Zielstellung

Zur Einführung in die Programmierung mit SiSy STM32 soll Ihnen dieser Schnelleinstieg helfen. Damit können Sie ein erstes Programm für ARM-Mikrocontroller erstellen.

Das Ziel ist es, 2 LEDs auf der Platine STM32F4-Discovery in kurzen Zeitabständen aufleuchten zu lassen und damit ein „Blinklicht“ zu erzeugen.

Voraussetzungen

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

Software

- SiSy-Ausgabe STM32 (ARM), Microcontroller++, Professional oder Developer ab der Version 3.4
- Ab Windows XP bis Windows 8
- Internet Explorer 7 oder höher
- Installierter USB-Treiber

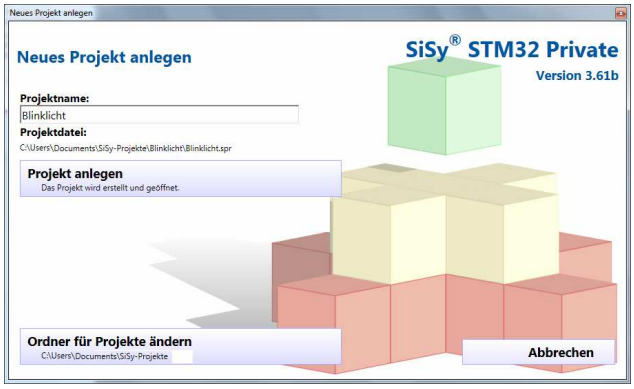
Hardware

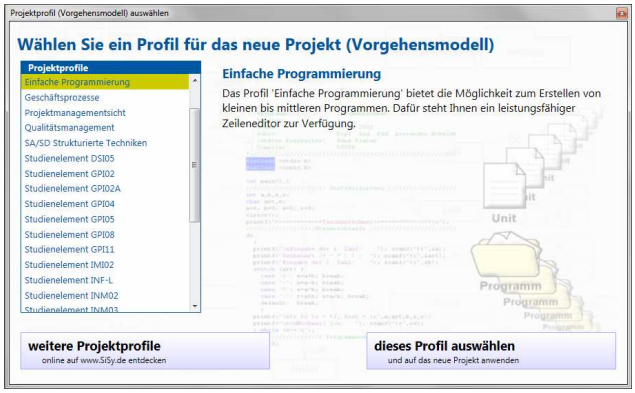
- 1 STM32F4-Discovery
- 1 Mini USB-Kabel

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

1. Ein neues Projekt anlegen

Starten Sie SiSy und wählen „Neues Projekt erstellen“, vergeben Sie den Projektnamen „Blinklicht“ und bestätigen Sie mit „Projekt anlegen“. Wählen Sie das Profil (Vorgehensmodell) „Einfache Programmierung“ aus.

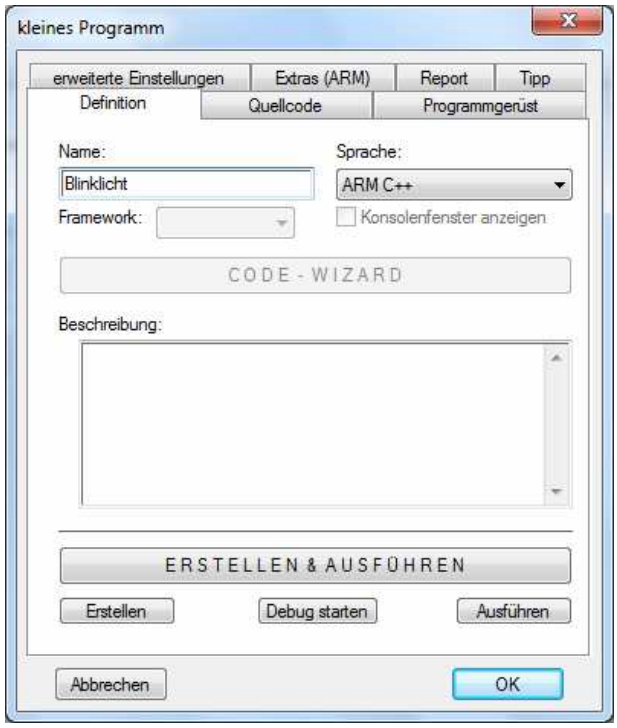




Es öffnet SiSy LibStore und stellt Vorlagen für die weitere Arbeit zur Verfügung. Wir arbeiten an dieser Stelle ohne Vorlagen und kehren zu SiSy zurück.

