hochschule für angewandte wissenschaften
fachbereich für elektrotechnik und informatik **hamburg**
university of applied sciences

Studienarbeit

Marco Neubauer

Anschluss eines Bluetoothmoduls an das Aktor- und Sensormodul

Studiengang: Technische Informatik
Betreuender Prüfer: Prof. Dr. rer. nat. Gunter Klemke
Abgegeben am 20. Januar 2004

BERLINER TOR 3 · D-20099 HAMBURG

Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung

Ziel dieser Studienarbeit ist es, einen Überblick über die Funktionen des Aktor- und Sensormodules (kurz AkSen) zu geben. Das AkSen (siehe Abbildung 1) wurde als Erweiterungsmodul, für das RCUBE-Projekt der FH-Brandenburg entwickelt. Die Hauptaufgabe des AkSen ist der Anschluss der Aktoren und Sensoren an das RCUBE-System.

Der zentrale Bestandteil des RCUBE-Systems ist ein CPU-Modul, das mit einem StongArm Pozessor bestückt ist. Der Prozessor hat eine Taktrate von 200 MHz und ist mit einem 32 MB grossen Hauptspeicher ausgestattet. Die Kommunikation zwischen den einzelnen Modulen erfolgt über eine CAN-Schnittstelle. Das AkSen kann aber auch zum Bau, kleiner reaktiver Systeme benutzt werden.

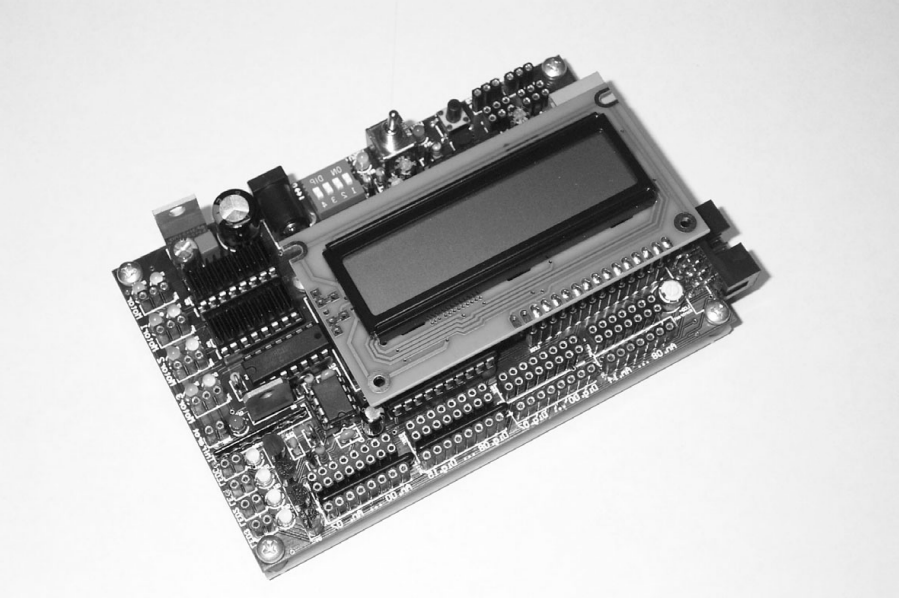

Abbildung 1: Aktor- und Sensormodul

Damit mehrere dieser AkSen Module kollektiv zusammenarbeiten können, müssen untereinander Daten ausgetauscht werden. Zu diesem Zweck bietet sich Bluetooth als Funktechnik an. Der Hauptteil dieser Stundienarbeit beschäftigt sich mit dem Anschluss eines Bluetoothmoduls an das AkSen.

Dies dient dazu, eine Funkverbindung aufzubauen, über die dann einfache Befehle an das AkSen geschickt werden. Dabei könnte es sich um einfache Befehle , wie „vowärts fahren“ oder „rückwärts fahren“ handeln, oder auch um komplexere Befehle, wie z. B. „fahre zu Position X Y“. In dieser Studienarbeit werden nur einfache Befehle implementiert, um die Verbindung zu testen. Außerdem wird es möglich sein, den Programmspeicher des Moduls über die Funkverbindung zu programmieren, sodass auf eine störende kabelgebundene Verbindung verzichtet werden kann.

2 Hardware

2.1 Aktor- und Sensormodul

Auf dem AkSen wird ein 8-Bit Mikrocontroller von Infineon vom Typ 80C515A[2] eingesetzt. Der 80C515A ist zum 8051 von Intel kompatibel. Die Taktfrequenz des Prozessors beträgt 12 MHz. Als Speicher wird ein PSD934F2V von STMicroelectronics eingesetzt. Dieser enthält einen Main Flash Speicher mit 256 kB und einen Secondary Flash Speicher mit 32 kB. Der Secondary Flash Speicher wird für den Bootloader benutzt, mit dem der Main Flash Speicher über die serielle Schnittstelle beschrieben werden kann. Vom Main Flash Speicher werden nur die ersten 64 kB als Programmspeicher benutzt, was den gesamten Adressbereich des Microcontrollers abdeckt. Der Programmspeicher ist in zwei Teile mit jeweils 32 kB aufgeteilt. Der erste Bereich wird für Bibliotheksfunktionen benutzt, wie zum Beispiel Funktionen zur Ansteuerung der Motoren. Der zweite Bereich steht als Speicher für eigene Programme bereit. Zusätzlich enthält der Chip einen CPLD[1], der als Adressdekoder benutzt wird. Es stehen eine Reihe externer Anschlussmöglichkeiten zu Verfügung:

