Versuch 3 – GPIO / Bibliothek / Bitmanipulationen

An das Piggyback kann über ein Kabel eine 7-Segmentanzeige mit 4 Stellen angeschlossen werden (Abbildung 1).

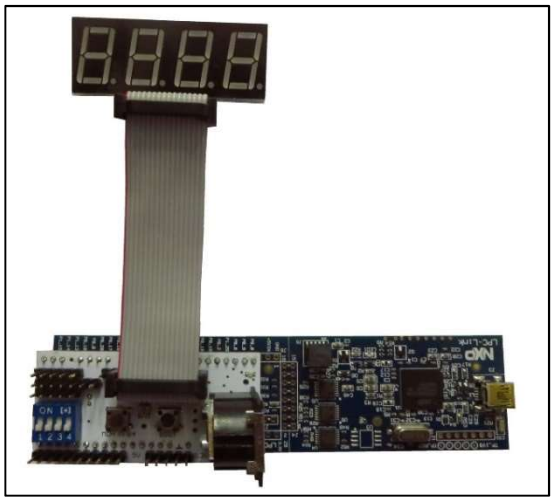


Abbildung 1: Anschluss des LED-Displays

In diesem Versuch soll eine Stelle gesteuert werden können. Man soll mit Hilfe des 4-fach DIL-Schalters (blau) eine vierstellige Dualzahl eingeben können, deren zugehörige Ziffer dann angezeigt wird (0000->0, 1001->9, 1010->A, 1111->F).
Außerdem soll man mit Hilfe der Taste neben dem DIL-Schalter entscheiden können, ob die Ziffer auch wirklich erscheint oder ob stattdessen nur der Dezimalpunkt leuchtet.
Man kann damit die gewünschte Ziffer blind einstellen und sich dann auf Tastendruck anzeigen lassen, ob das Ergebnis der Erwartung entspricht.

1 Vorbereitung

Wechseln Sie in einen neuen, leeren Workspace. Importieren Sie dann aus dem Archiv *v3.zip* die drei Projekte *lpc_chip_11uxx_lib*, *mch_aux_lib* und *v3*.
Falls Sie die Vorbereitung ohne das Programm MCUXpresso machen, dann extrahieren Sie die Projekte aus dem Archiv und sehen sich die im Folgenden erwähnten Dateien in einem Editor an.
Das Projekt *mch_aux_lib* enthält nützliche Makros für die Bitmanipulation (Kochrezepte) sowie eine Funktion zum schnellen Einstellen der Pins/GPIOs. Die Nutzung ist optional. Das Projekt *v3* enthält ein C-Skelett. Darin enthalten ist im Verzeichnis *inc* der Header *mch_pins.h*. Darin finden Sie Definitionen für die Pins (Nummern), die Sie optional ebenfalls zur Arbeitserleichterung verwenden können.
Dieselben Nummern finden Sie auch auf dem Beiblatt *pins_v2-v4.pdf*.

Für diesen Versuch benötigen Sie das Piggyback, das externe Netzteil, das 16-polige Verbindungskabel und die LED-Anzeige. Verbinden Sie das Display mit Hilfe des Kabels mit dem Piggyback (Abbildung 1). Schließen Sie dann das externe Netzteil an. Das Netzteil liefert den Strom für das Display.
Zuletzt schließen Sie den Debugger (in Abbildung 1 auf der rechten Seite) an den Entwicklungsrechner an.

Hinweis:

Sollten Sie sich im Verlauf der Vorbereitung nicht sicher sein, ob Ihre Hardware funktioniert, dann können Sie zum Test jederzeit das Testprogramm *v3.bin* mittels des Bootloaders in den µC laden.

Versuch 3 – GPIO / Bibliothek / Bitmanipulationen

2 Ansteuerung des Displays

Ein Segment einer Ziffer leuchtet, wenn sowohl Anschluss für die Ziffernposition (1-4) als auch der Anschluss für das Segment (A-G und der Dezimalpunkt DP) einen H-Pegel (1) ausgeben.

