auswertet
 - Bei Nak die Übertragung durch Stop oder RepStart beendet
 - Bei Ack die Übertragung fortsetzt

Der Master liest ein Byte und sendet ein Nak und Stop (oder RepStart), falls keine weiteren Daten gelesen werden sollen. Kann das -INT Signal nicht ausgewertet werden, ist dies die einzige Möglichkeit, durch ein erneutes Ack/Nak nach der Leseadresse festzustellen, ob noch weitere Daten im Empfangspuffer warten. Wenn -INT weiterhin low ist, kann der Master auch ein Ack senden und das nächste Byte lesen. Ist der Empfangspuffer leer und der Master liest trotzdem weiter, so bekommt er ein Nul-Character (0x00) zurück.

Erweiterungen

Drückt man den Taster an PB2 des AVR, kommt man in den Debug- & Konfigurationsmonitor. Die LED leuchtet dauernd rot und die I2C Schnittstelle ist abgeschaltet (Der Master bekommt ein Nak, wenn er die Slave Adresse anspricht). Der Monitor läuft über RS232 und meldet sich dort mit seinem Prompt „I2R>“. Er bietet die Möglichkeit, den Inhalt von GPRs, I/O-Registern, SRAM und EEPROM auszulesen und zu verändern. Man kann einen Reset auslösen (benutzt den Watchdog) oder einen Call oder Jump zu einer Flash-Adresse machen. Ein einfacher I2C-Sniffer erlaubt die Überwachung des angeschlossenen I2C-Buses. Mit den I2C-Master Funktionen kann man I2C-Slaves testen.

Monitor Commands

M	auslesen des Memoryspace inklusive GPRs und I/O-Register
E	auslesen des EEPROM
W aaa:dd	schreiben der Daten dd auf die Adresse aaa
P aaa:dd	programmieren der Daten dd ins EEPROM an Adresse aaa
C aaa	Unterprogrammaufruf der Flash-Adresse aaa
J aaa	Sprung zur Flash-Adresse aaa
R	Reset des Slave (benutzt Watchdog)
X	Monitor beenden und Grundfunktion aktivieren
S	Startet den I2C-Sniffer
I	Startet das I2C-Master Tool

Alle Commands können mit <ESC> abgebrochen werden. Vollständig eingegebene Bytes sind jedoch bereits geschrieben. Adressen werden ohne führende Nullen eingegeben. Der Monitor unterdrückt Adressen, die größer als der jeweilige Adressraum sind. Die Eingabe der Adresse kann durch <CR> oder Doppelpunkt abgeschlossen werden. Daten werden fortlaufend eingegeben, beendet durch <CR>.

I2C-Sniffer

Der Sniffer beobachtet die I2C-Schnittstelle und stellt den beobachteten Datenverkehr als RS232-Ausgabe da. Die Geschwindigkeit der Aufzeichnung wird allerdings durch den kleinen Ausgabepuffer und die Baudrate der seriellen Schnittstelle beschränkt und sollte nur als Notbehelf gesehen werden, um einfache Probleme der I2C-Schnittstelle zu beheben.

Der Sniffer verwendet folgende Zeichen, um den Datenverkehr darzustellen:

- > Start oder Repeated Start
- < Stop
- w Scheibadresse folgt
- r Leseadresse folgt
- + Acknowledge (Ack)
- Kein Acknowledge (Nak)

Beispiel: >wA0+01+20+>rA1+FF+FF-<

Der Master hat Adresse A0 zum Schreiben geöffnet und der Slave hat sich mit einem Ack gemeldet und damit bestätigt, dass er empfangsbereit ist. Danach überträgt der Master 2 Bytes. Der Slave bestätigt jeweils mit Ack, dass er weiterhin empfangsbereit ist. Der Master sendet ein Repeated Start, jetzt mit einer Leseadresse zum gleichen Slave (0xA0+1). Der Slave bestätigt mit Ack, dass Daten gelesen werden können und sendet das erste Byte an den Master. Der Master antwortet mit Ack um ein weiteres Byte zu lesen. Nach dem zweiten Byte beendet er die Übertragung mit Nak und Stop.

Was wir gerade gesehen haben, war das Lesen eines I2C-EEPROMs. Zunächst wurde die Adresse übertragen und dann zwei Bytes, beginnend ab Adresse 0x0120, gelesen.

I2C-Master

Das I2C-Master Tool benutzt eine ähnliche Syntax wie der Sniffer, um mit Slaves zu kommunizieren. Um das Beispiel vom Sniffer umzusetzen, muss man folgende Eingaben machen (**Ausgabe**).

>A0+01+20+>A1+**FF**+**FF**-<

Man kommt bei der Eingabe nur mit wenigen Steuerzeichen aus. Das erste Byte wird immer als Adresse gewertet.