- 16 digitale Ein-/Ausgänge
- 15 analoge Eingänge
- 4 Motortreiber
- 4 universelle Leistungstreiber
- 3 Servo Ausgänge
- 3 Encoder
- 1 Infrarot Sender
- 1 serielle V.24 Schnittstelle
- 1 4fach DIP-Schalter
- 1 2x16 Zeichen LCD
- optional eine CAN-Schnittstelle

Die Ein-/Ausgänge sind weitestgehend kompatibel zu denen des MIT 6270-Boards. Dadurch ist eine vielzahl an Peripherie vorhanden. Die 4 Motortreiber dienen zur Steuerung der Drehrichtung und Drehzahl der Motoren. Sie können mit jeweils 1A Dauerlast betrieben werden. Die Leistungstreiber können universell benutzt werden, und sind jeweils bis maximal 1 Watt belastbar. Die Analogeingange haben eine Auflösung von 10-Bit. Eine Besonderheit des AkSen sind die 3 Servo Ausgange. Hierfür gibt es in der Bibliothek Funktionen, wie auch für die anderen Ein- und Ausgange, die einen einfachen Zugriff auf die Hardware erlauben.

[1] Complex Programmable Logic Device

2.1.1 Serielle Schnittstelle des Aktor- und Sensormodules

Der Prozessor verfügt über fünf On-Chip Peripherie Komponenten. Dies sind Parallele I/O Ports, Timer/Zähler, Interrupthandling, A/D Wandler und ein serielles UART Interface. Dieses serielle Interface kann in 4 verschiedenen Betriebsmodi benutzt werden. Diese sind:

- **Mode 0:** Shift Register Mode
  Die Daten werden synchron über RxD gesendet und emfpangen. Der für die synchrone Übertragung nötige Takt, wird über TxD ausgegeben. Die Baudrate ist fest auf $\frac{1}{12}$ der Oszillatorfrequenz eingestellt.
- **Mode 1:** 8-Bit UART, Variable Baud Rate
  Die Daten werden asynchron über RxD und TxD gesendet und empfangen. Zusätzlich werden noch Start- und Stop-Bit übertragen. Die Baudrate ist variabel.
- **Mode 2:** 9-Bit UART, Feste Baud Rate
  Die Daten werden asynchron über RxD und TxD gesendet und empfangen. Zusätzlich zum Start- und Stop-Bit, wird noch ein weiteres Datenbit übertragen, das zum Beispiel als Parity-Bit genutzt werden kann. Die Baud Rate ist fest auf $\frac{1}{32}$ oder $\frac{1}{64}$ der Oszilatorfrequenz eingestellt.
- **Mode 3:** 9-Bit UART, Variable Baud Rate
  Die gleiche Funktion wie in Mode 2, aber mit variabler Baud Rate.

Mode 2 und 3 sind spezielle Betriebsmodi, die hauptsächlich für die Multiprozessorkommunikation vorgesehen sind.

2.2 Bluetooth Modul

Bei dem Bluetooth Modul handelt es sich um das RS+I der Firma Stollmann[1]. Das Modul verfügt über ein serielles V.24 Interface. Das Interface arbeitet aber mit 5V TTL Pegel, und nicht mit den im V.24 Standard definierten +-12V. Das Interface unterstütz Baudraten von 2400 - 230400 bit/s. Die Betriebsspannung kann zwischen 3,3 Volt oder 5 Volt gewählt werden.

Das Modul unterstützt drei Betriebsmodi, um eine Verbindung aufzubauen. Dies sind der AT-Mode, Autoconnect Mode und Autoconnect DTR Mode.

2.2.1 AT-Mode

Im AT-Mode wird das Modul über einen AT Befehlssatz gesteuert, der sehr ähnlich zu dem von Analogmodems ist. Der AT Mode bietet den grössten Einsatzbereich. Mit ihm ist es möglich dynamisch Verbindungen zu anderen Bluetoothgeräten aufzubauen und dynamisch auf eingehende Verbindungsanfragen zu reagieren.

Ein Verbindungsaufbau läuft wie folgt ab:

1. in AT-Mode wechseln
   `<1 sec. Pause>+++<1 sec. Pause>`
2. Liste der verfügbaren Clients aktualisieren
   `AT**BDINQ`
   Das aktualisieren der Liste kann bis max. 20 Sekunden dauern, wenn 9 Bluetooth empfänger in Reichweite sind.
3. Liste der Clients einlesen
   `AT**BDLIST`
   Dies gibt eine Liste zurück, in der die verfügbaren Clients mit Name, Adresse und ID aufgelistet sind.
4. Verbindung zum Client aufbauen
   `ATD <Client ID>`
5. AT-Mode verlassen
   `ATO`

Der Nachteil des AT-Modes ist, das die Anwendungen daran angepasst werden muss. Will man dies vermeiden oder ist es zu aufwendig, sollte man einen der beiden Autoconnectmodi benutzen. Es gibt noch eine Reihe weiterer AT-Befehle, diese sind im „BlueRS+E BlueRS+ Bluetooth Serial Adapter User manual“[6] beschrieben.

