Überarbeitung November 2011

Mit Hilfe dieses Tools lassen sich Kommandos vom PC über ein Terminalprogramm (wie HTerm) via RS232-Schnittstelle auf einen TWI-Bus übergeben und auf diesem Wege TWI-Slaves "manuell" bedienen. Das ist hilfreich beim Testen von selbst programmierten TWI-Slaves oder auch beim Erforschen von TWI-Geräten wie Beschleunigungssensoren, Drucksensoren und was es alles heute so gibt.
Die Eingabe erfolgt über die Tastatur, das Tastaturecho sowie die Rückmeldung wird im Terminalprogramm angezeigt.
Grundsätzlich können auch Textdatei (auch aus mehreren Zeilen bestehend) versandt werden.

Die Funktionsweise des Programms
Nach dem Start wartet das Programm im Idle-Modus auf einen Interrupt:
entweder USART_Rx_Complete oder TWI.

Der USART_Rx_complete wird ausgelöst, sobald die Serielle Schnittstelle ein Byte empfängt.
Das eingegangene Byte wird im rs232_buf[] abgelegt und es wird geprüft, ob das Zeichen ein RETURN ist.
Wenn ein RETURN erkannt wurde, dann ist eine vollständige Eingabezeile empfangen worden.
Nun wird der Eingangspuffer analysiert, es werden Ziffern zu Zahlen zusammengesetzt bis eine NICHT Ziffer erkannt wird.
Ab diesen Zeitpunkt werden die Bytes 1:1 in den Ausgabepuffer write_buf[] geschrieben.
Ist der gesamte Eingabepuffer gelesen, wird der Ausgabepuffer an den TWI-Master übergeben, der dann Byte für Byte per TWI versendet.
Immer wenn ein Byte verschickt ist, wird ein TWI Interrupt aufgerufen und in der TWI_ISR das nächste Byte auf den TWI-Bus geschrieben - solange, bis alle Daten verschickt sind.
Wird auf einen TWI-Slave lesend zugegriffen, dann wird die Antwort über einen separaten Puffer read_buf[] abgeholt, der write_buf[] bleibt unverändert.
Zwischendrin geht das Programm immer wieder in den Idle-Modus über - und spart Stromkosten.

Parallel zum Versenden der TWI-Daten können bereits die nächsten Bytes über die serielle Schnittstelle in den Empfangspuffer eingelesen werden.

Die Bedienung
Die Eingabe beginnt notwendigerweise mit der TWI-Adresse des anzusprechenden Slaves.
Danach folgen die zu sendenden Daten und ein abschließendes RETURN (es MUSS ein abschließendes CHR13 an das Programm gesendet werden, damit der Abschluss der Eingabe erkannt wird.
In HTerm muss dazu unter "Send on Enter" die Option "CR" gewählt werden).

Jeweils bis zu 3 Ziffern einer Dezimalzahl werden vom Programm zu einem Byte gefügt.
Ein ',' trennt mehrere Ziffernfolgen (Bytes) voneinander ab.
Das Komma kann nach der letzten Zifferfolge vor einem String entfallen (sobald ein Zeichen (!isdigit()) eingeben wird, wird der nachfolgende Rest als Textstring interpretiert).

Ein Beispiel:
64,255,10,abcd (+return) oder
64,0xFF,10abcd (+return) -> sendet die Bytes: 0x40 0xFF 0x0A 0x61 0x62 0x63 0x64

Ziffernfolgen werden grundsätzlich durch Kommata voneinander abgegrenzt.

ein '+' behält an dieser Position im Sendepuffer den letzten Wert bei:
64,255 (dies sei die letzte Eingabe) sendet 0x40 0xFF
++ sendet 0x40 0xFF (also noch einmal dieselbe Bytefolge)
+11 sendet 0x40 0x0B
66+ sendet 0x42 0x0B

Anmerkung aus Sicht 2011:
Das mit dem '+' ist eigentlich überflüssig, da man im Terminalprogramm mit den Cursortasten in der Historie wesentlich komfortabler zurückblättern und editieren kann.

