

CT Praktikum: Modulare Programmierung – Linker

1 Einleitung

In diesem Praktikum realisieren Sie auf dem Display des SPI-Praktikums das Spiel *Tic-Tac-Toe*.

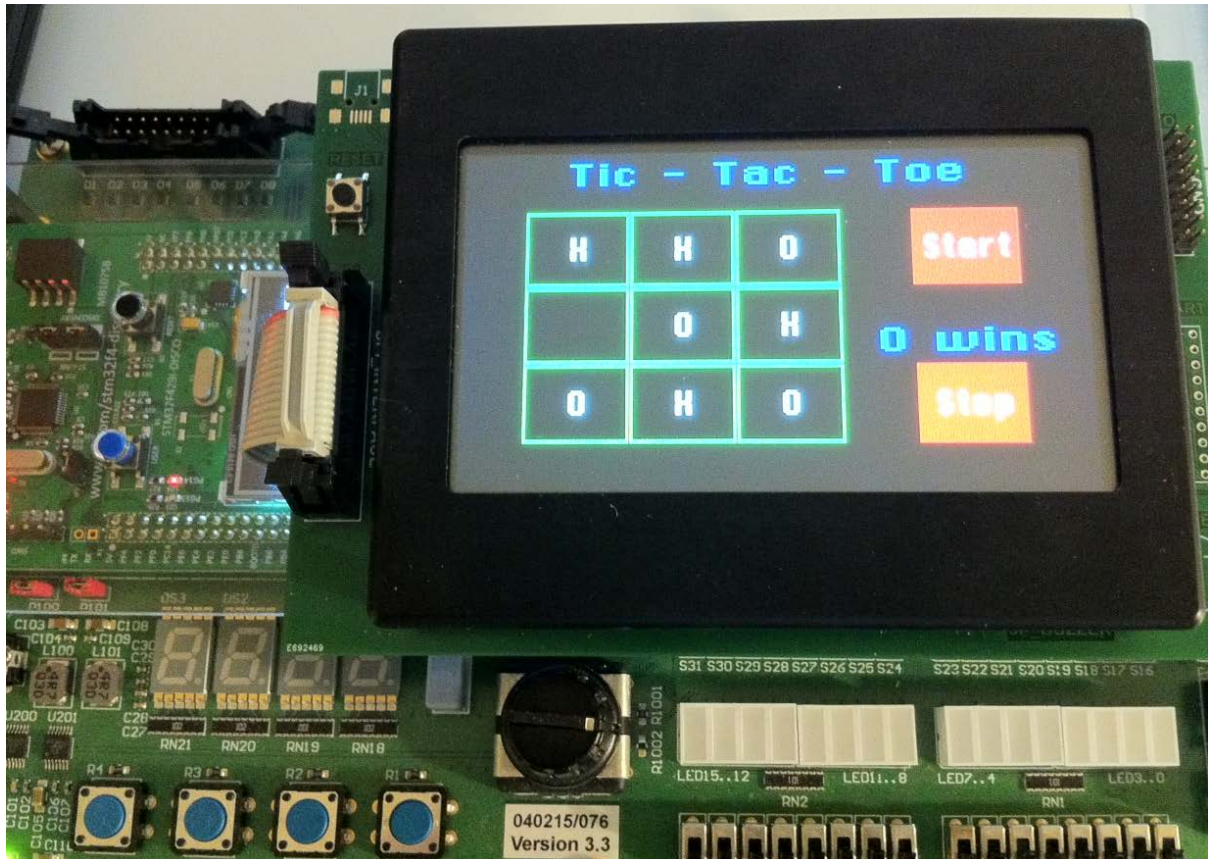


Abbildung 1: Spielbrett

In diesem Spiel setzen zwei Spieler (X und O) abwechselnd einen ihrer Steine auf dem 3x3 Spielbrett (siehe Abbildung 1). Derjenige gewinnt, der als Erster eine Spalte, eine Reihe oder eine Diagonale mit seinen Steinen besetzt.