2.2.2 Autoconnect Mode

Dieser Modus dient als „Kabelersatz“. Das heisst, das die Bluetoothverbindung für die Anwendung völlig transparent ist. In diesem Modus wird das Modul im vorraus programmiert (siehe Abschnitt 4.1). Sobald Daten gesendet werden sollen, wird die Verbindung zu dem vorher definierten Client aufgebaut. Der Autoconnect Mode hat einige Vorteile:

- Die Anwendung muss nicht geändert werden.
- Ist für Anwendungen geeignet, bei denen es nicht möglich ist, die Anwendung zu ändern, z.B. Sensoren.

2.2.3 Autoconnect DTR Mode

Der Autoconnect DTR Mode ist eine andere Variante des Autoconnect Modes. In diesem Modus wird die Verbindung aufgebaut, wenn die Handshakeleitung DTR aktiviert wird. Auch in diesem Modus ist keine Änderung der Anwendung nötig, wenn die DTR Handshakeleitung vorher auch benutzt wurden.

3 Realisierung

3.1 Hardware

Der Prozessor liefert auf den Ausgangspins des UARTS TTL Pegel. Um diesen TTL Pegel an den Standardpegel einer RS-232 Schnittstelle anzupassen, ist im Originaldesign ein Pegelwandler vom Typ MAX 232 vorgesehen (siehe Datenblatt [5]). Das Bluetooth-Modul ist auch auf TTL-Pegel ausgelegt. Die zusätzliche Pegelwandlung ist nicht mehr nötig. Der Pegelwandler vom Typ MAX 232 ist auf dem Board gesockelt. Wie man aus Tabelle 1 ersehen kann, sind an diesem Sockel alle Anschlüsse vorhanden, die man zum Anbschluss des Bluetooth-Modules benötigt. Das sind VCC +5V, GND, RDX und TXD, siehe Tabelle 2. Dadurch ergibt sich eine einfache Möglichkeit, das Bluetooth-Modul mit dem AkSen-Modul zu verbinden. Das Bluetoothmodul ist über ein Adapterkabel an diesen Sockel angeschlossen, was einen einfachen Wechsel zwischen einer drahtgebundenen Verbindung, mit dem MAX 232, und einer schnurlosen Verbindung mit dem Bluetoothmodul ermöglicht. Dieses Adapterkabel besteht aus einem einfachem IC-Sockel und einem Flachbandkabel.

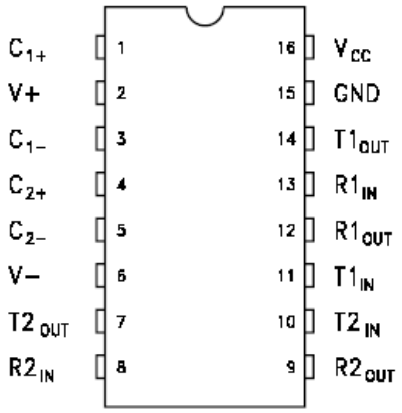

Abbildung 2: Pinout MAX 232

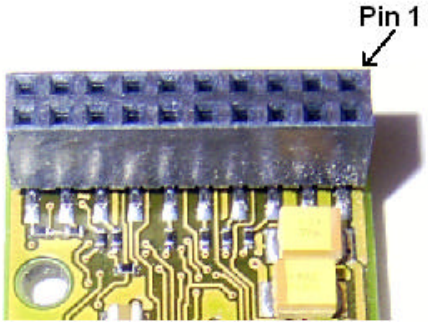

Abbildung 3: Pinout Bluetoothmodul

Die Stromaufnahme des RS+I Moduls beträgt maximal 46mA. Dies ist er Fall, wenn eine Bluetoothverbindung mit 115200Bit/s besteht und Daten gesendet werden.

PIN	SYMBOL	NAME and FUNKTION
1	C_1+	Positive Terminal for the first Charge Pump Capacitor
2	$V+$	Doubled Voltage Terminal
3	C_1-	Negative Terminal for the first Charge Pump Capacitor
4	C_2+	Positive Terminal for the second Charge Pump Capacitor
5	C_2-	Negative Terminal for the second Charge Pump Capacitor
6	$V-$	Inverted Voltage Terminal
7	$T2_{OUT}$	Second Transmitter Output Voltage
8	$R2_{IN}$	Second Receiver Input Voltage
9	$R2_{OUT}$	Second Receiver Output Voltage
10	$T2_{IN}$	Second Transmitter Input Voltage
11	$T1_{IN}$	First Transmitter Input Voltage
12	$R1_{OUT}$	First Receiver Output Voltage
13	$R1_{IN}$	First Receiver Input Voltage
14	$T1_{OUT}$	First Transmitter Output Voltage
15	GND	Ground
16	V_{CC}	Supply Voltage

Tabelle 1: Pinout Beschreibung MAX 232

PIN	Signal	BlueRS+I usage
1	GND	0V-Power
2	V_{CC}	+5V / +3.3V -Power
3	GND	GND
4	TXD	Transmit Data
5	GND	GND
6	RXD	Receive Data
7		reserved
8	RTS	RTS low active
9		reserved
10	CTS	CTS low active
11	RESET	RESET low active
12	DTR	DTR low active
13		reserved
14	DCD	DCD low active
15	RI	RI low active
16	DSR	DSR low active
17	UA	User Output 1
18	UE	User Input 1
19	UA2	User Output 2
20	UE2	User Input 2

Tabelle 2: Pinout RSI Bluetooth Modul

Der Pegelwandler MAX 232 hatte eine Stromaufname von maximal 10 mA. Die Batterrielaufzeit wird also nicht allzu sehr durch das Bluetoothmodul vermindert.