- > Repeated Start (der erste Start wird automatisch erzeugt)
- <CR> Stop (mit vorangestelltem Nak bei read)
- + Acknowledge (Ack) bei read

Konfiguration

Alle einstellbaren Optionen haben reservierte EEPROM-Adressen und können mit Hilfe der Monitor-Commands E & P leicht geändert werden. Die Änderungen werden nach einem Reset wirksam.

```
I2R>E
00:0B0C01FF
```

Baudrate

Die Baudrate wird in Form des Ladewertes für UBRRL auf der EEPROM-Adresse 0x00 abgelegt. Der voreingestellte Wert ist 0x0B und entspricht 38400 Baud bei 7,3728 MHz. UBRRH ist immer 0x00 und der USART immer auf 8N1 voreingestellt. Die Baudrate kann auf folgende Werte geändert werden:

0xbf	=2400 Baud
0x2f	=9600 Baud
0x17	=19200 Baud
0x0b	=38400 Baud
0x07	=57600 Baud
0x03	=115200 Baud

Da man sich dabei schnell mal in den Fuß geschossen hat, kann man durch langes (3s) Drücken und Halten der Taste an PB2 die Baudrate auf die Voreinstellung von 38400 zurücksetzen. Ohne Taster müsste man dann PB2 für 3s erden.

I2C-Adresse

Die I2C-Adresse befindet sich im EEPROM auf Adresse 0x01. Hier muss eine gültige I2C-Write-Adresse eingetragen sein. Keine reservierten und keine ungeraden Adressen verwenden! Die Voreinstellung ist 0x0C.

Sonstige Optionen

Die sonstigen Optionen befinden sich im EEPROM an der Adresse 0x02 in Form einer Bitmap.

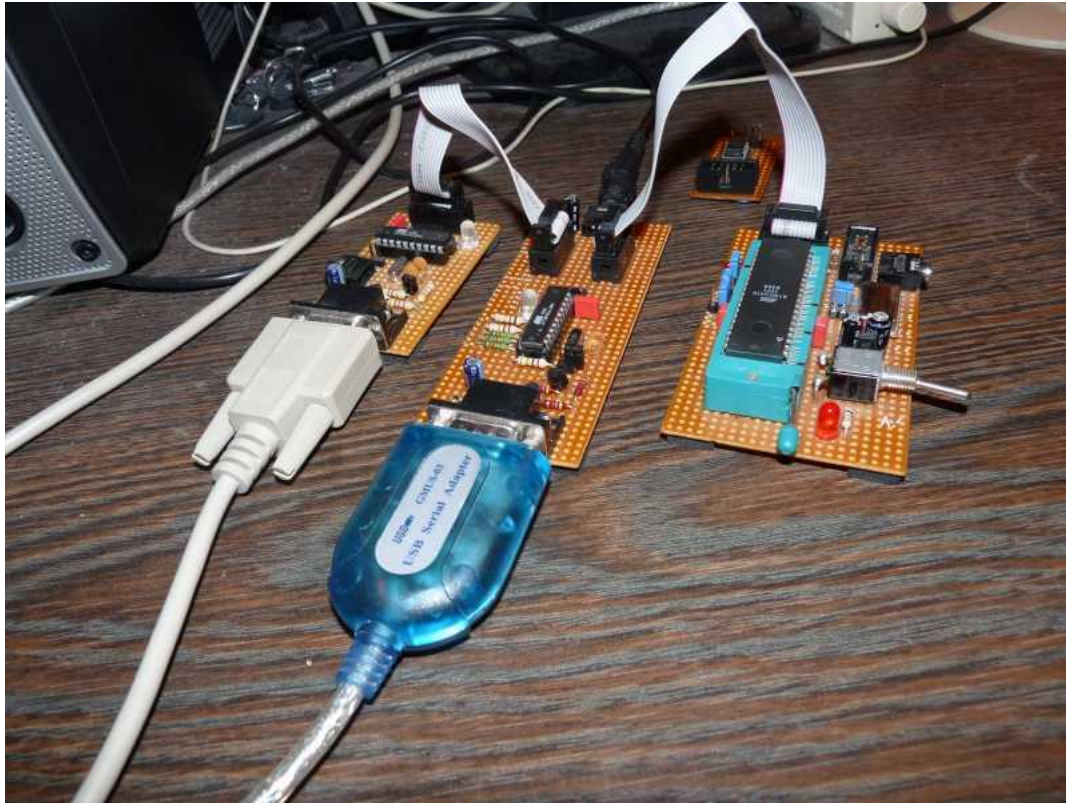
Bit 0 0b00000001	Pullups einschalten. Schaltbare Pullups sind die Voraussetzung, um einen Programmierer parallel zum I2C-Bus zu betreiben.
Bit 1 0b00000010	Debug/Configuration Monitor nach Reset starten. Sinnvoll, wenn der Konverter nur als I2C-Tool eingesetzt werden soll.
Bit 2 0b00000100	Meldungen unterdrücken. Die Einschaltmeldung mit der Versionsangabe und die ISP-Reset-Meldung werden unterdrückt. Für produktive Umgebungen.
Bit 3 0b00001000	Xon/Xoff-Flusskontrolle aktivieren. Nur für Übertragung mit reinem ASCII-Zeichensatz möglich (Xon/Xoff werden aus dem Datenstrom gefiltert).
Bit 4 0b00010000	Hardware Flusskontrolle aktivieren. Erfordert die Belegung von PD3 mit dem -RFR (RTS) der DTE, sonst hängt der Transmitter.

