CT Praktikum: Modulare Programmierung – Linker

1 Einleitung

In diesem Praktikum lernen Fehler zu beheben bei der Verwendung von fremden Libraries und wie Sie den Debugger dazu bringen, auch in die Library Funktionen hinein zu springen. Zusätzlich lernen Sie die Output Daten des Linkers zu lesen.

Das Projekt besteht aus einer Library mit dazu passenden Header Files.

Sie müssen der Entwicklungsumgebung angeben, wo die Header Files liegen, fehlende Include Direktiven im main.c angeben, und schliesslich angeben, wo der Linker die Library findet.

Danach geht es darum, mit dem erfolgreich gebildeten Programm zu debuggen. Dabei lernen Sie, wie Sie beim Builden zwischen Libraries mit und ohne Debug Information wechseln, und wie Sie dem Debugger sagen, wo er die Sourcen für Source-Line-Debugging findet.

2 Lernziele

- Sie können Compiler und Linker Fehlermeldungen, welche im Zusammenhang mit modularer Programmierung entstehen können, interpretieren und korrigieren
- Sie können Libraries inklusive notwendiger Header Files einbinden
- Sie können ELF Symbol Sections ausgeben und interpretieren
- Sie können Linker Map Files interpretieren

3 Aufgaben 1

3.1 Aufgabe 1.1: Compiler und Linker Fehlermeldungen interpretieren und korrigieren

In einem ersten Schritt soll das unfertige Projekt zum Laufen gebracht werden. Dazu müssen die Fehlermeldungen des Präprozessors, des Compilers und des Linkers interpretiert und korrigiert werden.

Siehe dazu die Task-Liste in der Datei ***main.c***.

Modulare Programmierung bedeutet für dieses Projekt, dass neben dem Code im ***app*** Ordner zusätzliche Library Header Files im ***.\inc*** Ordner liegen. Dieser Ordner muss an geeigneter Stelle in den Projekt Properties (C/C++ Tab) angegeben werden, damit diese Header Dateien vom Präprozessor/Compiler auch gefunden werden.

Analog müssen die einzelnen Libraries im Linker Tab der Projekt Properties unter *Misc Controls* dem ***lib*** Ordner angegeben werden. Geben sie dazu die benötigte Library in diesem Feld ein (z.B. ***lib\read_write.lib***)

Wenn Sie alle Fehler erfolgreich korrigiert haben, können sie das Programm auf das CT-Board laden.

Die Funktion des Programms ist simpel: Wenn Sie auf T0 drücken, wird beim ersten Mal drücken von dunkel auf ein fixes Muster gewechselt, mit jedem weiteren T0 Drücken, werden die LEDs invertiert.

3.2 Aufgabe 1.2 Debugging

a) Versuchen Sie das Programm im Debugger in Einzelschritten auszuführen (**Step/F11**). Was beobachten Sie bei der Funktion ***read8(BUTTONS)***? (Infos zu Debugging finden Sie auf CT Board Wiki (https://ennis.zhaw.ch) unter «Compile and Debugging»)

Was beobachten Sie?

Man kann nicht in die Funktion springen!

b) Ersetzen Sie im Linker Tab die Referenz auf ***lib\read_write.lib*** mit ***lib_debug\read_write.lib***. Was beobachten Sie wenn Sie nun kompilieren, linken und debuggen?

c) Schliesslich ersetzen sie die Referenz durch ***lib_debug_with_src\read_write.lib***. Beim Debugging mit Sourcen müssen Sie dem Debugger zusätzlich noch angeben, wo sich die Source-Files genau befinden. Im Library File selbst befinden sich nämlich keine Source-Files, lediglich die Symbole. Geben sie hierfür im *Command Window* des Debuggers folgenden Befehl ein:

set src = C:\<Ihr Pfad>\project\lib_debug_with_src

Der Debugger weiss jetzt, wo sich die Source-Files befinden. Achten Sie darauf, dass es im Pfad kein Leerzeichen hat.