3.2 Berechnung der möglichen Baudraten

Die Kommunikation mit dem Bluetoothmodul, funktioniert in Betriebsmode 1. Je nachdem, welcher Betriebsmode benutzt wird, gibt es verschiedene Möglichkeiten, die erforderliche Baudrate zu erzeugen. Im Mode 1 erfolgte die Erzeugung der Baudrate entweder durch einen speziellen internen Baudgenerator(siehe Abbildung 4) oder über einen der internen Timer, der dann aber nicht mehr anderweitig benutzt werden kann.

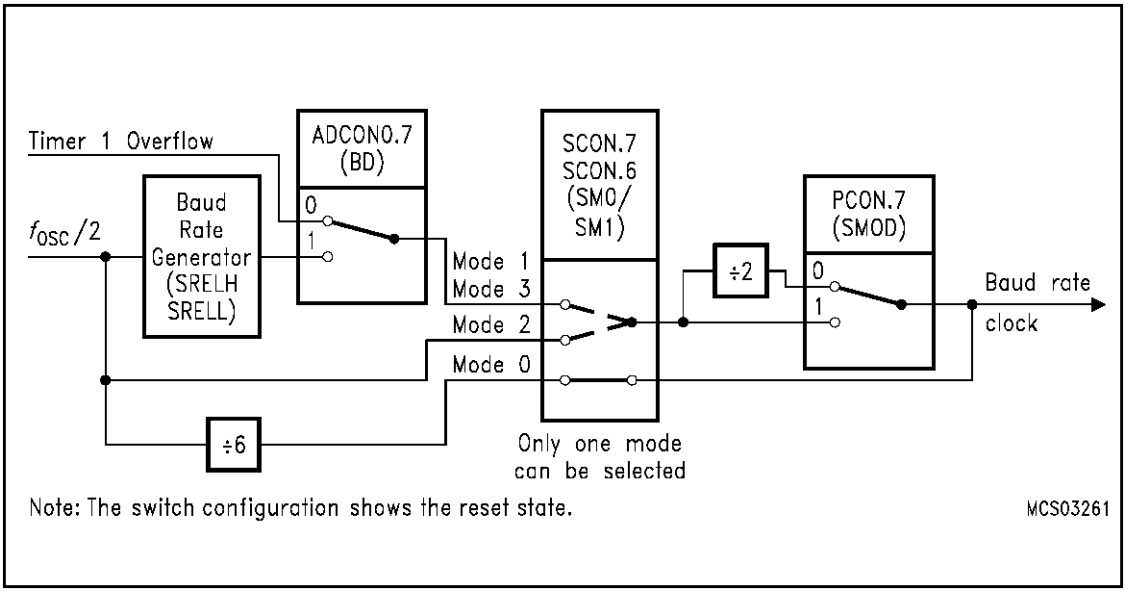

Abbildung 4: Diagramm des Baudrate Generators

Die möglichen Baudraten, die mit dem internen Baud Rate Generator erzeugt werden können, sind durch die folgende Formel berechenbar. Register SMOD gibt dabei an, ob die Baudrate noch einmal halbiert wird. Register SREL ist der Reload Wert des Baud Rate Generators.

$$Mode1,3Baudrate = \frac{2^{SMOD} \cdot OszillatorFrequenz}{64 \cdot (2^{10} - SREL)} \tag{1}$$

Für die Erzeugung der Baud Rate mit Hilfe des Timer 1, gilt folgende Formel.

$$Mode1,3Baudrate = \frac{2^{SMOD} \cdot OszillatorFrequenz}{32 \cdot 12 \cdot (256 - (TH1))} \tag{2}$$

Timer 1 steht nicht für die Erzeugung der Baudrate zur Verfügung, weil er schon von der AkSen Bibliothek genutzt wird. Somit kann die Baudrate nur durch den

Baudrate	**mögliche Baudrate**	**Abweichung**
1200	1201	0,08%
2400	2403	0,12%
4800	4807	0,14%
9600	9600	0,15%
19200	18750	-2,4%
38400	37500	-2,4%
57600	53571	-7,5%
115200	125000	7,84%

Tabelle 3: Häufige Baudraten mit Abweichung bei Verwendung des Baudgenerators und Register $SMOD = 1$

internen Baudgenerator erzeugt werden. Wie man aus Tabelle Tabelle 3 erkennen kann, sind nur Baudraten bis 9600 Baud möglich. Die Baudraten 19200 und 38400 weichen jeweils um nur $-2,4\%$ ab, und könnten noch funktionieren. Eine Testverbindung zu einer seriellen Schnittstelle am PC hat einwandfrei funktioniert, aber eine Verbindung mit dem Stollmann RSI Modul funktionierte nicht. Den erforderlichen Wert für das Register SREL kann man Tabelle 5 entnehmen. Bei höheren Baudraten reicht die Genauigkeit des Baudgenerators nicht mehr aus, sodass die erzeugten Baudraten teils erheblich von den Standard Raten abweichen. Da der Baudgenerator den Takt abhängig von der Oszillatorfrequenz des Prozessors erzeugt, ergibt sich dadurch die Möglichkeit, die Oszillatorfrequenz so zu wählen, das zum Beispiel auch eine Baudrate von 115200 möglich ist. Dies könnte aber zu Nebenwirkungen in der AkSen Bibliothek führen. Ausserdem entstehen dadurch weitere Probleme beim Empfang (siehe Abschnitt 4.2).