Wenn das Bit.0 der TWI-Adresse gesetzt ist (das Read-Bit), dann werden soviele Bytes vom Slave gelesen, wie hinter der Adresse eingegeben werden (mindestens jedoch 1 Byte).
Wird hinter der Adresse eine Ziffernfolge eingegeben, dann wird diese in ein Byte umgewandelt, das als Anzahl der einzulesenen Zeichen gewählt wird.
Zur Vollständigkeit sei folgendes Verhalten erwähnt:
Wird ein einzelnes Zeichen nach der Adresse eingegeben, das nicht als Zahl interpretiert werden kann, dann wird der ASCII-Wert des Zeichens als Zahl der einzulesenenden Zeichen gewählt.

64,abcd schreibt die Bytes a,b,c,d an Adresse 64
65,abcd liest 4 Byte von Adresse 65
65,4 liest ebenfalls 4 Byte von Adresse 65
65 liest genau 1 Byte von Adresse 65
65A diese Eingabe wählt den ASCII-Wert von 'A' als Anzahl der einzulesen Zeichen

Ein "\" (z.B. am Ende der Adresse) schaltet um in den "Textmodus", in dem die ASCII-Eingabe ohne weitere Interpretation als Zeichenfolge übernommen wird.
(Dadurch kann eine Texteingabe auch mit Ziffern beginnen, ohne dass diese zur Zahl umgewandelt werden.)
Der erste vorgefundene "\" in einer Zeile schaltet in den Textmodus und setzt ein Flag.
Der zweite "\" gibt dann ein ASCII 0 aus und beendet das weitere Einlesen dieser Zeile, nachfolgende Zeichen werden ignoriert (können somit ggf. in Textdateien als Kommentar genutzt werden).
Zur Vereinfachung gibt es noch die Regel:
Ein "\" als letztes Zeichen einer Eingabe liefert ein ASCII 0.
Die '0' ist z.B. zur Terminierung von Textstrings erforderlich, die man in ein Eeprom schreiben will.

Ein Beispiel:
64\1234\ sendet den String "1234" an das Gerät 64 und schließt die Eingabe mit CHR(0) ab.
64abcd\ sendet den String "abcd" mit abschließendem CHR(0).
64ab\cd sendet den String "abcd", der '\' tut nichts sichtbares (schaltet nur das CHR0_Flag ein).
64ab\cd\e\f\ gibt nach dem d (dem 2. Backslash) eine 0 aus und beendet die Ausgabe.
Beschränkung: es können maximal MAX_BUFFER -1 Byte (ca. 220) eingelesen werden.
Eingabezeilen müssen demzufolge kürzer als die Konstante MAX_BUFFER sein.

Es können auch Textdateien übertragen werden (z.B. zum Beschreiben eines Eeproms):
sinnvolles Format:
162,000,000\abcd\ -> Adresse-TWI, Adresse-High-Byte, Adresse-Low-Byte\Daten[\]->CHR(0)
Beispiel:
162,000,096\Menue1\ schreibt den Text "Menue1" mit anschl. CHR(0) an Adresse 96
Alle Zeilen sollten aber gleich (bzw. ähnlich) lang sein !
Bei kürzer werdenden Zeilen ist die Übertragung einer vorausgehenden (längeren) Zeile über den TWI-Bus eventuell noch nicht abgeschlossen - das hängt natürlich von der Baud-Rate und dem TWI-Takt ab.

Beim Senden von Textdateien ist ein potentielles Timing-Problem zu beachten.
In der Zeit zwischen einem RETURN und dem ersten Zeichen der folgenden Zeile muss das Programm die gesamten gepufferten Daten interpretieren, vom rs232_buf[] in den write_buf[] schreiben und dieses Array an das TWI-Modul übergeben.
Erst dann ist es wieder bereit, neue Daten zu empfangen.
D.h. die Baud-Rate sollte nicht zu hoch und FCPU / TWI-Takt nicht zu niedrig sein.
Bei 9600 Baud und 100KHz TWI-Takt habe ich ohne Probleme Dateien mit 30 Zeichen/Zeile übertragen.
Sollten Daten verloren gehen, dann muss man an den oben benannten Schrauben drehen ...