Was beobachten Sie wenn Sie nun kompilieren, linken und debuggen?

Was ist der Grund für das veränderte Verhalten?

lib_debug => Man kann nun das Disassembly anschauen, aber nicht die Sourcen.

lib_debug_with_src => Man kann nun auch die Source anschauen und steppen.

3.3 Aufgabe 1.3: Symbole extrahieren

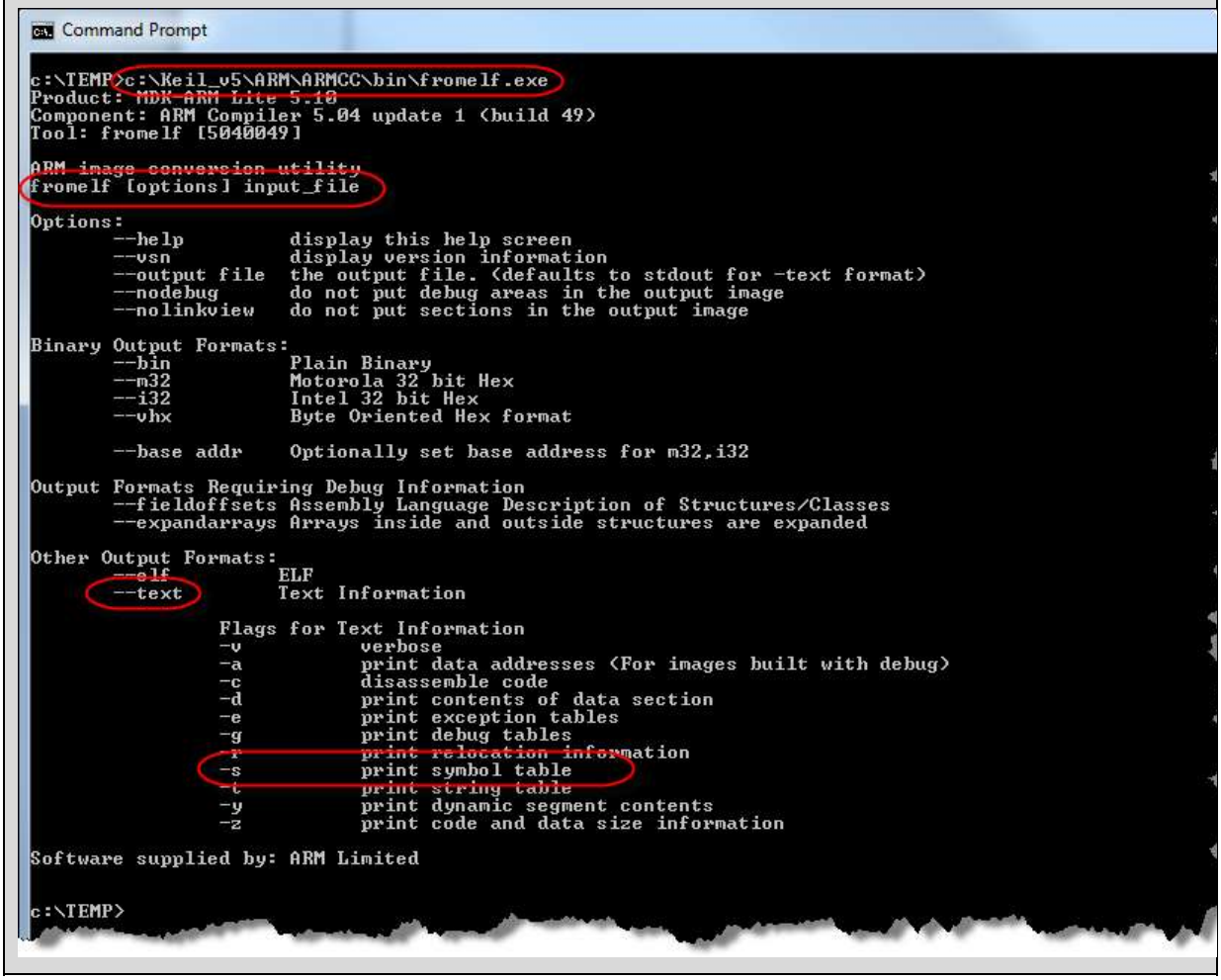
Das Tool ***fromelf.exe*** kann den Inhalt der binären ELF Dateien in lesbarer Form ausgeben. Die Objekt Dateien (file.o), die Libraries (file.lib) und die Programme (file.axf) sind alle in ELF File Format gegeben.