Baud Rate		f_{osc} (MHz)	SMOD	BD	Timer 1	
					Mode	Reload Value
Mode 1, 3 :	62.5 Kbaud	12.0	1	0	2	FF_H
	125 Kbaud	24.0	1	0	2	FF_H
	19.5 Kbaud	11.059	1	0	2	FD_H
	9.6 Kbaud	11.059	0	0	2	FD_H
	4.8 Kbaud	11.059	0	0	2	FA_H
	2.4 Kbaud	11.059	0	0	2	$F4_H$
	1.2 Kbaud	11.059	0	0	2	$E8_H$
	110 Baud	6.0	0	0	2	72_H
	110 Baud	12.0	0	0	1	$FEEB_H$
					Baud Rate Generator Reload value	
	375 Kbaud	12.0	1	1	$3FF_H$	
	562.5 Kbaud	18.0	1	1	$3FF_H$	
	750 Kbaud	24.0	1	1	$3FF_H$	
	9.6 Kbaud	12.0	1	1	$3D9_H$	
	9.6 Kbaud	18.0	1	1	$3C5_H$	
	9.6 Kbaud	24.0	1	1	$3B2_H$	
Mode 0 :	1 Mbaud	12.0	-	-	-	-
	1.5 Mbaud	18.0	-	-	-	-
	2 Mbaud	24.0	-	-	-	-
Mode 2 :	187.5 Kbaud	12.0	0	-	-	-
	375 Kbaud	12.0	1	-	-	-
	281 Kbaud	18.0	0	-	-	-
	562.5 Kbaud	18.0	1	-	-	-
	375 Kbaud	24.0	0	-	-	-
	750 Kbaud	24.0	1	-	-	-

Abbildung 5: Auszug aus dem Datenblatt des C515 mit den am häufigsten genutzten Baudraten und den dazu gehörenden SREL Register Werten

4 Software

Um die Funktion der Bluetoothverbindung zu testen, gibt es zwei kleine Testprogramme. Eins, das auf dem AkSen läuft, und als zweites ein kleines Programm das auf einem Palm mit Bluetoothschnittstelle läuft. Hier könnte man alternativ auch ein Programm für den PC schreiben, das dann auf das AkSen zugreifft.

4.1 Konfiguration Bluetoothmodul RSI

Für die Konfiguration des RSI als Kabelersatz, gibt es ein komfortables Konfigurationsprogramm, mit dem man alle Einstellungen im vorraus einstellen kann. Das Programm ist nur für MS Windows verfügbar. Da es aber nur ein GUI Frontend für das textbasierende Konsoleninterface des RSI ist, kann man die Konfiguration mit jedem seriellen Terminalprogramm durchführen. Eine genaue Beschreibung des Konsoleninterfaces ist im „BlueRS+E BlueRS+ Bluetooth Serial Adapter User manual“[6] zu finden. Die wichtigsten Einstellungen sind der „Autoconnect“ Mode, die Parameter der seriellen Schnittstelle(siehe Abbildung 6, oben rechts), wie Baudrate, Anzahl Stopbits und Parity, und die verschiedenen Bluetoothparameter (siehe Abbildung 6, unten links), wie Name und Sicherheitspin.

4.2 Testprogramm AkSen

Da erstmal keine Verbindungen vom AkSen hergestellt werden sollen, ist der Autoconnect Mode des RSI die einfachst Variante. So kann, unabhängig von der Software des AkSen, von extern eine Verbindung zum Aktor- und Sensormodul hergestellt werden. Dies hat den Vorteil das der Flashspeicher auch über die Bluetoothverbindung programmiert werden kann.

Auf dem AkSen läuft ein kleines Testprogramm, welches die Kommandos, die über die Bluetoothverbindung empfangen werden, ausführt. Der Emfpang der Daten funktioniert nur im Pollingbetrieb, und nicht per Interrupt. Der Grund hierfür ist die AkSen Bibliothek, die verhindert, das man eigene Interrupt Service Routienen benutzen kann. Da die AkSen Bibliothek nur in binärer Form zugänglich ist, lässt sich daran auch nichts ändern. Eine Möglichkeit dies zu ändern wäre es, eine eigene Bibliothek zu schreiben. Da dies den Rahmen dieser Arbeit übersteigen würde, bleibt nur die Möglichkeit des Pollings.

Das Problem das hierraus ensteht, ist folgendes: Der serielle Port muss im Programm sehr oft periodisch abgefragt werden. Es muss also sehr genau darauf geachtet werden, wieviel Zeit im restliche Programm verbraucht wird. Bei einer Baudrate von 9600 Baud und bei Ausnutzung der gesamten Übertragungsbandbreite,

