Zielstellung

Zur Einführung in die Programmierung mit SiSy XMC soll Ihnen dieser Schnelleinstieg helfen. Damit können Sie ein erstes Programm für ARM-Mikrocontroller erstellen.

Das Ziel ist es, eine LED auf der Platine XMC4500 Relax Lite Kit einzuschalten. Die LED ist bereits fest mit dem Pin 1.0 verbunden.

Voraussetzungen

Für die Bearbeitung der Aufgaben benötigen Sie folgende Software und Hardware:

Software
- SiSy ab Version 3
- SiSy-Ausgabe XMC, Microcontroller++ oder Professional

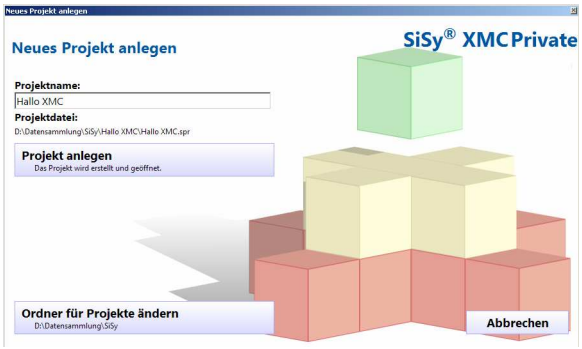
Hardware
- XMC4500 Relax Lite Kit
- 1 Micro USB-Kabel

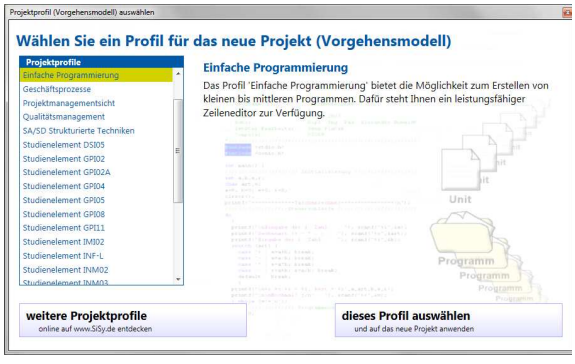
Schaltung:
- LED an Pin 1.0

Im SiSy LibStore finden Sie Beispielprogramme und Programmvorlagen zum Download, die kontinuierlich aktualisiert werden. Eine ausführliche Beschreibung zum SiSy LibStore und den Hilfefunktionen, z.B. Syntax zu Befehlen oder Druckmöglichkeiten, finden Sie im Benutzerhandbuch von SiSy

1. Ein neues Projekt anlegen

Starten Sie SiSy und wählen Sie den Menüpunkt „Neues Projekt erstellen“, vergeben Sie den Projektnamen „Hallo XMC“ und bestätigen Sie mit „Projekt anlegen“. Aktivieren Sie das Profil „Einfache Programmierung“ und bestätigen Sie mit „Ok“.

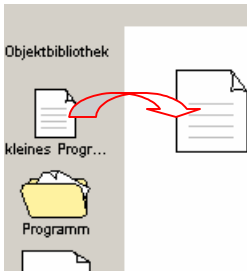




Es öffnet SiSy LibStore und bietet Vorlagen zur Auswahl an.

Arbeiten Sie hier ohne Vorlage und kehren zu SiSy zurück.

2. Kleines C-Programm anlegen



Erstellen Sie ein Programm für den ARM Mikrocontroller, indem Sie per Drag&Drop aus der Objektbibliothek ein Objekt „kleines Programm“ in das Diagrammfenster ziehen. In dem aufgeblendeten Dialogfenster geben Sie den Namen „Hallo“ ein. Der Datei- und Programmname wird dabei automatisch vergeben. Wählen Sie die Sprache ARM C++.

Im nächsten Fenster wählen Sie die Hardware „XMC4500 Relax Lite Kit“ und den Programmer „J-Link“ aus. Aktivieren Sie im nächsten Fenster „Grundgerüst“ und „Struktur laden“.

Das geladene Programmgerüst steht Ihnen nun uneingeschränkt für die Weiterverarbeitung zur Verfügung. Beginnen Sie mit der Vervollständigung im Programmkopf, z.B.

```
//-----------------------------------------
// Titel     : Hallo
//-----------------------------------------
// Funktion  : eine LED einschalten
// Schaltung : LED an Port 1 Bit 0
//-----------------------------------------
// Hardware  : XMC4500
// Takt      : 120 MHz
// Sprache   : ARM C
// Datum     : heute
// Version   : 1
// Autor     : ich
//-----------------------------------------
#include <stddef.h>
#include <stdlib.h>
#include "hardware.h"
```

3. Grundlagen

GPIO (General Purpose Input/Output) ist ein allgemeiner Pin, der als Kontakt aus dem Controllergehäuse herausgeführt. ist. Der XMC4500 verfügt, je nach Gehäuseform, über die GPIO-Ports *PORT 0* bis *PORT 6* und *über PORT 14* und *PORT 15*.
Die Steuerung der Datenrichtung des Pins erfolgt über das Input Output Control Register IOCR. Die Namen der jeweiligen IOCR-Register orientieren sich an dem Basis-Pin, welches als Erstes abgebildet wird.
Das Einschalten der LED erfolgt mit der Ausgabe über das OUT-Register.

```
PORT1->OUT |= (GPIO_OUTPUT_LEVEL_HIGH<<0);
```

Hinweis:

Ausführliche Erläuterungen zu den Grundlagen finden Sie in dem „myXMC Lehrbuch“ und in dem „XMC Tutorial“

4. Quellcode erstellen

Das Ausgabegerät (LED) soll vom Prozessorport GPIOD gesteuert werden. Die Realisierung erfolgt über GPIO Port 1 Pin 0.