Führen Sie ***fromelf.exe*** in einem Command Prompt aus.

Ein Command Prompt öffnen Sie indem sie cmd.exe ausführen.

Das Tool ***fromelf.exe*** wird über diesen Pfad ausgeführt (Pfad kann abweichen je nachdem wo KEIL installiert wurde): ***C:\Keil_v5\ARM\ARMCC\bin\fromelf.exe***.

z.B.



Toggle.o

```
  #  Symbol Name                 Value      Bind  Sec  Type  Vis  Size
=============================================================================

  1  toggle.c                    0x00000000   Lc  Abs  File  De
  2  [Anonymous Symbol]          0x00000000   Lc    3  Sect  De
  3  $t.0                        0x00000000   Lc    3   --   De
  4  __arm_cp.0_0                0x00000014   Lc    3   --   De   0x4
  5  __arm_cp.0_1                0x00000018   Lc    3   --   De   0x4
  6  $d.1                        0x00000014   Lc    3   --   De
  7  value                       0x00000000   Lc    7  Data  De   0x1
  8  [Anonymous Symbol]          0x00000000   Lc    7  Sect  De
  9  [Anonymous Symbol]          0x00000000   Lc    8  Sect  De
 10  [Anonymous Symbol]          0x00000000   Lc   11  Sect  De
 11  [Anonymous Symbol]          0x00000000   Lc   15  Sect  De
 12  [Anonymous Symbol]          0x00000000   Lc   17  Sect  De
 13  toggle                      0x00000001   Gb    3  Code  Hi   0x14
 14  write8                      0x00000000   Gb  Ref   --   De
```

main.o

```
  #  Symbol Name                 Value      Bind  Sec  Type  Vis  Size
=============================================================================

  1  main.c                      0x00000000   Lc  Abs  File  De
  2  [Anonymous Symbol]          0x00000000   Lc    3  Sect  De
  3  $t.0                        0x00000000   Lc    3   --   De
  4  __arm_cp.0_0                0x00000038   Lc    3   --   De   0x4
  5  __arm_cp.0_1                0x0000003c   Lc    3   --   De   0x4
  6  $d.1                        0x00000038   Lc    3   --   De
  7  last                        0x00000000   Lc    7  Data  De   0x1
  8  [Anonymous Symbol]          0x00000000   Lc    7  Sect  De
  9  [Anonymous Symbol]          0x00000000   Lc    9  Sect  De
 10  [Anonymous Symbol]          0x00000000   Lc   12  Sect  De
 11  [Anonymous Symbol]          0x00000000   Lc   16  Sect  De
 12  [Anonymous Symbol]          0x00000000   Lc   18  Sect  De
 13  main                        0x00000001   Gb    3  Code  Hi   0x38
 14  read8                       0x00000000   Gb  Ref   --   De
 15  toggle                      0x00000000   Gb  Ref   --   De
 16  __ARM_use_no_argv           0x00000000   Gb    8  Data  De   0x4
```

