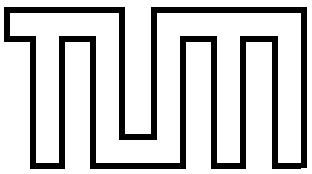

Technische Universität
München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Aufgabe 8 – DCF77-Decoder

Dr. Christian Buckl
buckl@in.tum.de

Simon Barner
barner@in.tum.de

Michael Geisinger
geisinge@in.tum.de

Stephan Sommer
sommerst@in.tum.de

Wintersemester 2008/09

Aufgabe 8: DFC77-Decoder

Die Zielstellung der vorliegenden Aufgabe ist es, das DFC77-Zeitsignal mit einem Mikrocontroller auszuwerten und die aktuelle Uhrzeit über die serielle Schnittstelle auszugeben.

Der Zeitsignal-Sender DCF77

Der Zeitsignal-Sender DCF77 ist ein Langwellensender, der von der Physikalisch-Technischen Bundesanstalt (PTB) Braunschweig (in Zusammenarbeit mit einer ehemaligen Abteilung der Telekom) betrieben wird. Mit Hilfe dieses Senders, der mit einer Frequenz von 77,5 kHz und einer Leistung von 50 kW arbeitet, erfüllt die PTB ihre Verpflichtung zur Verbreitung des amtlichen Zeitsignals, so wie es im *Gesetz über die Zeitbestimmung* (25.7.1978) festgeschrieben ist.

Wegen der guten Ausbreitungseigenschaften auch in Gebäuden und relativ unabhängig von der Umgebung ist der Langwellensender DCF77 (D = Deutschland, C = Langwelle, F = Frankfurt, 77 = Trägerfrequenz) gut zur Verbreitung der in Deutschland offiziell gültigen Zeit geeignet.

Der Sender ging 1959 in Betrieb und wurde zunächst mit dem Zeitsignal einer Quarz-Uhr versorgt. Seit 1969 wird das Zeitsignal von der PTB in Braunschweig mittels vier hochgenauer Cäsium-Atomuhren (*CS1* bis *CS4*) erzeugt. Während bereits seit 1973 über den Sender zusätzlich auch Datum und Uhrzeit übermittelt wurden, werden seit 2006 in bislang für die Verbreitung von Betriebsinformationen des Senders reservierten Bits Wetterinformationen sowie Katastrophenwarnungen übermittelt.

Der redundant ausgelegte Sender befindet sich in der Sendefunkstelle Mainflingen (Koordinaten: 50°01′ Nord, 09°00′ Ost) etwa 25 km südöstlich von Frankfurt am Main.

Versuchsaufbau

Der Versuchsaufbau für das vorliegende Aufgabenblatt besteht aus den folgenden Komponenten:

- Empfänger für das DFC77-Signal
- Atmel STK500-Testplatine mit Atmel ATmega8515-Mikrocontroller zur Auswertung des Zeitsignals
- Host-PC zur Programmierung des Controllers und Anzeige der aktuellen Uhrzeit (über die serielle Schnittstelle)