Tödlich ist das TX-Echo (das Zurücksenden der empfangenen Daten ans Terminal).
Dieses Echo verhindert definitiv das erfolgreiche Übertragen von Textdateien.
Also vor der Übertragung von Textdateien bitte ausschalten (-e).

(Anmerkung: In der Eingabeaufforderung (cdm.exe) kann man über "copy textdatei.txt COMx:" kopieren, sofern vorher die BAUD-Rate der Schnittstelle entsprechend des Empfängers eingestellt worden ist.
Hterm hat eine eingebaute Option zum Kopieren (Senden) von Dateien.)

Das Zahlenformat kann während der Eingabe beliebig gewählt werden:

134	- dezimale Eingabe bis dreistellig wird in Byte umgewandelt (Eingabe % 256), d.h. die Eingabe 260 wird zu 4 !
255	- liefert 255
256	- liefert aber 1 (Eingabe % 256 !)
0xff	- Hexadezimaldarstellung für ein Byte
0b11111111	- binäre Eingabe für ein Byte
0w12345	- dezimale Eingabe bis fünfstellig für ein Word (damit lassen sich auch 16bit-Werte wie EEprom-Adressen ohne Umrechnung eingeben)
0w1	- liefert zwei Byte zurück: 0x00, 0x01
0w1000	- liefert zwei Byte zurück: 3, 232

Aufruf von eingebauten Befehlen, es wird als erstes Zeichen "-" vorangestellt:

-0	Taktet den TWI-Bus durch Einstellen von TWBR auf 10 (kleinster Wert für m8)
-1	Taktet den TWI-Bus durch Einstellen von TWBR auf 20
-2	Taktet den TWI-Bus durch Einstellen von TWBR auf 40
-3	Taktet den TWI-Bus durch Einstellen von TWBR auf 64
-4	Taktet den TWI-Bus durch Einstellen von TWBR auf 128
-5	Taktet den TWI-Bus durch Einstellen von TWBR auf 255 Der resultierende TWI-Takt ist abhängig vom CPU-Takt
-x	Taktet den TWI-Bus auf 100KHz, das ist die default-Einstellung beim Programmstart
-b|d|h	Wählt als Ausgabeformat die binäre, dezimale oder hexadezimale Darstellung
-e	toggle Tx-Echo (default EIN) , gibt die Tastatureingabe auf dem Terminal aus
-l	sucht nach TWI-Geräten und zeigt die Adressen der gefundenen Geräte an
-s	Darstellung als TWI-Slave empfangener Daten: toggle ASCII-/Numerische Darstellung
-?	gibt diese Hilfe aus

[Überarbeitung 13.11.11]
Fehlerbeseitung: Bei -x wurde ein falscher TWI-Takt angezeigt.

Während der Phasen der Untätigkeit bietet das Programm nun als TWI-Slave seine Dienste an.
Die da wären:
Vom Master des TWI-Busses Daten entgegenzunehmen und über die serielle Schnittstelle an ein Terminal zu senden.
Dies ist insbesondere dann nützlich, wenn die serielle Schnittstelle des TWI-Masters bereits anderweitig belegt ist, aber die Ausgabe von Debug-Daten etc. gewünscht ist.
Oder wenn Daten - die in der finalen Lösung an einen TWI-Datenlogger gesendet werden - während der Testphase zur Begutachtung an das Terminal geschickt werden sollen.

Als Ausgabeformat der Daten kann zwischen der ASCII- oder einer numerischen Darstellung gewählt werden.
Letztere bietet die Option der Ausgabe in der Dezimal-, Hexadezimal- oder Binärdarstellung an.

Die Umsetzung erfolgt so:
Das TWI-Modul wurde um einen TWI-Slave erweitert. Der TWI-Slave-Modus ist der Standardmodus des Programmes. Nur wenn Daten per TWI versandt werden, wird in den TWI-Master-Modus geschaltet.