Architektur

Das Praktikum zeigt ein Beispiel modularer Programmierung anhand einer Variante der Model-View-Controller Architektur (MVC).

- Das **Model** beinhaltet die Logik des Spiels (Spielzüge verwalten, Gewinner ermitteln).
- Die **View** präsentiert das Spiel und nimmt User-Eingaben entgegen.
- Der **Controller** vermittelt zwischen **Model** und **View**.

Ein Ziel dieser Architektur ist es, die View austauschbar zu machen. Z.B. könnte der SPI-Touchscreen durch ein anderes geeignetes Display und Eingabe ersetzt werden, ohne dass der Rest (Model und Controller) geändert werden müsste. Siehe Abbildung 2: Model View Controller (MVC) Architektur (die Pfeile bedeuten „uses“ oder „calls“).

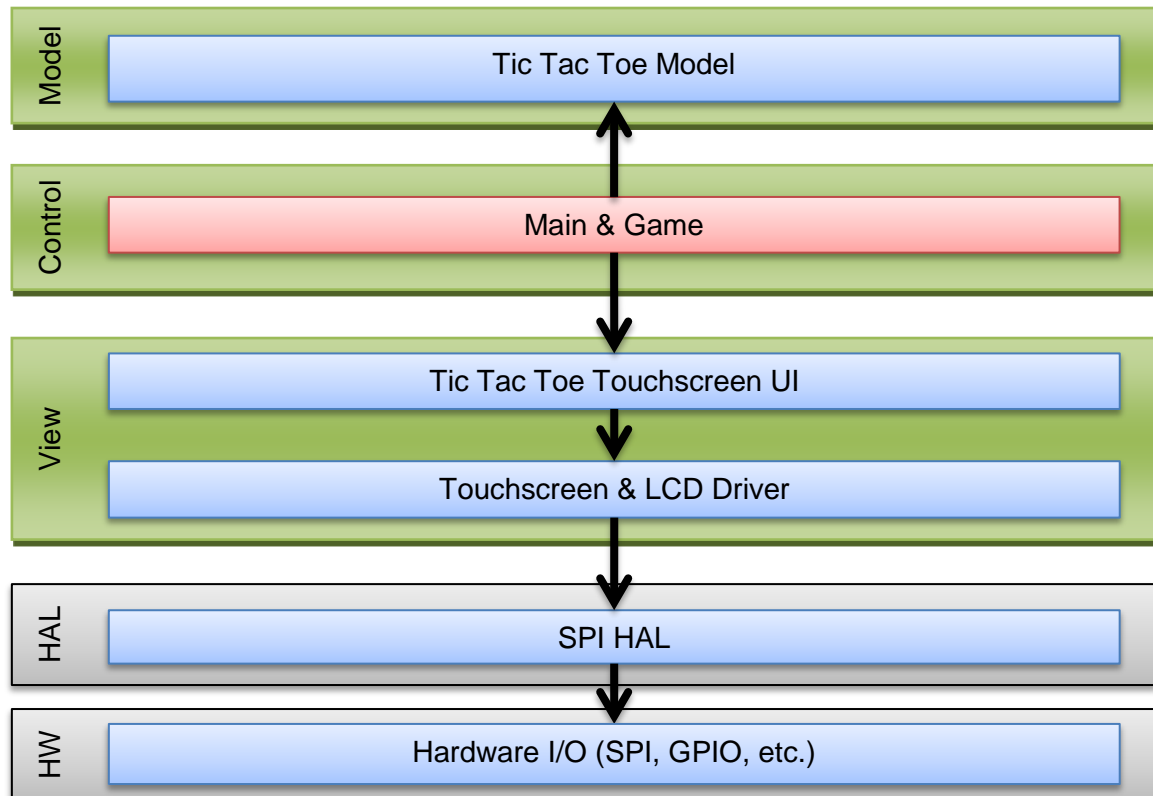


Abbildung 2: Model View Controller (MVC) Architektur

Das Projekt besteht aus Libraries mit dazu passenden Header Files. Ein Teil des Controllers ist als C-Sourcen vorhanden. Der fehlende Teil ist von Ihnen zu implementieren.