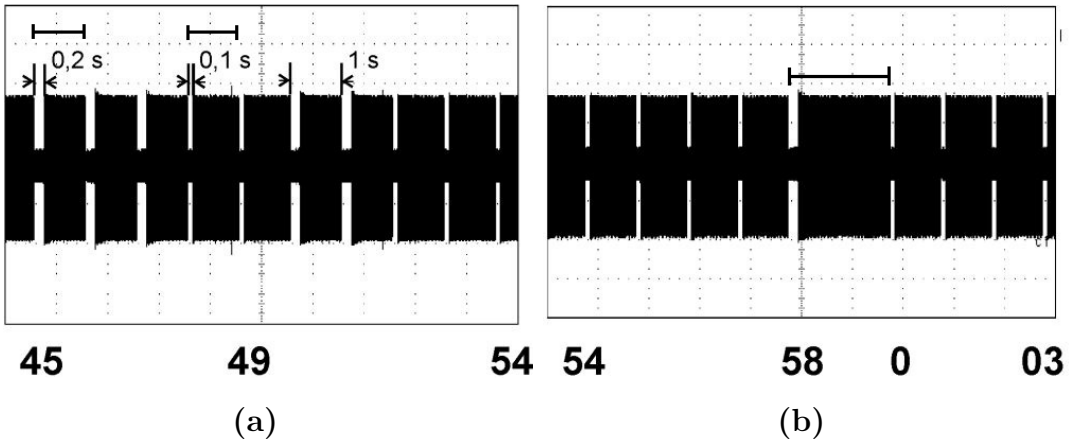

Abbildung 1: DFC77-Signal: **(a)** Sekundenmarken: **Eins** (200 ms low gefolgt von 800 ms high (links)) und **Null** (100 ms low gefolgt von 900 ms high (rechts)).
(b) Minutenmarke: **Null**, in der 59. Sekunde erfolgt keine Absenkung, Bit ist konstant **Null**.

Das Zeitsignal

Die Amplitude der DCF77 Trägerschwingung wird mit Sekundenmarken moduliert: Zu Beginn jeder Sekunde, mit Ausnahme der letzten Sekunde jeder Minute als Kennung für den folgenden Minutenbeginn, wird die Amplitude für die Dauer von 100 ms oder 200 ms phasensynchron mit der Trägerschwingung auf etwa 25 % abgesenkt (siehe Abbildung 1).

Die unterschiedliche Dauer der Sekundenmarken dient zur binären Kodierung von Uhrzeit und Datum.

- Sekundenmarken der Dauer 100 ms entsprechen einer binären **Null**, solche der Dauer 200 ms einer binären **Eins**.
- In der 59. Sekunde erfolgt keine Absenkung, wodurch die nachfolgende Sekundenmarke den Beginn einer Minute kennzeichnet und der Empfänger synchronisiert werden kann.
- Einmal während jeder Minute werden die Nummern von Minute, Stunde, Tag, Wochentag, Monat und Jahr BCD-kodiert übertragen.
- Vom Kalenderjahr werden nur die Einer- und Zehnerstelle übertragen (Jahr 2000-Problem!).
- Der ausgesendete Code besteht also aus 60 Zeichen (von denen nur 59 tatsächlich gesendet werden) und enthält jeweils die Information für die folgende Minute.

Die genaue Bedeutung der einzelnen Felder sind Tabelle 1 und Abbildung 2 zu entnehmen.

Bit	Anzahl Bits	Bedeutung der Werte
0	1	Start einer neuen Minute (ist immer 0)
1-14	14	bis 1977: Differenz UT1-UTC; ab 2006: Betriebsinformationen seit Ende 2006: Wetter- sowie Katastrophenschutz-Informationen
15	1	Rufbit (bis Mitte 2003: Reserveantenne)
16	1	1: Am Ende dieser Stunde wird MEZ/MESZ umgestellt.
17	1	0: MEZ, 1: MESZ
18	1	0: MESZ, 1: MEZ
19	1	1: Am Ende dieser Stunde wird eine Schaltsekunde eingefügt.
20	1	Beginn der Zeitinformation (ist immer 1)
21-27	7	Minute (Werte der einzelnen Bits: 1, 2, 4, 8, 10, 20, 40)
28	1	Parität Minute
29-34	6	Stunde (Werte der einzelnen Bits: 1, 2, 4, 8, 10, 20)
35	1	Parität Stunde
36-41	6	Monatstag (Werte der einzelnen Bits: 1, 2, 4, 8, 10, 20)
42-44	3	Wochentag (Werte der einzelnen Bits: 1, 2, 4)
45-49	5	Monat (Werte der einzelnen Bits: 1, 2, 4, 8, 10)
50-57	8	Jahr (zweistellig; Werte der einzelnen Bits: 1, 2, 4, 8, 10, 20, 40, 80)
58	1	Parität Datum