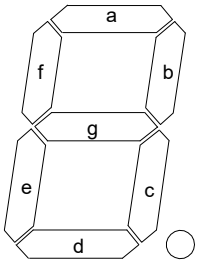
2.1 Einmalige Initialisierung

Initialisieren Sie zunächst alle Anschlüsse für das Display in der Funktion *void v3_init(void)* soweit, dass Sie später im Programm nur noch Werte auszugeben brauchen. Nach der Initialisierung sollen alle Anschlüsse ein Low (0) ausgeben.

Hinweis:
Die schon in der Datei *mch_pins.h* definierten Makros sind vom Typ „Makro ohne Parameter". Das führt zu einer reinen Textersetzung. Ein Makro kann an jeder beliebigen Stelle im Programm verwendet werden, solange es als ein Element (mindestens ein Leerzeichen davor und ein Leerzeichen danach) erkannt werden kann.

2.2 Test der Segmente

Testen Sie jetzt, ob Sie jedes Segment (A - G, DP) der Ziffer an der letzten Stelle des Displays einschalten können. Dazu schreiben Sie am besten die benötigten Anweisungen in das Hauptprogramm nach der Initialisierung. Sie können dann ja mit dem Debugger in Einzelschritten über diese Anweisungen gehen und dabei das Display beobachten. Denken Sie dabei daran, dass auch das Signal für die gewünschte Ziffernposition (hier: 4) ein H ausgeben muss.



Später können Sie diesen Test entweder ganz löschen, auskommentieren oder mit Hilfe der bedingten Kompilierung (Vorlesung!) zeitweise von der Übersetzung ausschließen. Zu Übungszwecken sollten Sie hier die letzte Methode (bedingte Kompilierung) einsetzen.

Hinweis:
Falls das Segment D nicht oder deutlich schwächer als die anderen Segmente leuchtet, dann haben Sie bei der Initialisierung vermutlich einen Fehler gemacht. Lesen Sie dazu die Beschreibung Peripherieregisters des zugehörigen Pins im UM **genau**.

2.3 Muster für die Ziffern und Buchstaben

Notieren Sie in einer Tabelle, welche Segmente pro Ziffer 0 - 9 bzw. pro Buchstaben A - F leuchten müssen (1) bzw. aus bleiben (0).
Diese Informationen sollen später in einem Integer (uint8_t) gespeichert werden. Die Spalte für das Segment a soll der Bitposition 6 entsprechen, die Spalte für das Segment g der Bitposition 0. Die Bitposition 7 des Integer bleibt immer 0. Sie können als Vorlage Tabelle 1 benutzen. Für die Ziffer 0 ist die Tabelle bereits richtig ausgefüllt. Die letzte Spalte soll den jeweiligen 8 Bit Wert in hexadezimaler Schreibweise angeben.

Ziffer	7	6	5	4	3	2	1	0	Hex
0	0	1	1	1	1	1	1	0	0x7E
1	0								
2	0								

Tabelle 1: Bitmuster für Siebensegmentanzeige, abgelegt in einem 8 Bit Integer

2.4 Ziffernanzeige

Für die spätere bequeme Nutzung soll jetzt eine Funktion *void v3_set_pattern(uint8_t x)* geschrieben werden, der man als Parameter das gewünschte Muster übergibt. Die Funktion stellt dann die Anschlüsse A-G am Display passend ein. Mit dem Aufruf *v3_set_pattern(0x7e);* wird also die 0 dargestellt (siehe Muster aus Tabelle 1).

Mit der Funktion *v3_set_pattern* kann man beliebige Muster ausgeben. In diesem Versuch sollen aber fast ausschließlich die Hexadezimalziffern erscheinen. Daher ist eine weitere Funktion (oder eine andere Methode), mit der man einen Wert im Bereich 0-15 in das passende Muster übersetzen kann, eine nützliche Hilfe.
Schreiben Sie eine Funktion *uint8_t v3_get_pattern(uint8_t x)*, der man als Parameter einen Wert von 0 – 15 übergibt und die als Rückgabewert das zugehörige Anzeigemuster liefert. (Es gibt auch andere Lösungen, beispielsweise mit Hilfe eines Feldes. Wie Sie das Problem lösen, ist weniger wichtig als dass es gelöst wird.)

