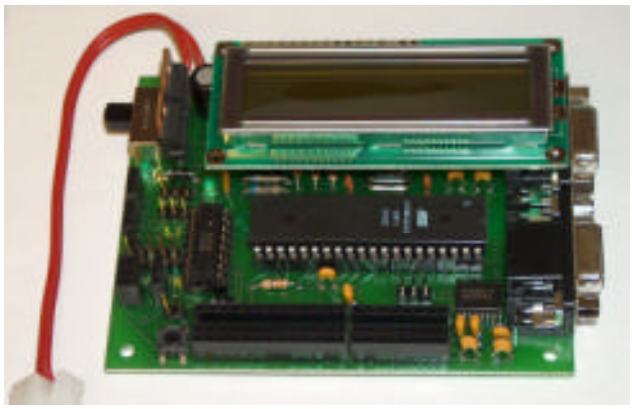
Softwarebibliothek für
KR-mega32-16

Thomas Krause, Krause Robotik
thomas.krause@krause-robotik.de



Willkommen zur Dokumentation zur Softwarebibliothek KR-mega32-16v20.

Diese Dokumentation beschäftigt sich nur mit der Bibliothek und ihren Funktionen. Grundlegende Kenntnisse zur Programmiersprache C werden vorrausgesetzt. Dazu gibt es zahlreiche Bücher und Dokumentationen im WWW.

Die Bibliothek ist bereits fertig kompiliert und muss nur in die eigenen Projekte integriert werden. Das KR-mega32-16 Board ist ein Mikrocontrollerboard, das unter Berücksichtigung der Besonderheiten auf kleinen mobilen Robotern entwickelt wurde. Es wird in dieser Dokumentation daher zur Vereinfachung als Roboboard bezeichnet.

Einbinden in eigene Projekte

Um dem Compiler alle Funktionen bekannt zu machen muß nur die Headerdatei „*KR-mega32-16v20.h*“ mit dem Befehl :

#include „KR-mega32-16v20.h“

eingebunden werden.
Im makefile muss die gleichnamige Bibliothekdatei “*KR-mega32-16v20.o*” eingebunden werden. Im Beispiel „rprintf“ ist das bereits als Standard umgesetzt. Es ist sinnvoll das dort verwendete makefile einmal anzupassen und dann für alle weiteren Projekte zu kopieren und nur die Angabe der Zieldatei anzupassen.

Funktionsumfang

Die Bibliothek umfasst alle Funktionen für den Zugriff auf die Peripherie des Boards. Die Funktionen sind in folgende Bereiche unterteilt:

1. Portzugriffe
2. I2C Bus
3. Ausgabefunktionen
4. UART
5. LCD
6. Servoroutinen
7. MotorPWM
8. Encoder
9. Zubehör

1. Port zugriffe

Das Roboboard verfügt über 21 Ports auf die direkt zugegriffen werden kann. Die Ports befinden sich alle auf der unteren Buchsenleistenanordnung. Die Ports sind rechts nach links von 1 bis 21 durchnummeriert.
Alle Ports können als digitale Ein - oder Ausgänge genutzt werden. Die Ports 1 bis 8 können außerdem auch als analoge Eingänge verwendet werden.

Um den vollen Funktionsumfang der Ports zu aktivieren sind folgende Befehle zur Initiierung notwendig:

```
i2cInit();          // für den Zugriff auf die Ports 17 – 21
a2dInit();          // für den Zugriff auf die Ports 1 – 8
                    // als analoge Eingänge
```

Es sind keine Parameter notwendig.

digitale Ausgänge:

Für die Nutzung als Ausgang gibt es zwei Befehle:

setDigitalOut(int port);

Dieser Befehl setzt den mit Port (kann Werte von 1 – 21 annehmen) genannten Port auf HIGH (+ 5V).

clearDigitalOut(int port);

Dieser Befehl setzt den mit Port (kann Werte von 1 – 21 annehmen) genannten Port auf LOW (+ 0V; GND)

