Versuch 5 : Ausblick

Aufgabe 5_1 : CMSIS

Bei den bisherigen Programmen des Praktikums wurde direkt auf die IO-Register der verschiedenen IO-Bausteine wie z.B. GPIOB, UART1, Timer2 usw. zugegriffen.
Für die Mikrocontroller der Cortex –Reihe wird von ARM eine Bibliothek angeboten, die die Verwendung der IO-Bausteine erleichtert. Diese Library heißt CMSIS (**C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard) und ist ein herstellerunabhängiger Hardware Abstraction Layer für die Cortex-M Prozessoren.

Um den Umgang mit dieser Bibliothek kennenzulernen sollen Sie das Modul USART.c aus dem Versuch2 mit Hilfe der CMSIS –Bibliothek programmieren und in das Programm des letzten Praktikumstermins einbinden.

Nehmen Sie als Basis das Projekt V5.1_CMSIS. Hier ist der Versuch v4.3 komplett mit der Verwendung der CMSIS-Library umgesetzt.
Im Modul USART1.c müssen nur noch von Ihnen die Funktionen InitUSART1, ReadChar und WriteChar ergänzt werden.

Vervollständigen Sie diese Funktionen und verwenden Sie die passenden Bibliotheks-Funktionen.

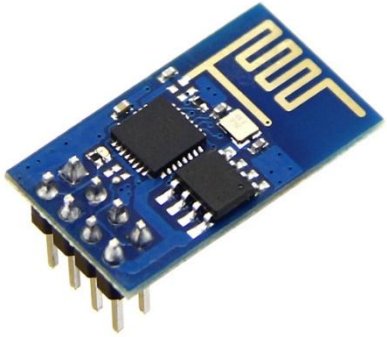
Hinweis:

- Auf der Internetseite http://stm32.kosyak.info/doc/index.html finden Sie eine gute Beschreibung der CMSIS-Funktionen. In den Unterlagen zum Versuch in moodle habe ich Ihnen das USART-Kapitel der CMSIS-Dokumentation zur Verfügung gestellt.

- Die Idee bei der Verwendung von CMSIS ist, dass nie direkt auf die IO-Register zugegriffen wird. Zur Initialisierung wird daher zunächst eine Datenstruktur (z.B. USART_InitStructure) ausgefüllt und dann mit einer Initialisierungsfunktion die Werte in die eigentlichen IO-Register übertragen. Dabei wird die korrekte Bitposition im IO-Register berücksichtigt, die der Programmierer nicht mehr wissen muss. Sie verwenden z.B. die Konstante „USART_Mode_Rx“ beim Belegen der Init-Stucture , um den Receiver der USART-Schnittstelle einzuschalten. Die CMSIS-Bibliothek setzt daraufhin das Bit2 im Register USART_CR1 (siehe Versuch 2).
  Ebenso wird das Einstellen der Baudrate vereinfacht. Sie müssen nur den Wert 9600 angeben, die CMSIS-Bibliothek berechnet daraus die richtige Belegung von Mantisse und Fraktion im BRR-Register.

Aufgabe 5_2 : WLAN

Als letzter Teil des Praktikums wird der STM32 Mikrocontroller mit einem WLAN-Modul ESP8266 ergänzt. Dieser Mikrocontroller enthält den kompletten TCP/IP-Stack und ein WLAN-Modul. In unserem Versuch werden wir den ESP8266 dazu verwenden, um die Eingaben über putty durch die Eingaben über einen Webbrowser (Smartphone, Notebook) zu ersetzen.



Binden Sie dazu das Programmmodul ESP.c in Ihr Projekt ein und fügen mittels #include die Datei ESP.h in das main-Modul ein.
Rufen Sie dann die Funktion „Program_to_ESP(Nummer,inputBuffer,&cmdflag);)“ als letzten Aufruf im Initialisierungsteil des main-Programms auf. Als Parameter geben Sie die Nummer Ihres Arbeitsplatzes, den Inputbuffer des UART1-Handlers sowie die Adresse des cmdflags ein.
Der Aufruf Writestring (V5.1 …) muss entfernt werden (die Ausgabe von Writestring würde ja nun nicht zu putty, sondern zum ESP gehen).

Im USART1-Handler muss als Endezeichen eines Kommandos statt ‘.‘ nun ‘\n‘ verwendet werden, zusätzlich muss die Ausgabe des Echos entfernt werden.

Beim Start des Programms wird ein WLAN-Accesspoint gestartet mit der SSID „ESP-WLAN x“, wobei x die die Arbeitsplatznummer ist.
Das WLAN-Passwort des ESP lautet „1234567890“.

Anschließend müssen Sie auf Ihrem Tablet oder Handy den Webbrowser starten und die URL „192.168.4.1“ aufrufen.

Sie können auf einem Android-System alternativ die App „MCT-Input“ installieren (siehe moodle->Unterlagen V5)