2. Kleines C-Programm anlegen

Erstellen Sie ein Programm für den STM Mikrocontroller, indem Sie per Drag & Drop aus der Objektbibliothek ein Objekt „kleines Programm“ in das Diagrammfenster ziehen. In dem aufgeblendeten Dialogfenster vergeben Sie den Namen „Blinklicht“. Der Datei- und Programmname wird dabei automatisch vergeben. Wählen Sie die Sprache ARM C++. Im nächsten Dialogfenster wählen Sie die verwendete Hardware aus, in unserem Fall „STM32F407-Discovery“ und den Programmer ST-Link. Im anschließenden Diagrammfenster markieren Sie „Grundgerüst“ und aktivieren die Schaltfläche „Struktur laden“. Damit sind alle Einstellungen erledigt und ein Programm-Grundgerüst steht zur Verfügung. Zur Kontrolle können alle Einstellungen noch einmal kontrolliert werden, indem Sie das Objekt „kleines Programm“ markieren und mittels rechter Maustaste „Definieren“ auswählen. Sie erhalten folgendes Fenster:



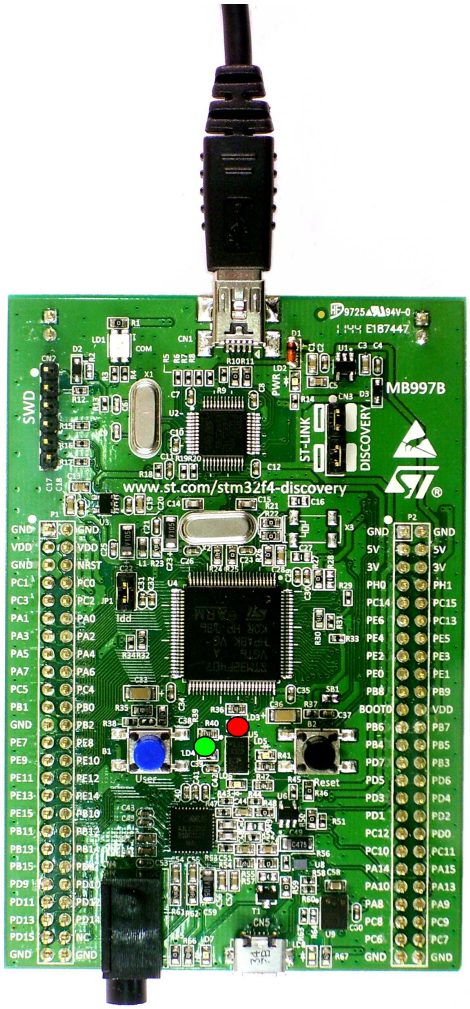
Das geladene Programmgerüst steht Ihnen nun uneingeschränkt für die Weiterverarbeitung zur Verfügung.

3. Quellcode erstellen

Die Ausgabegeräte (LEDs) sollen vom Prozessorport GPIOD gesteuert werden. Die Realisierung erfolgt über GPIO Pin 12 und 13.

Ergänzen Sie die Programmvorlage mit nachfolgend aufgeführtem Quellcode.

```
//-----------------------------------------
// Titel     : Beispiel Blinky
//-----------------------------------------
// Funktion  : lässt die rote und grüne LED blinken
// Schaltung : LEDs an GPIO Port D12, D13
//-----------------------------------------
// Hardware  : STM32F4 Discovery
// Takt      : 168 MHz
// Sprache   : ARM C
// Datum     : heute
// Version   : 1
// Autor     : ich
//-----------------------------------------
#include <stddef.h>
#include <stdlib.h>
#include "stm32f4xx.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen

    /* GPIOD Takt einschalten  */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

    /* Konfiguriere GPIO Port D15 */
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_StructInit (&GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_12|GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
 }

void delay(vu32 nCount)
{
    while (nCount--);
}

int main(void)
{
    SystemInit();
    initApplication();
    GPIO_SetBits(GPIOD,GPIO_Pin_13);
    do{
        GPIO_ToggleBits(GPIOD,GPIO_Pin_12|GPIO_Pin_13);
        delay(10000000);

    } while (true);
    return 0;
}

extern "C" void SysTick_Handler(void)
{
    // Application SysTick default 10ms
}
```



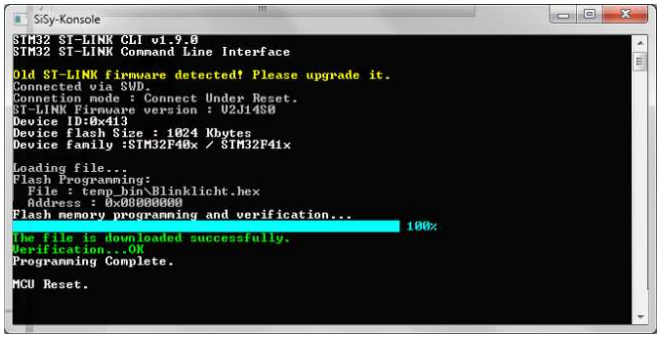
4. Kompilieren und Linken

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

5. Hardware anschließen und brennen

Das Board STM32F4-Discovery 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 im STM32F4 direkt programmiert werden.

Verbinden Sie das Board mit dem Programmierkabel an dem USB-Port Ihres Rechners. Zum Brennen wählen Sie die Schaltfläche „Brennen“. In Abhängigkeit Ihrer Konfiguration erhalten Sie im Ausgabefenster eine entsprechende Meldung:



6. Mikrocontrollerlösung testen

Zum Testen des Programms ist es erforderlich, dass das STM32F4-Discovery angeschlossen ist. Das Programm startet automatisch und die LEDs auf Ihrem Board blinken.

Hinweis:

Nutzen Sie die in SiSy integrierten Tutorials!