Tabelle 1: DFC77-Kodierung

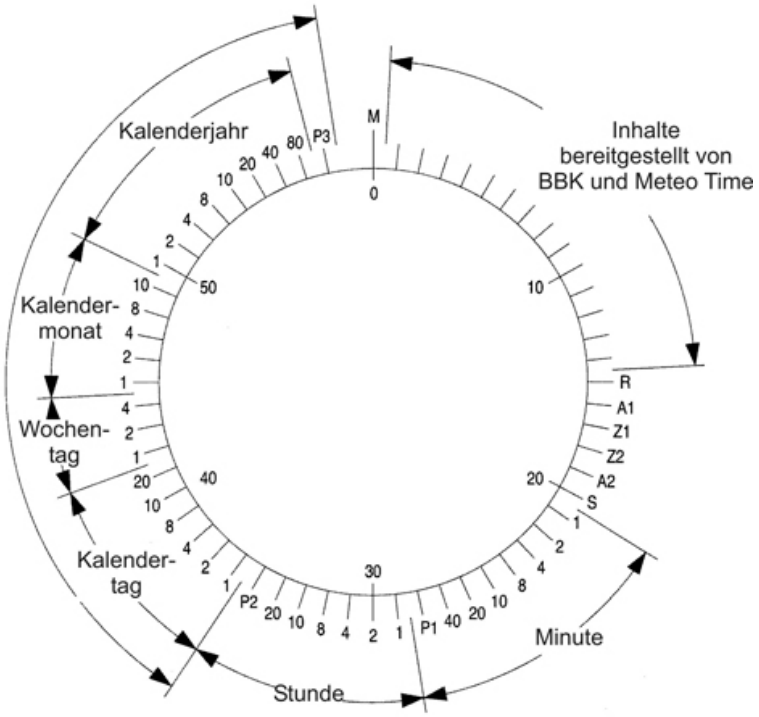

Abbildung 2: DFC77-Kodierung

Erläuterungen:

- Für die Aufgabe sind die Bits 20 bis 58 relevant, in denen die Zeitinformation für die jeweils nachfolgende Minute seriell in Form von BCD-Zahlen übertragen wird. Hierbei wird jeweils mit dem niederwertigsten Bit begonnen; die Wertigkeiten der Bits sind in der obigen Tabelle angegeben.
- Zur Absicherung der Daten werden Paritätsbits verwendet, hierbei handelt es sich um eine gerade Parität, d.h. die Summe der Werte der Datenbits und dem Wert des Paritätsbits muss 0 ergeben (modulo 2).
- Die Kodierung des Wochentages erfolgt gemäß der Norm ISO 8601 bzw. DIN EN 28601 (Montag = 1, Sonntag = 7)

Mikrocontroller Atmel ATmega8515

DFC77-Decoder sind mittlerweile in vielen Funkuhren verbaut, die mit sehr leistungsarmen, dafür kostengünstigen und energiesparsamen (Batteriebetrieb!) Controllern ausgerüstet sind.

Daher soll im Zuge dieser Aufgabe das DFC77-Signals mit Hilfe des *Atmel ATmega8515* 8-Bit-Mikrocontrollers (siehe Tabelle 2) erfolgen, der auf der Testplatine *STK500* verbaut ist.