Sofern die jeweilige Option eingeschaltet ist, stoppt die Flusskontrolle den Transmitter, wenn -RFR high ist oder ein Xoff empfangen wurde. Wenn -RFR wieder auf low geht oder wenn Xon empfangen wurde, wird der Transmitter wieder eingeschaltet. Wenn der Empfangspuffer mit 24 Bytes oder mehr gefüllt ist wird -RFS auf high gesetzt oder Xoff gesendet. Wenn der Empfangspuffer nur noch 16 Bytes oder weniger enthält, wird -RFS auf low gesetzt oder Xon gesendet.

Die Voreinstellung ist 0b00000001 (0x01) und entspricht Pullups ein, alle anderen Optionen aus.

Anwendungsbeispiel

Um die Anwendungsmöglichkeiten zu verdeutlichen, befindet sich im Ordner Debugger eine Sammlung von „Debug Skeleton *.asm“ Dateien, die an verschiedene AVRr angepasst sind. Diese stellen ein Gerüst für die Anwendungsentwicklung mit I2C-Master Zugriff über die ISP-Schnittstelle auf RS232 Ein-/Ausgabe dar. Sie enthalten einen, dem I2C-Slave USART Debug- und Konfigurationsmonitor ähnlichen Monitor, der unter anderem den Zugriff auf die Speicher und Register des AVR erlaubt.



AVR 910 Programmer

I2C-Slave USART

Mega16 als Master

Im Bild sieht man einen Versuchsaufbau, bei dem auf einem Mega16 auf Basis des Debug-Skeletons eine Anwendung entwickelt werden kann. An seinem ISP-Stecker sind der I2C-Slave USART und ein Programmer (durchgeschleift am Slave) parallel angeschlossen. Damit der Programmer parallel zum I2C-Slave arbeiten kann, darf der Self-Programming Jumper des I2C-Slave nicht gesetzt sein. Auf der PC-Seite werden hier zwei COM-Schnittstellen benötigt, was zunächst als Nachteil erscheint. Der Vorteil ist dabei, dass Terminal-Programm und Programmer-Software parallel geöffnet bleiben können. Während des Programmiervorgangs erscheint auf dem Terminal:

```
-- ISP/I2C Reset --
```

Debug Skeleton Monitor

Der Monitor wird durch <ESC> aktiviert. Es erscheint der Monitor-Prompt:

- G> = Go - Interrupts erlaubt und Main-Loop wird ausgeführt
- H> = Halted - Interrupts erlaubt und Main-Loop wird nicht ausgeführt
- DH> = Disabled and Halted - Interrupts verboten und Main-Loop wird nicht ausgeführt
- DG> = Erscheint nur, wenn Interrupts nie erlaubt wurden

DSM Commands

M	(Tiny) auslesen des Memoryspace inklusive GPRs und I/O-Register
E	(Tiny) auslesen des EEPROM
M aaa	(Mega) auslesen des Memoryspace ab aaa für 0x100 Bytes
E aaa	(Mega) auslesen des EEPROM ab aaa für 0x100 Bytes
W aaa:dd	schreiben der Daten dd auf die Adresse aaa
P aaa:dd	programmieren der Daten dd ins EEPROM an Adresse aaa
C aaaa	Unterprogrammaufruf der Flash-Adresse aaaa
J aaaa	Sprung zur Flash-Adresse aaaa
H	Halt - Ausführung des Main-Loop unterbrechen, Interrupts erlauben
D	Disable & Halt - Main-Loop unterbrechen und Interrupts verbieten
G	Go - Main-Loop im Monitor weiter fortsetzen und Interrupts erlauben
R	Reset des Slave (benutzt Watchdog)
X	Monitor beenden, Main-Loop fortsetzen und Interrupts erlauben

Alle Commands können mit <ESC> abgebrochen werden. Vollständig eingegebene Bytes sind jedoch bereits geschrieben. Adressen werden ohne führende Nullen eingegeben. Der Monitor unterdrückt Adressen, die größer als der jeweilige Adressraum sind. Die Eingabe der Adresse kann durch <CR> oder Doppelpunkt abgeschlossen werden. Daten werden fortlaufend eingegeben, beendet durch <CR>.