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

3.1 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 benötigten Libraries mit Spaces getrennt in diesem Feld ein (z.B. *lib\tictactoe.lib lib\...*)

Wenn Sie alle Fehler erfolgreich korrigiert haben und das *main.c* Programm gemäss *game.h* implementiert haben, können Sie das Spiel auf das CT-Board laden.

Das SPI Touchscreen LCD muss an **Port P5** angeschlossen werden.

Lassen Sie das Spiel laufen und prüfen Sie die Funktion ☺!

3.2 TicTacToe Debugging

Versuchen Sie das Spiel im Debugger in Einzelschritten auszuführen. Ab einer gewissen Zeile zeigt sich ein ungewöhnliches Verhalten.

Was beobachten Sie?

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

Schliesslich ersetzen sie die Referenz durch

lib_debug_with_src\tictactoe_ui_touchscreen.lib. Was beobachten Sie wenn Sie nun kompilieren, linken und debuggen?

Beim debugging mit sourcen müssen sie dem Debugger angeben wo sich die source files genau befinden. In der Library selbst befinden sich keine source files, lediglich die Symbole. Geben sie im *Command Window* des Debuggers folgenden Befehl ein:

```
set src = C:\....\lib_debug_with_src
```

Der Debugger weiss nun wo sich die source files befinden. Die C Datei muss manuell geöffnet werden, um Breakpoints zu setzen.

Was ist der Grund für das veränderte Verhalten?

3.3 TicTacToe Library Symbole extrahieren

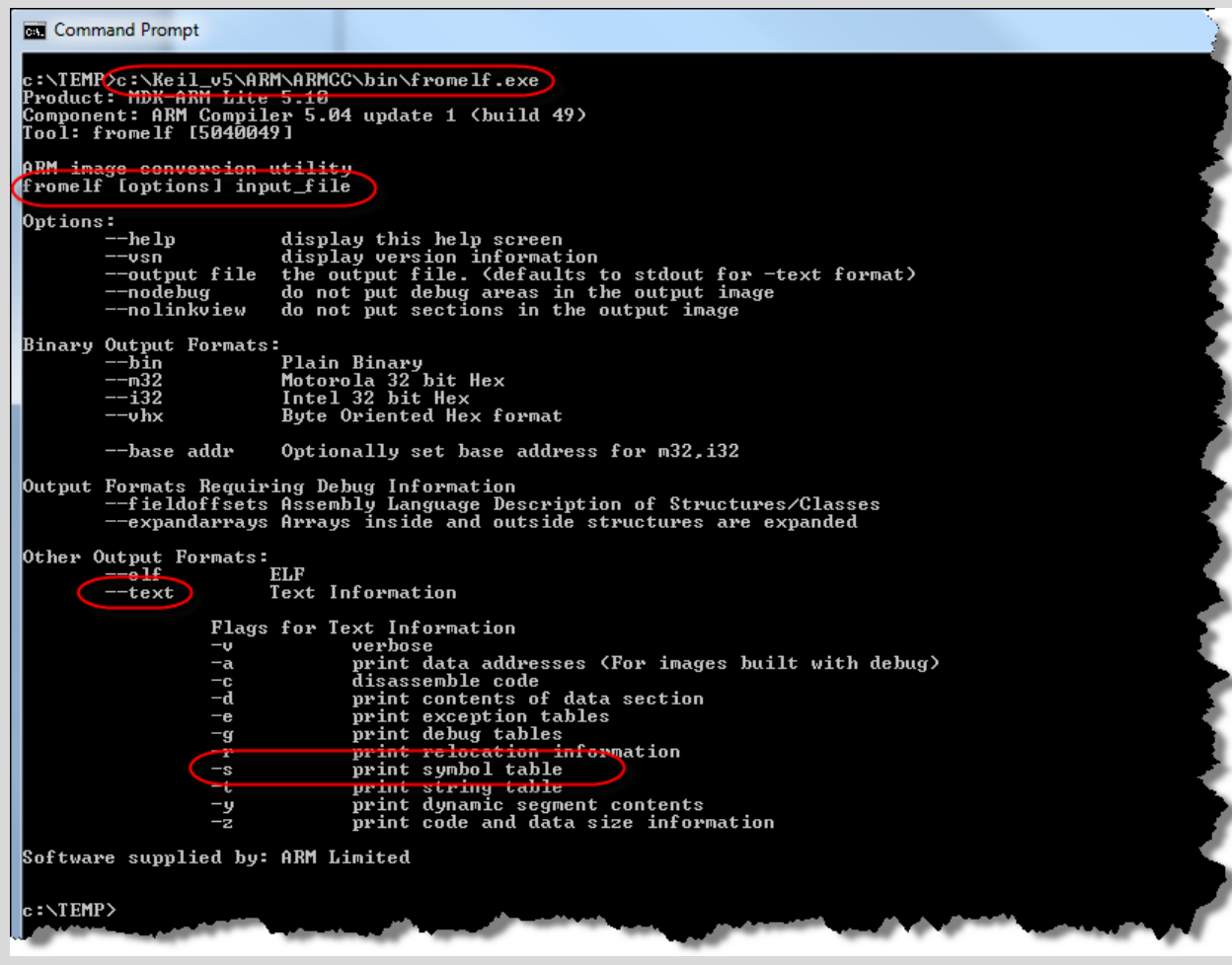
Das Tool *fromelf.exe* kann den Inhalt der binären ELF Dateien in lesbarer Form ausgeben. Die Objekt Dateien (file.o), die Libraries (file.lib) und die Programme (file.axf) sind alle in ELF File Format gegeben.