Für den Zugriff als digitaler Eingang steht folgender Befehl zur Verfügung:

int digital(int port);

Der Befehl liefert einen Integerwert zurück. Liegen am genannten Port 1....16 + 5V an, gibt er eine 1 zurück. Liegen am Port 17...21 0 V an gibt er eine 1 zurück.
0V an Port 1...16 bzw. +5 V an Port 17...21 gibt er eine 0 zurück.

analoge Eingänge:

Hier gibt es nur einen Befehl:

int analog(int port);

Der Befehl liefert einen Integerwert (0 – 255) zurück der proportional der Spannung am mit port (kann Werte zwischen 1 und 8 annehmen) benannten Port ist.
Liegt am Port Nummer port 0V an, gibt die Funktion 0 zurück. Liegen 5 V an, gibt sie 255 zurück. Dazwischen werden linear Werte zwischen 0 und 255 ausgegeben.

2. I²C-BUS

Das Roboboard verfügt über einen I²C-Bus über den sich weitere Sensoren und Aktoren anschließen und ansprechen lassen.

Für den I²C-Bus stehen folgende Befehle zur Verfügung:

void i2cInit(void);

Initialisiert den I²C-Bus.
Achtung: Wird das LCD-Display initialisiert wird auch der I²C-Bus automatisch mit aktiviert.

void i2cSetBitrate(u16 bitrateKHz);

Setzt die Übertragungsgeschwindigkeit am Bus. Der Wert gibt kHz an. Die maximale Übertragungsrate beträgt 500 kHz. Achten Sie darauf, was sie für Geräte angeschlossen haben, und was deren maximale Übertragungsrate ist. Der Bus darf nur so schnell sein, wie der langsamste Busteilnehmer. Auch bei langen Leitungen kann es sein, daß die Übertragungsgeschwindigkeit runtergesetzt werden muß.

*void i2cMasterSend(u08 deviceAddr, u08 length, u08 *data);*

Sendet Daten, die im Feld data liegen an das Gerät mit der Adresse deviceAddr. Es werden so viele Bytes gesendet wie unter length angegeben sind.
Ein typisches Beispiel könnte so aussehen:

...
char adresse = 62;
char data[3];
char datalaenge = 0h03;

data[0] = 3;
data[1] = 'd'
data[2] = 32;

i2cMasterSend(adresse, datalaenge, data);
…

void i2cMasterReceive(u08 deviceAddr, u08 length, u08 data);*

Empfängt so viele Bytes wie mit length angeben vom Gerät mit der Adresse deviceAddr. Die Daten werden im Feld data abgelegt. Es ist darauf zu achten, daß das Feld data bei der Definition groß genug gewählt wird.

3. Ausgabefunktionen

Als Ausgabefunktion wird der Befehl:

void rprintf(String);

genutzt.
Die Ausgabe kann wahlweise auf dem LC-Display oder über die serielle Schnittstelle erfolgen. Wie die Ausgabe auf die verschiedenen Ausgaben umgeleitet wird, wird in den nächsten beiden Abschnitten behandelt.

Sie finden im Beispiel „rprintftest.c" einen guten Überblick über den Umgang mit rprintf. Die Funktionalität von rprintf ist der von printf aus der Standard-C Definition angelehnt. Die Entwicklung befindet sich noch vor der Version 1.0. Das heißt, es kann noch vereinzelt kleine Fehler und fehlende Funktionalitäten geben. Dennoch sind bereits die meisten Funktionalitäten zu finden.

4. UART

Die UART-Schnittstelle ist in dieser Bibliothek auf die asynchrone Datenübertragung an den PC eingestellt. Es ist aber auch problemlos die Kopplung mit einem weiteren Roboboard oder anderen Boards möglich..
Die Grundfunktionalitäten werden über folgende Befehle erreicht:

void uartInit(void);

Initialisiert die serielle Schnittstelle.

void uartSetBaudRate(u32 baudrate);

Setzt die Baudrate, mit der gearbeitet werden soll.
Standard ist 9600, wenn dieser Befehl nicht aufgerufen wird.

void uartSendByte(u08 data);

Sendet ein Byte über die serielle Schnittstelle.

int uartGetByte(&data);

Empfängt ein Byte von der seriellen Schnittstelle und speichert es in der Variable „data“.