Während das Programm bisher ausschließlich auf Eingaben über die serielle Schnittstelle (vom PC her) wartete, wird nun der Controller vor dem Aktivieren des Sleep-Modus zum TWI-Slave degradiert.
Wenn ein TWI-Interrupt den Sleep-Modus unterbricht, dann wird in der TWI_ISR geprüft, ob Daten versendet werden (weil das Modul im Master-Modus ist) oder ob der TWI-Slave ein Byte empfängt (weil das Modus im Slave-Modus arbeitet).
Anschließend wird in main() geprüft, ob der TWI-Slave einen vollständiger Datensatz per TWI empfangen hat.
Wenn ja, dann wird dieser über die Serielle Schnittstelle ausgegeben.
Wenn nein, dann geht das Programm wieder in den Idle-Modus über.

Der Rx-Interrupt bewirkt die Aktionen wie bisher und oben beschrieben.

Ein TWI-Interrupt im Slave-Modus (sofern der Slave mit seiner TWI-Adresse angesprochen wurde) bewirkt, dass alle eingegehenden Datenbytes bis zu einem TWI-Stop eingelesen und anschließend über die Serielle Schnittstelle (an den PC) weitergereicht werden.

Die Kommando '-s' wurde ergänzt:
Die Ausgabe der als TWI-Slave empfangenen Daten kann zwischen ASCII- und Numerischer Darstellung der empfangenen Daten umgeschaltet werden.
Die TWI-Adresse des Slaves ist in der global.h vereinbart und per default auf 0x04 gesetzt.

Grundsätzlicher Hinweis:
Das RS232_2_TWI Interface agiert zeitweise als Master im System.
Unproblematisch ist das, solange alle anderen Busteilnehmer Slaves sind.
Ist im Bus jedoch noch ein weiterer Master vorhanden, dann kann es - bei gleichzeitigem Zugriffsversuch beider Master - zu Problemen kommen, da die Behandlung von Arbitrierungsfehlern im Multimasterbetrieb nicht vorgesehen ist.

Manchmal gibt es Mißverständisse wegen der unterschiedliche Interpretation der TWI-Adress-Angaben: der eine spricht ein Gerät mit Adresse 32 an, der andere dasselbe Gerät mit Adresse 64.

Der erstere redet von den TWI-Adressen im Bereich von 0 bis 127, die jeweils um ein Bit nach links geschoben werden müssen.
Zum Schreiben muss dann das (beim Shiften) freigewordene Bit.0 gesetzt werden.

Eingängiger ist mir die andere Lesart:
Die Definition der geradzahligen Adressen 0, 2, 4 ... 252, 254 als Adressen für ein TWI-Write, die jeweils um 1 incrementierte Adresse für ein TWI-Read auf demselben Gerät.
Das Ergebnis beider Denkweisen ist am Ende identisch und macht das Shiften der Adresse überflüssig.

Notwendige Hardware
Das Programm läuft auf einem ATMEGA8. Wenn einige Registernamen angepasst werden, dann natürlich auch auf ATMEGA88/168, der Speicherbedarf bei ca. 6.5kb.

Wegen des Einsatzes der Seriellen Schnittstelle ist ein Quarz erforderlich (+ der beiden Kondensatoren).
SDA und SCL der TWI-Schnittstelle benötigt jeweils ein Widerstand 4.7k als Pullup nach VCC.
RX und TX der Seriellen Schnittstelle können direkt auf einen USB-RS232-Adapter geschaltet werden (der ggf. auch die Stromversorgung übernehmen kann).
Alternativ muss ein MAX232 o.ä. eingesetzt werden, wenn noch die klassische Serielle Schnittstelle mit dem +/-12V Pegel im Einsatz ist.

Unterm Strich hält sich der Hardwareaufwand also in erschwinglichen Grenzen.

Hilfreich ist übrigens, wenn an der SDA- und SCL-Leitung LowCurrentLeds gegen VCC geschaltet werden, dann kann man sofort erkennen, wenn der Bus klemmt, weil irgendein TWI-Lümmel eine Leitung auf gnd zieht und die Kommunikation blockiert: das führt zu Dauerleuchten an den LED's.

Michael S.