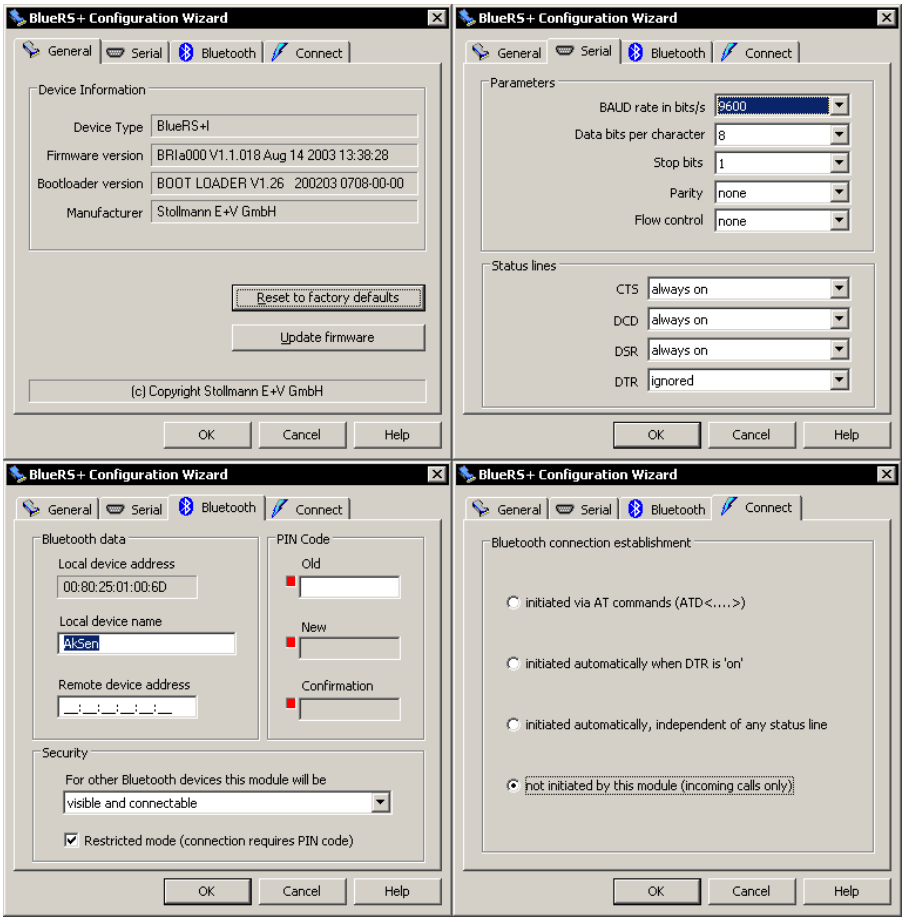

Abbildung 6: RSI+ Konfigurationstool

steht alle $833,3\mu sec$ ein neues Zeichen zur Verfügung (siehe Beispiel 3). Bei 115200 Baud alle $69,4\mu sec$ (siehe Beispiel 4). Die Wahrscheinlichkeit das ein Zeichen verloren geht nimmt also stetig zu. Insbesondere ist es auch nicht vorhersehbar, wieviel Zeit verbraucht wird, wenn ein Interrupt, zum Beispiel durch einen der Encoder, ausgelöst wird.

$$\frac{1sec}{\frac{9600\text{Baud}}{8\text{Bit}}} = 833,3\mu sec \tag{3}$$

$$\frac{1sec}{\frac{115200\text{Baud}}{8\text{Bit}}} = 69,4\mu sec \tag{4}$$

Deshalb bestehen die Kommandos nur aus einem Byte. Es wird dadurch zwar nicht garantiert, das kein Zeichen verloren geht, aber wenn ein Zeichen empfangen wird, ist zumindest sichergestellt, dass das Zeichen fehlerfrei übertragen wurde. Dies wird durch das Bluetoothprotokoll sichergestellt. Wenn ein Zeichen verloren geht, liegt es daran, das ein altes Zeichen im Empfangspuffer überschrieben wurde, weil der

Empfangspuffer zu selten abgefragt wurde. Um die Übertragung abzusichern, müsste man ein Protokoll definieren, das dann z.B. über eine Checksumme die Datenintegrität sicherstellt. Die verfügbaren Kommandos sind durch nachfolgende Definitionen beschrieben.

```
/* remote commands */
#define CMD_NOP              0
#define CMD_STOP             1
#define CMD_FORWARD          2
#define CMD_BACKWARD         3
#define CMD_ROTATE_LEFT      4
#define CMD_ROTATE_RIGHT     5
#define CMD_ACTION           6
```

Die Kommandos dienen dazu, ein einfaches Programm zu erstellen, mit dem sich das AkSen fernsteuern lässt. Zum kompilieren wird der SDCC[2][4] benötigt. SDCC ist ein Freeware C-Compiler, speziell für Intel MCS51 basierende Mikroprozessoren (8031, 8032, 8051, 8052, etc), Zilog Z80 MCUs, und Dallas DS80C390. Zu diesen speziellen Anpassungen gehören Präprozessoranweisungen und ein paar neue Keywords, wie zum Beispiel „interrupt“. Hiermit lassen sich auf einfache Art und Weise Interrupt Service Routienen in C programmieren. Dies hat aber wegen oben beschriebener Gründe für diese Arbeit keine Relevanz.

4.3 Testprogramm Palm

Das Testprogramm auf dem Palm(siehe Abbildung 7) dient zum steuern des Aktor- und Sensor Modules. Die Anforderungen an den Palm sind ein vorhandenes Bluetoothmodul und ein 5-way Navigationskeypad. Ein Gerät das diese Anforderungen erfüllt ist zum Beispiel der Palm Tungsten T. Das Progamm baut eine Bluetooth RfComm Verbindung zum AkSen auf. Die RfComm Verbindung wird auf dem Palm wie eine normale serielle Verbindung gehandhabt. Über das 5-Way Navigationskeypad lässt sich das AkSen steuern. Zum Erstellen des Programms werden die prc-tools[3] benötigt. Die prc-tools basieren auf dem Open Source Compiler GCC. Zusätzlich wird das SDK für Palm OS Version 5 benötigt, in dem die Headerfiles für die PalmOS interne API enthalten sind, das es bei Palm gibt.

[2]Small Device C Compiler

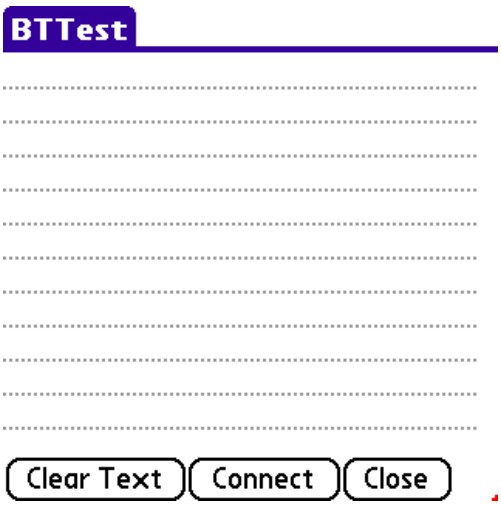

Abbildung 7: Screenshot AkSen Remote

Abbildung 8: Suche nach AkSen Modul

5 Zusammenfassung

Ziel dieser Studienarbeit war es, die Funktionen und Möglichkeiten des Aktor- und Sensormoduls aufzuzeigen. Und als Hauptziel die Steuerung des AkSen über eine Funkverbindung.

Der Anschluss des Bluetoothmodules an das Aktor- und Sensormodul bereitete keine größeren Probleme. Die Funkverbindung konnte ohne Einschränkungen die kabelgebundene Verbindung ersetzen. Der Programmspeicher des Aktor- und Sensormodules ließ sich direkt beschreiben. Und mit einem Testprobgramm ist es möglich, das AkSen zu steuern.

Es zeigten sich aber auch einige Einschränkungen. Insbesondere das Interrupthandling. So ist es im Moment leider nicht möglich, mit der mitgelieferten Bibliothek, das Interrrupthandlig selbst im Userprogramm zu handhaben. Dies hat zur Folge, das der serielle Empfang nur mittels Polling möglich ist, welches einige Nachteile mit sich bringt.

Auch hat sich gezeigt, dass das Aktor- und Sensormodul ein guter Ersatz für das MIT 6270 Board ist, weil es zum einen Hardwarekompatibel ist, und zum anderen auch über wesentlich mehr Anschlussmöglichkeiten verfügt. Ausserdem ist es auch wesentlich schneller, da die Programme nicht durch einen Bytecode Interpreter ausgeführt werden, und erlaubt es, komplexere Anwendungen zu entwickeln. Eine dieser Anwendungen wäre es vielleicht, mit der jetzt möglichen Funkübertragung, eine einfache Variante vom Roboterfussball zu realisieren, bei der sich die einzelnen Roboter, über den Spielverlauf austauschen und ihr Verhalten dementsprechend anpassen.

Literatur

[1] : *Stollmann Entwicklungs- und Vertriebs-GmbH.* – URL `http://www.stollmann.de`

[2] Infineon Technologies AG (Veranst.): *User Manual C515A.* 8 1997. – URL `http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=8028&parent_oid=13733`

[3] : *Palm OS programming with GCC.* Webseite. – URL `http://prc-tools.sourceforge.net/`

[4] : *Small Device C Compiler.* Webseite. – URL `http://sdcc.sf.net`

[5] STMicroelectronics (Veranst.): *Datasheet ST232CN.* 1 2003. – URL `http://us.st.com/stonline/books/pdf/docs/6420.pdf`

[6] Stollmann Entwicklungs- und Vertriebs-GmbH (Veranst.): *BlueRS+E BlueRS+ Bluetooth Serial Adapter User manual.* 1.04. 8 2003. – URL `ftp://ftp.stollmann.de/Bluetooth_Adapter/BlueRS+I/`