Ergänzen Sie die Programmvorlage mit nachfolgend aufgeführtem Quellcode.

```c
#include <stddef.h>
#include <stdlib.h>
#include "hardware.h"

void initApplication()
{
 // u.a. nötig für waitMs(..) und waitUs(..)
    SysTick_Config(SystemCoreClock/100);
 // weitere Initialisierungen durchführen

 // Port 1 Input Output Control Register, Bit 0 = OUT
  PORT1->IOCR0 |= (GPIO_OUTPUT_TYPE_PUSH_PULL<<0*8);
 // Port 1 Output Register, Bit 0 = on
  PORT1->OUT   |= (GPIO_OUTPUT_LEVEL_HIGH<<0);
}

int main(void)
{
    SystemInit();
    initApplication();
    do{

        // bleibt erst mal leer

    } while (true);
    return 0;
}

extern "C" void SysTick_Handler(void)
{
    // Application SysTick
    // bleibt vorerst auch leer
}
```

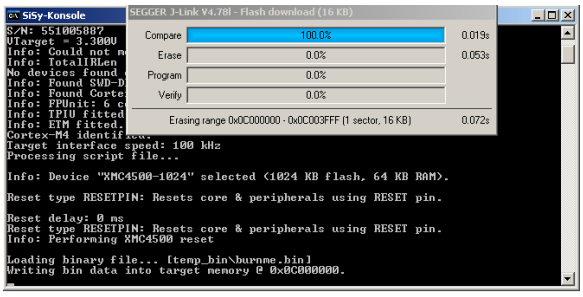
5. Kompilieren und Linken

Der eingegebene Quellcode muss nun in Maschinencode für den ARM Prozessor übersetzt werden. Wählen Sie dazu die Schaltflächen „Kompilieren“ und „Linken“. Bei fehlerfreier Übersetzung liegt das Programm unter dem Namen „Hallo.elf“ vor und kann auf den FLASH-Programmspeicher des Prozessors gebrannt werden..

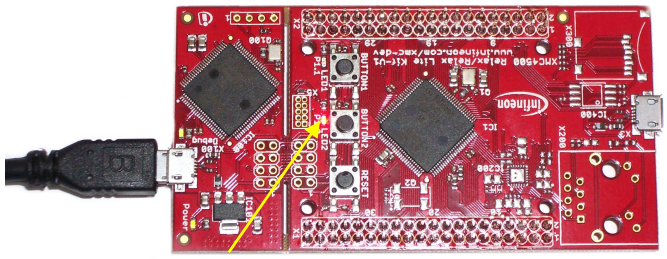
6. Hardware anschließen, brennen und testen

Das Board XMC4500 Relax Lite Kit verfügt über eine ISP (In System Programming) Schnittstelle. Der Prozessor muss also nicht für die Programmierung aus dem System entfernt werden, um ihn in einem gesonderten Programmiergerät zu brennen, sondern kann auf dem Board direkt programmiert werden.

Verbinden Sie das Board mit dem Micro USB-Kabel an dem USB-Port Ihres Rechners. Zum Brennen wählen Sie die Schaltfläche „Brennen“. Der Fortschritt beim Brennen wird angezeigt.



Im Anschluß an das „Brennen“ wird das Programm gestartet. In diesem Fall leuchtet die rote LED neben dem mittleren Taster.



7. Das Programm erweitern

Im nächsten Programm soll die LED nicht nur eingeschaltet werden, sondern auch blinken. Dafür nutzen wir eine Wartefunktion, um das Blinken in der Mainloop zu realisieren. In SiSy gibt es dafür die Funktion WaitMs.
Das Ein- und Ausschalten der LED erfolgt durch Setzen der entsprechenden Bits im OUT-Register:
Einschalten mit bitweiser OR-Verknüpfung:

```
PORT1->OUT |= BIT0;
```

Ausschalten mit invertierter Bitmaske und bitweise AND-Verknüpfung:

```
PORT1->OUT &= ~BIT0;
```

Daraus ergibt sich folgender Quellcode:

```c
//-----------------------------------------------
// Titel     : Blinken mit dem XMC4500
//-----------------------------------------------
// Funktion  : LED blinkt
// Schaltung : LED an Port 1, Bit 0
//-----------------------------------------------
// Hardware  : XMC4500 Releax kit
// Takt      : 120 MHz
// Sprache   : ARM C
//-----------------------------------------------
#include <stddef.h>
#include <stdlib.h>
#include "hardware.h"

void initApplication()
{
  // u.a. nötig für waitMs(..) und waitUs(..)
    SysTick_Config(SystemCoreClock/100);
  // Port 1 IOCR, Bit 0 = OUT
PORT1->IOCR0 |= (GPIO_OUTPUT_TYPE_PUSH_PULL << 0);
}

int main(void)
{
    SystemInit();
    initApplication();
    do
    {
   PORT1->OUT   |= BIT0;    // Port 1 OR, Bit 0 = 1
   waitMs(200);             // 1. Halbzyklus
   PORT1->OUT   &= ~BIT0;   // Port 1 OR, Bit 0 = 0
   waitMs(200);             // 2. Halbzyklus
     } while (true);
    return 0;
}

extern "C" void SysTick_Handler(void)
{
// Application SysTick
}
```

Hinweis:

Das ausführliche Tutorial finden Sie unter

www.myXMC.de