avr_libmake: Konzept und Tool für das Handling von Assembler-Bibliotheken

V0.22 (c) 2006-2010 Joerg Wolfram

1 Allgemeines

Das Programm unterliegt der GPL (GNU General Public Licence) Version 3 oder höher, jede Nutzung der Software/Informationen nonkonform zur LGPL oder ausserhalb des Geltungsbereiches der GPL ist untersagt! Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, dass es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, auch ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK.
In der Version 0.20 habe ich einige Änderungen vorgenommen. Das betrifft zuallererst die Ergänzung der Bibliotheksdateien um die Dateiendung ".asm", damit Editoren mit Syntax-Highlighting den richtigen Dateityp erkennen können. Die Dateien befinden sich jetzt Bibliotheksweise in Verzeichnissen, was einerseits übersichtlicher ist und andererseits Bibliotheks-Updates einfacher macht. Zuguterletzt ist noch das Register **ctrl** dazugekommen, um Funktionen einfacher steuern zu können.

2 Anforderungen an ein Bibliothekskonzept

Um den Organisationsaufwand beim Einsatz der Bibliotheken zu minimieren, die Codegröße zu minimieren und eine weitestgehend offene Schnittstelle zu schaffen, müssen folgende Mindestanforderungen erfüllt sein:

- einfacher Einsatz der Bibliotheken, Minimierung des Verwaltungsaufwandes (Definitionen etc.)
- unabhängige Bibliotheken, die sich nicht (direkt) gegenseitig aufrufen dürfen
- Möglichkeit der Bereitstellung globaler Funktionen, die auch von anderen Bibliotheken genutzt werden können
- automatisches Weglassen nicht benötigter Bibliotheken und Funktionen
- automatische Ermittlung der Sprungdistanzen zur Optimierung der Funktionsaufrufe
- bei Durchführung der letzten beiden Punkte sollen Quelltext und Bibliotheken nicht verändert werden

3 Grundlegendes Konzept

Nach vielen Versuchen hat sich das folgende Konzept als brauchbar, wenn auch vielleicht nicht optimal erwiesen. Ein Nachteil kann eventuell (wenn auch für mich nicht) in der engen Anbindung an Linux und avra als Assembler bestehen, sofern man auf andere Betriebssysteme/Tools angewiesen ist. Die Bibliotheken selbst sollten sich aber auch ohne diese Voraussetzungen nutzen lassen.
Das Prinzip ist letztendlich sehr simpel. Zu Beginn des Hauptquelltextes wird eine Datei mit Makros eingebunden. Jedes Makro enthält den Funktionsaufruf als absoluten Sprung/Aufruf und setzt (definiert) einen Freigabewert. Bibliotheksaufrufe werden generell als Makroaufrufe getätigt. Am Ende des Hautquelltextes wird dann die eigentliche Bibliothek eingebunden. In der Bibliothek selbst werden dann abhängig von den bereits gesetzten Freigabewerten noch weitere Freigabewerte gesetzt (interne Abhängigkeiten). Durch bedingte Assemblierung der einzelnen Funktionen abhängig davon, ob der Freigabewert definiert ist, werden nur die wirklich benötigten Bibliotheksfunktionen assembliert.
Nach einem ersten Assemblervorgang wird die erstellte Listdatei auf Sprung- und Aufrufdistanzen bei den Bibliotheksfunktionen überprüft. Liegen diese im Bereich von relativen Sprüngen/Aufrufen, werden in der Makrodefinitionsdatei die Funktionsaufrufe entsprechend abgeändert. Zum Abschluss erfolgt noch ein zweiter Assemblervorgang.

4 Realisierung

4.1 Nomenklatur

Das realisierte Konzept bringt einige Restriktionen bei der Bezeichnung von Bibliotheken und Bibliotheksfunktionen mit:

- Die Bibliotheksdateien heissen alle **library.asm**, die Makrodateien **macros.asm**, und befinden sich in einem Ordner mit Namen der Bibliothek.
- Bibliotheksfunktionen beginnen mit dem Bibliotheksname (incl. "lib") gefolgt von Unterstrich und dem Funktionsnamen.
- Die Makros zum Aufruf der Bibliotheksfunktionen heissen genau so wie die Funktionen

Für eine PWM-Bibliothek würde sich zum Beispiel folgendes ergeben:

- Die Bibliotheksdatei heisst **libpwm1/library.asm**, die Makrodatei **libpwm1/macros.asm**.
- Eine Ausgabefunktion heisst dann **libpwm1_out**
- In die Hauptquelltextdatei wird am Anfang **.include "macros.inc"** und am Ende **.include "libs.inc"** geschrieben
- Um die Funktion zu nutzen, muss nur das Makro **libpwm1_out** aufgerufen werden.

4.2 Automatisierung

Will oder kann man das Skript **avr_libmake.pl** zur Automatisierung nicht nutzen, ist es auch möglich, am Anfang der Haupt-Quelltextdatei die einzelnen Makrodateien und am Ende die entsprechenden Bibliotheksdateien einzubinden. Einfacher gehts es jedoch mit dem Perl-Skript. Bevor es eingesetzt werden kann, muss es aber eventuell noch angepasst werden. Dazu gehören der Library-Pfad und der Assembleraufruf. Dann wird es am besten nach **/usr/local/lib** kopiert und das wars auch schon mit der Installation. Gestartet wird es am besten im Projektverzeichnis mit:
avr_libmake mainsource.asm
Wenn alles geklappt hat kann die .hex-Datei in den AVR geflasht werden, wenn nicht, fehlt vielleicht eine Bibliothek oder es ist ein Fehler im Quelltext. Der Quelltext der auf diesen Seiten herunterladbaren Bibliotheken ist zumindest syntaktisch überprüft, sollte also keine Assemblierfehler produzieren. Wenn doch, fehlen eventuell Registerdefinitionen.

4.3 Lokaler Speicher und Registernutzung

Manche Bibliotheken nutzen einen kleinen SRAM-Bereich für z.B. temporäre Daten oder einen Arithmetikstack. Zusätzlich müssen eventuell auch noch Definitionen für Adressen etc. festgelegt werden. Dazu dient die Datei **definitions.asm** im Bibliotheksverzeichnis. Entweder man kopiert den Inhalt in die Quelldatei oder man bindet sie über ".include libxxx/definitions.asm" ein.
Die Bibliotheken nutzen meist die folgenden Register, näheres steht in den jeweiligen Bibliotheksdokumentationen.

Register	Funktion
r0,r1	werden für Multiplikationen und temporär genutzt
byte0	beliebiges Register, welches mit 0x00 vorbelegt ist (z.B. r2)
byte1	beliebiges Register, welches mit 0x01 vorbelegt ist (z.B. r3)
ereg	beliebiges Register aus dem oberen Registerbereich, liefert Fehlercode (z.B. r25)
ctrl	beliebiges Register aus dem oberen Registerbereich, Format- und Steuerungsfunktionen (z.B. r24)
tempreg1... tempreg4	beliebige Register aus dem oberen Registerbereich
tempreg5... tempreg8	beliebige Register aus dem unteren Registerbereich
XL,XH	Ein-/Ausgabe von Daten, temporär genutzt
YL,YH	Pointer, Adressen im SRAM, temporäre Nutzung
ZL,ZH	Pointer, Adressen im SRAM, EEPROM oder Flash