Abbildungsverzeichnis

Tabellenverzeichnis

A Listings

A.1 Listing AkSen Remote

Listing 1: remote.c

```
1  /* remote.c
2  *
3  * email: marco.neubauer@informatik.haw-hamburg.de
4  *
5  * $Id: remote.c,v 1.9 2003/12/18 10:59:40 icetea Exp $
6  */
7  #include <stdio.h>
8  #include <regc515c.h>
9  #include <stub.h>
10
11 #include "remote.h"
12
13
14 unsigned int strlen(char* string) {
15     unsigned int len = 0;
16     while (*string++ != 0)
17         len++;
18     return len;
19 }
20
21 /*
22 *  blocks until transmiter is ready
23 */
24 void send_byte(unsigned char c) {
25     while (!(SCON & 0x02)) {}
26     SCON &= 0xFD;
27     SBUF = c;
28 }
29
30 void send_string(char *string) {
31     for (;*string != 0;string++)
32         send_byte(*string);
33 }
34
35 int receive() {
36     unsigned char rec;
37     if (SCON & 0x01) {
38         rec = SBUF;
39         SCON &= 0xFE;
40         return rec;
41     }
42     return -1;
43 }
44
45 /*
46 * send AT command mode "escape sequence"
47 */
48 int command_mode() {
49     sleep(1000);
50     send_string("+++");
51     sleep(1000);
52     return 0;
53 }
54
55 /*
56 *  initialise serial port
57 */
58 void init_serial() {
59     ADCON0 |= 0x80; // enable Baud Rate Genarator
60     PCON |= 0x80;   // double Baud Rate
61     SRELH = 0x03;   // set reload value for Baud Rate Generator
```

```
62      SRELL = 0xD9;    // 0x03D9 = 9600 Baud
63      SCON = 0x70;     // set serial mode 1, set RI only if stop bit is present
64      SCON &= 0xFC;    // clear receive flag RI and TI
65      SBUF = ' '; // send one char, to be sure, that TI ist set
66  }
67
68  /*********************************************/
69  /* MAIN-Routine                              */
70  /*********************************************/
71  void AksenMain(void) {
72      int in;
73      int timeout = 0;
74      int command = CMD_STOP;
75
76      /* init serial port */
77      init_serial();
78
79      while (1) {
80
81          /* clear display */
82          /*lcd_cls();*/
83          lcd_setxy(0, 0);
84
85          /* receive command */
86          in = receive();
87          /*if (in != -1) {
88              send_byte(in);
89              lcd_putchar(in);
90      }*/
91
92          /* handle input */
93          switch (in) {
94          case CMD_FORWARD:
95          case CMD_BACKWARD:
96          case CMD_ROTATE_LEFT:
97          case CMD_ROTATE_RIGHT:
98          case CMD_ACTION:
99              command = in;
100             timeout = dip() * 4;
101             break;
102         default:
103             if (timeout-- <= 0) {   /* timeout */
104                 command = CMD_STOP;
105                 timeout = 0;
106             }
107             break;
108         }
109
110         /* display timeout value */
111         lcd_setxy(0, 0);
112         lcd_puts("timeout: ");
113         lcd_sint(timeout);
114         lcd_puts("     ");
115         lcd_setxy(1, 0);
116
117         /* handle command */
118         switch (command) {
119         case CMD_FORWARD:
120             motor_richtung(0, 0);
121             motor_richtung(1, 0);
122             motor_pwm(0, 5);
123             motor_pwm(1, 5);
124             lcd_puts("vorwaerts       ");
125             break;
126         case CMD_BACKWARD:
127             motor_richtung(0, 1);
128             motor_richtung(1, 1);
129             motor_pwm(0, 5);
130             motor_pwm(1, 5);
131             lcd_puts("rueckwaerts     ");
```

```
132             break;
133         case CMD_ROTATE_LEFT:
134             motor_richtung(0, 1);
135             motor_richtung(1, 0);
136             motor_pwm(0, 5);
137             motor_pwm(1, 5);
138             lcd_puts("links␣␣␣␣␣␣␣␣␣␣␣");
139             break;
140         case CMD_ROTATE_RIGHT:
141             motor_richtung(0, 0);
142             motor_richtung(1, 1);
143             motor_pwm(0, 5);
144             motor_pwm(1, 5);
145             lcd_puts("rechts␣␣␣␣␣␣␣␣␣␣");
146             break;
147         case CMD_ACTION:
148             motor_pwm(0, 0);
149             motor_pwm(1, 0);
150             lcd_puts("action␣␣␣␣␣␣␣␣␣␣");
151             break;
152         case CMD_STOP:
153             motor_pwm(0, 0);
154             motor_pwm(1, 0);
155             lcd_puts("stop␣␣␣␣␣␣␣␣␣␣␣␣");
156             break;
157         case CMD_NOP:
158             lcd_puts("nop␣␣␣␣␣␣␣␣␣␣␣␣␣");
159             break;
160         default:
161             break;
162         }
163         /* sleep(60); */
164     }
165 }
```

Listing 2: remote.h

```
1  #ifndef REMOTE_H
2  #define REMOTE_H 1
3
4  /* AT command return values */
5  #define RETVAL_OK           0
6  #define RETVAL_CONNECT      1
7  #define RETVAL_RING         2
8  #define RETVAL_NO_CARRIER   3
9  #define RETVAL_ERROR        4
10 #define RETVAL_NO_DAILTONE  5
11 #define RETVAL_BUSY         6
12 #define RETVAL_NO_ANSWER    7
13
14 /* remote commands */
15 #define CMD_NOP             0
16 #define CMD_STOP            1
17 #define CMD_FORWARD         2
18 #define CMD_BACKWARD        3
19 #define CMD_ROTATE_LEFT     4
20 #define CMD_ROTATE_RIGHT    5
21 #define CMD_ACTION          6
22 unsigned int strlen(char* string);
23 void send_byte(unsigned char c);
24 void send_string(char *string);
25 int receive();
26 int command_mode();
27 void init_serial();
28 void AksenMain(void);
29
30 #endif /* REMOTE_H */
```

Listing 3: Makefile

```
1  AKS_INC = "../../Header"
2
3  remote.ihx: remote.rel AkSen.rel
4          sdcc -V  --model-large --stack-loc 0x15 --stack-auto AkSen.rel remote.rel
5
6
7  remote.rel: remote.c remote.h
8          sdcc -c -I${AKS_INC} --model-large --stack-auto -V remote.c
9
10 AkSen.rel:
11         cp ../../Stub/AkSen.rel .
12         cp ../../Stub/AkSen.lst .
13
14 clean:
15         rm *.lst
16         rm *.rel
17         rm *.sym
18         rm *.asm
19         rm *.map
20         rm *.ihx
21         rm *.lnk
22         rm *.rst
```