Jetzt können Sie recht einfach die Darstellung aller Ziffern und Buchstaben testen, indem Sie in einer Schleife alle Werte 0 – 15 anzeigen lassen.

3 Einlesen der Dualzahl

Nun sollen die vier Schalter 1-4 abgefragt und daraus ein Wert gebildet werden. Der Wert könnte z. B. in der Variable *zahl* abgelegt werden. Wenn der Schalter S1 auf „on“ steht, soll das Bit 3 der Variable *zahl* auf 1 stehen, sonst auf 0. Wenn S2 auf „on“ steht, dann soll das Bit 2 der Variable *zahl* auf 1 stehen, sonst auf 0. S3 bestimmt analog den Wert des Bits 1 und S4 den Wert des Bits 0.

3.1 Einmalige Initialisierung

Initialisieren Sie zunächst die Pin-Eigenschaften für die vier Schalter. Danach stellen Sie die zugehörigen GPIOs auf die passende Richtung. Erweitern Sie dazu einfach die schon vorhandene Funktion *void v3_init(void).*

Hinweis:
Die Schalter haben dieselben Eigenschaften wie die BOOT-Taste.

3.2 Lesen der Schalterstellungen

Schreiben Sie dann eine Funktion *uint8_t v3_read_dil(void)*, in der die einzelnen Schalterstellungen wie oben angegeben in eine lokale Variable übertragen werden. Die nicht verwendeten Bitpositionen 4 – 7 der Variable bleiben 0. Diese Zahl ist dann der Rückgabewert der Funktion.

Versuch 3 – GPIO / Bibliothek / Bitmanipulationen

4 Zusammensetzen der Programmteile

In einer Endlosschleife können Sie jetzt die Anwendung aus den zuvor geschriebenen Einzelteilen zusammensetzen.

4.1 Sofortige Anzeige der Zahl

Zunächst soll die am DIL-Schalter eingestellte Zahl sofort angezeigt werden. Pro Schleifendurchlauf müssen Sie zunächst die Zahl einlesen, die am Schalter gerade eingestellt ist, dann diese Zahl in das zugehörige Muster übersetzen und zuletzt dieses Muster auf der Anzeige ausgeben.

4.2 Blinde Eingabe

Fragen Sie jetzt in der Endlosschleife die Taste *KEY_2* ab. Ist sie nicht gedrückt, dann wird nur der Dezimalpunkt eingeschaltet. Wenn die Taste gedrückt wird, dann wird wie unter 4.1 die aktuelle Schalterstellung angezeigt. Sie müssen dazu natürlich den Pin und den GPIO für *KEY_2* ebenfalls einmalig einstellen.

Hinweis:
Lesen Sie die Beschreibung des Peripherieregisters für den Pin **genau**, damit Sie nicht eine wesentliche Einstellung übersehen.
Sollten Sie die Abfrage dieser Taste nicht zum Funktionieren bekommen, dann verwenden Sie ersatzweise die BOOT-Taste. Damit können Sie auf jeden Fall die Programmlogik testen.

Lernziele

- Umgang mit der Entwicklungsumgebung
- Einstellen eines Anschlusses des µC, Nutzung eines GPIO
- Verwendung von Bibliotheksfunktionen
- Bitmanipulationen (Setzen/Löschen/Testen)

Material zur Vorbereitung

- User Manual UM10462, Revision 5.3 für den µC
- Pinliste lpc11u_pins.pdf
- Projektarchiv v3.zip für MCUXpresso

Material zur Durchführung

- Basisplatine,
- Piggyback weiß
- LED-4fach-Anzeige
- Flachbandkabel 16polig
- Netzteil mit DIN5-Anschluß
- Entwicklungswerkzeug (hier MCUXpresso Version 10.0.0)