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.



```
c:\TEMP>c:\Keil_v5\ARM\ARMCC\bin\fromelf.exe
Product: MDK-ARM Lite 5.10
Component: ARM Compiler 5.04 update 1 (build 49)
Tool: fromelf [5040049]

ARM image conversion utility
fromelf [options] input_file

Options:
--help          display this help screen
--vsn           display version information
--output file   the output file. (defaults to stdout for -text format)
--nodebug       do not put debug areas in the output image
--nolinkview    do not put sections in the output image

Binary Output Formats:
--bin           Plain Binary
--m32          Motorola 32 bit Hex
--i32          Intel 32 bit Hex
--vhx          Byte Oriented Hex format
--base addr    Optionally set base address for m32,i32

Output Formats Requiring Debug Information
--fieldoffsets Assembly Language Description of Structures/Classes
--expandarrays Arrays inside and outside structures are expanded

Other Output Formats:
--elf          ELF
--text        Text Information

Flags for Text Information
-v            verbose
-a            print data addresses <For images built with debug>
-c            disassemble code
-d            print contents of data section
-e            print exception tables
-g            print debug tables
-r            print relocation information
-s            print symbol table
-t            print string table
-y            print dynamic segment contents
-z            print code and data size information

Software supplied by: ARM Limited

c:\TEMP>
```

Stellen Sie damit für die Library *lib\tictactoe_ui_touchscreen.lib* die Symbol Tabelle dar.

Welches sind lokale Symbole?

Welches sind exportierte Symbole?

Welches sind importierte Symbole (referenzierte Symbole)?

Lokal	Importiert	Exportiert

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?

--

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?

--

3.4 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.

Erklären Sie anhand der TicTacToe Linker Map Files wie das Memory Map aussieht.

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

3.5 Bewertung

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

Bewertungskriterien	Gewichtung
TicTacToe kann gespielt werden wie in Aufgabe 3.1 gefordert.	1/4
Antworten gegeben für Aufgabe 3.2	1/4
Antworten gegeben für Aufgabe 3.3	1/4
Antworten gegeben für Aufgabe 3.4	1/4