read_write.lib

```
  #  Symbol Name                 Value      Bind  Sec  Type  Vis  Size
=============================================================================

  1  write.c                     0x00000000   Lc  Abs  File  De
  2  $t                          0x00000000   Lc    1   --   De
  3  .text                       0x00000000   Lc    1  Sect  De
  4  BuildAttributes$$THM_ISAv3M$S$PE$A:L22$X:L11$S22$IEEE1$IW$USESV6$~STKCKD$USESV7$~SHL$OSPACE$EBA8$REQ8$PRES8$EABIv2
                              0x00000000   Lc  Abs   --   De
  5  __ARM_grp_.debug_frame$5    0x00000000   Lc    4  Data  De
  6  write8                      0x00000001   Gb    1  Code  Hi   0x4
  7  Lib$$Request$$armlib        0x00000000   Wk  Ref  Code  Hi
```

```
  #  Symbol Name                 Value      Bind  Sec  Type  Vis  Size
=============================================================================

  1  read.c                      0x00000000   Lc  Abs  File  De
  2  $t                          0x00000000   Lc    1   --   De
  3  .text                       0x00000000   Lc    1  Sect  De
  4  BuildAttributes$$THM_ISAv3M$S$PE$A:L22$X:L11$S22$IEEE1$IW$USESV6$~STKCKD$USESV7$~SHL$OSPACE$EBA8$REQ8$PRES8$EABIv2
                              0x00000000   Lc  Abs   --   De
  5  __ARM_grp_.debug_frame$5    0x00000000   Lc    4  Data  De
  6  read8                       0x00000001   Gb    1  Code  Hi   0x4
  7  Lib$$Request$$armlib        0x00000000   Wk  Ref  Code  Hi
```

Generieren Sie für ***Objects\toggle.o***, ***Objects\main.o*** und ***lib\read_write.lib*** die Symbol Tabelle. Fokusieren Sie auf Code und Data Einträge und ignorieren Sie Debug Einträge

Welches sind lokale Symbole?

Welches sind exportierte Symbole?

Welches sind importierte Symbole (referenzierte Symbole)?

Hinweis: Gesucht sind nicht die Namen der einzelnen Symbole, sondern mit welcher Bezeichnung diese in der generierten Tabelle hinten markiert werden.

Lokal	**Importiert**	**Exportiert**
Binding: lc	Binding: Gb Visibility: De Abs: Ref	Binding: Gb Visibility: Hi Abs: Zahl

Vergleichen Sie die obige Antwort mit dem entsprechenden Header File.

Was ist mit den als exportierten Symbolen gemeldeten Einträgen, die nicht im Header File stehen?

Die Kommen von <stdint.h>

Wenn Sie eine Library haben und keine dazu passende Header Files, können Sie dann mit dem ***fromelf.exe*** Tool alle nötigen Informationen aus der Library extrahieren um selber ein Header File zu schreiben? Fehlt etwas?

Nein, die Parameter und Rückgabewerte können nicht ausgelesen werden.

3.4 Aufgabe 1.4: Linker Map Interpretieren

Beim Linken wird ein Map File kreiert. Prüfen Sie, welche der Informationen unter "Project" → "Options for Target ..." → Tab "Listing" im Map File vorkommen.

- Memory Map
- Callgraph
- Symbols
- Cross Reference
- Size Info
- Totals Info
- Unused Sections Info
- Veneers Info

Erklären Sie anhand des Linker Map Files, wie das Memory Map aussieht.

Wo sind welche Konstanten, welche Funktionen und welche Daten abgelegt?

Exec Addr	Load Addr	Size	Type	Attr	Idx	E Section Name	Object
0x08001338	0x08001338	0x00000040	Code	RO	2	.text.main	main.o
0x0800138c	0x0800138c	0x0000001c	Code	RO	12	.text.toggle	toggle.o
0x20000000	0x080015a8	0x00000001	Data	RW	14	.data.value	toggle.o
0x20000008	-	0x00000001	Zero	RW	4	.bss.last	main.o

3.5 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Aufgabe	**Bewertungskriterien**	**Gewich-tung**
1.1	Das Programm ist gemäss Aufgaben-stellung auf dem Board ausführbar.	1/4
1.2	Fragestellung beantwortet	1/4
1.3	Fragestellung beantwortet	1/4
1.4	Fragestellung beantwortet	1/4