Mit dem Befehl:

rprintfInit(uartSendByte);

kann die Ausgabefunktion der rprintf-Routinen auf die serielle Schnittstelle umgeleitet werden.

5. LCD

Zum Roboboard wird standardmäßig ein 2x16 Zeichen Display mitgeliefert. Es können aber auch andere Formate angeschlossen werden.

void lcdInit(void);

Initialisiert das Display. Dabei wird auch der I²C-Bus initialisiert, da das Display über ein I²C-Baustein an das Board angeschlossen ist.

void lcdHome(void);

Setzt den Cursor auf die Ausgangsposition (oben links).

void lcdClear(void);

Löscht das Display.

void lcdGotoXY(u08 row, u08 col);

Setzt den Cursor an die gegebene Position.

void lcdDataWrite(u08 data);

Schickt ein Zeichen an das Display.

Mit dem folgenden Befehl kann die rprintf-Ausgabe auf das LCD umgelenkt werden:

rprintfInit(lcdDataWrite);

6. Servoroutinen

An das Roboboard können zwei Servos angeschlossen werden.

void servoInit(void);

Initialisiert die Servoroutinen.

void servoOff(void);

Gibt die Routinen wieder frei.

void servoSetPosition(u08 channel, u08 position);

Setzt die Position für den Servo.
channel ist die Nummer des Servos (0 oder 1).
position ist die Position des Servos und kann zwischen 0 und 255 liegen.

7. MotorPWM

Das Roboboard kann zwei Motoren (bis 600 mA) direkt ansteuern. Die Motoren können von 0 Volt bis zur vollen Batteriespannung (vorwärts und rückwärts) geregelt werden.

void motorInit(void);

Initialisiert die PWMsteuerung für die Motoren. Dafür wird der Timer0 verwendet.

void motor(int port, int power);

Setzt die Geschwindigkeit, mit der der Motor angesteuert wird.
port kann 1 0der 2 sein und gibt an, welcher Motor gesetzt wird.
power kann zwischen –100 und 100 liegen. Positive Werte steuern den Motor vorwärts, negative Werte rückwärts an.

void motorStop(int port);

Stoppt den Motor. port ist 1 oder 2, je nachdem welcher Motor abgeschaltet werden soll.

9. Encoder

An das Roboboard können bis zu drei Encoder direkt angeschlossen und ausgewertet werden. Die meisten Radencoder haben zwei Kanäle. Damit ist es möglich die Richtung der Drehbewegung zu erkennen. Kanal A wird an einen Interrupteingang angeschlossen und Kanal B an den Digitalen Eingangsport 14 bis 16.

Portbelegung Radencoder 0:
Kanal A an Interrupteingang 0
Kanal B an Digitalport 16

Portbelegung Radencoder 1:
Kanal A an Interrupteingang 1
Kanal B an Digitalport 15

Portbelegung Radencoder 2:
Kanal A an Interrupteingang 2
Kanal B an Digitalport 14

Nutzung der Encoderfunktionen:

void encoderInit(void);

Initialisierung der Encoder:

u32 encoderGetPosition(int port)

Gibt die Position des jeweiligen Encoders zurück. „port“ gibt an, welcher Encoder ausgelesen wird.

8. Zubehör

Hier sollen einige Zubehörfunktionen und Definitionen genannt werden.
Der Umfang ist nicht vollständig und wird noch erweitert. Wir bitten, uns bei Fragen und Anregungen, was vermisst wird, zu benachrichtigen, damit diese Rubrik sinnvoll ergänzt werden kann.

Wartefunktion:

void timerPause(int msec);

Diese Funktion wartet die in msec angegebenen Millisekunden.

Anhang:

Pinbelegung: