E 308

Versuch 1: Parallel Port, Timer, serielle Schnittstelle, Interrupts

1. Abfragen von Tasten und Ausgeben des aktuellen Tastenzustandes

An den Pins P1H.4-P1H.7 des C164 sind 4 Tasten angeschlossen. Diese Pins sind also als Eingänge zu konfigurieren. Im Ruhezustand liefern die Tasten HIGH-, gedrückt LOW-Pegel.

Für die Nutzung der prellenden Eingabetasten sollen Funktionen mit den folgenden Prototypen realisiert werden:

bit KeyDown() liefert 1 zurück, wenn ein Tastendruck vorliegt
liefert 0 zurück, wenn kein Tastendruck vorliegt

char GetKey(void) liefert das ASCII-Zeichen zurück, das der gedrückten Taste zugeordnet wird – mögliche Werte: '1', '2', '3', '4'.

Eine Veränderung des Tastenzustandes ist wegen des Kontaktprellens nur dann zu akzeptieren, wenn diese länger als 52msec anliegt. Zum Entprellen der Tasten verwenden Sie Timer 2 des GPT1-Moduls, dessen mit einer Periode von 26msec auftretender Überlauf einen Interrupt auslöst. In der ISR wird der aktuelle Tastenzustand festgestellt und untersucht, ob eine zuvor aufgetretene Veränderung bereits zum zweiten Male vorliegt. Erst dann sollte ein Aufruf der Funktion *KeyDown()* den Tastendruck melden.

$2^{16} \cdot 0{,}4\mu s$ (Überlauf)

Typische Anwendung der Tasten:

```
.....
if(KeyDown()) {
    switch(GetKey()) {
        case '1':    printf("Taste 1 ist gedrückt!!\n\r");
.....
```

a) Realisieren Sie im Dave-Modul für des GPT1-Systems (C- und Header-Datei) die obigen Funktionen!
Testen Sie diese mit der oben auszugsweise aufgeführten Testsequenz zunächst in der Simulation – Timer-Interrupt, Entprellen der Taste, Flag-Behandlung usw.!
Erst nach erfolgreicher Simulation wird in der Schaltung getestet!

An den Pins P1L.4-P1L.7 sind LED's angeschlossen und leuchten bei Anlegen eines LOW-Pegels. P1L.4-P1L.7 sind also Ausgangssignale.

b) Ordnen Sie jeder Taste eine LED zu, die immer dann ihren Zustand ändert, wenn diese Taste gedrückt wird!

2. Ausgabe zum LC-Display

Die Textausgabe mittels Aufruf der Standardfunktion "*printf()*" hat den Nachteil, dass nachfolgende Anweisungen erst ausgeführt werden, wenn alle durch "*printf()*" formatierten ASCII-Zeichen über die serielle Schnittstelle ausgegeben sind. Bei 9600 Baud dauert die Ausgabe eines Zeichens ca. t_z = 10 Bit * 1/9600 sec/Bit ≈ 1msec. In vielen Anwendungen sind solche Zeiten des Wartens auf ein Ereignis (hier: serielle Schnittstelle ist bereit ein weiteres Zeichen auszugeben) nicht tolerierbar.

bitte umblättern!

Im konkreten Fall sollte die Lösung des Problems darin bestehen, dass die Ausgabe interrupt-gesteuert erfolgt. D. h. durch Ausgabe eines ersten Zeichens wird die Übertragung angestossen, alle weiteren Zeichen werden in der Interrupt-Service-Routine ausgegeben. Dies erfordert, dass der auszugebende Text in einem geeignet dimensionierten Puffer bereitgestellt wird, dies kann z.B. mittels "sprintf()" durchgeführt werden.

Das für die Ausgabe vorhandene LCD-Modul besitzt 4 Zeilen mit maximal 20 Zeichen. Unter anderen sind folgende Sonderzeichen zur Steuerung der LCD-Anzeige verfügbar:

Alles Löschen und Cursorposition 1.Zeile, 1.Spalte einstellen:	'\xC'
Cursor in Spalte s(1..20) der Zeile z(1..4) positionieren:	'\x1B' 'O' s z

c) Erweitern Sie die serielle Ausgabe (das Dave-produzierte Modul ist entsprechend zu erweitern) in geeigneter Weise für die Ausgabe auf das LC-Display. Folgende Funktionen sind zu realisieren:

```
void DoPrintZ (int iZnr, char *pBuf); // Text in pBuf auf Zeile iZnr ausgeben
void ClrScr(void);                    // Display löschen
```

Hinweis:
Ordnen Sie dem Display einen Puffer zu, der geeignet so zu dimensionieren ist, dass er genau eine auszugebende Zeile aufnehmen kann (inklusive u.U. nötiger Steuerzeichen)! Werden längere Texte übergeben, so werden diese auf eine Zeilenlänge gekürzt. Kurze Texte werden auf die Zeilenlänge durch Blanks erweitert.