Taktfrequenz	8 MHz (16 MHz mit externem Quarz möglich)
Flash (Programmspeicher)	8 kB
SRAM (Abreitsspeicher)	512 Bytes
EEPROM (Konstantenspeicher)	512 Bytes

Tabelle 2: Eigenschaften des *ATmega8515*

Des weiteren hat dieser Mikrocontroller unter anderem die folgenden Peripherieeinheiten, die für die Lösung der Aufgabe nützlich sind:

- 5 Digitale 8-Bit-I/O-Ports
  - Port `A`, ..., `E`
  - Datenrichtungs-Register `DDRA`, ...`DDRE`
  - Schreibender Zugriff über die Register: `PORTA`, ..., `PORTE`
  - Lesender Zugriff über die Register: `PINA`, ..., `PINE`
  - Einige der I/O-Pins haben können auch Sonderfunktionen wahrnehmen!
- Je ein 8- und ein 16-Zähler/-Timer mit Interruptfunktionalität
- Ein UART-Modul zur Ansteuerung der seriellen Schnittstelle (RS232)

Weitere Funktionen, die zwar für die Bearbeitung der Aufgabe nicht von Bedeutung sind, aber die Fähigkeiten des Controllers vor Augen führen:

- PWM-Generatoren zur Erzeugung von Pulsweiten-modulierten Signalen
- Analog-Komparator-Einheit
- Unterstützung von Hardware-SPI
- Watchdog-Timer (WDT)
- Bootloader-Funktionalität

Weiterführende Informationen sind in den Dokumenten `atmega8515.pdf` und `stk500.pdf` zu finden.

Mikrocontroller-Programmierung mit AVR-Studio und WinAVR

Einführung

Die Mikrocontoller der ATmega-Serie (sowie viele weitere Modelle der Firma Atmel) lassen sich komfortabel mit *avr-gcc* (unter Windows: *WinAVR*) in *C* programmieren. Mit der zusammen mit dem Compiler ausgelieferten *avr-libc* steht eine auf Atmel-Mikrocontroller angepasste Variante der C-Standardbibliothek zur Verfügung.

Übersicht über die wichtigsten Mikrocontroller-spezifischen Kommandos:

`DDR{A, ..., E} = 0xff;`	Datenrichtungsregister für I/O-Ports `A`, ..., `E`) Bit gesetzt: Pin ist Ausgang, Bit nicht gesetzt: Pin ist Eingang
`PORT{A, ..., E} = 0xf0;`	Schreiben von I/O-Ports (teilweise mit Spezialfunktionen)
`uint8_t in = PIN{A, ..., E};`	Lesen von I/O-Ports
`cli();`	Sperren von Interrupts. Wichtig für atomare Operationen.
`sei();`	Einschalten der Interrupts.
`volatile bool flag;`	Variablen, die sowohl aus dem Kontext des Hauptprogrammes (`main()`) als auch in einem Interrupt-Handler verwendet werden, müssen `volatile` deklariert werden.

Inbetriebnahme

Um Ihren Mikrocontroller in Betrieb zu nehmen, gehen Sie bitte wie folgt vor:

1. Um Beschädigungen zu vermeiden, muss die Testplatine **vor** dem Auf- oder Abstecken von Kabeln jeglicher Art ausgeschaltet werden.
2. Programmierverbindung: Verbinden Sie die serielle Schnittstelle des Host-PCs mit der Schnittstelle `RS232 CTRL` des *STK500* sowie die Programmierschnittstelle `ISP6PIN` mit `SPROG3` (6-adriges Kabel).
3. Serielle Schnittstelle: Verbinden Sie die Pins `PD0` / `PD1` mit Hilfe eines zwei-adrigen Kabels mit den Pins `RXD` / `TXD RS232 SPARE`
4. LEDs: Verbinden Sie `PORTB` mit `LEDS` (10-poliges Kabel)
5. Switches: Verbinden Sie `PORTA` mit `SWITCHES` (10-poliges Kabel)
6. Schließen Sie das DFC77-Signalkabel (gelb) am Pin `PD6` und die Masse (braun) am zugehörigen Pin `GND` an.

Erstellung eines Projektes

Um den Mikrocontroller testweise zu programmieren sind die folgenden Schritte nötig:

1. Starten Sie das *AVR Studio* und legen Sie ein neues *AVR-GCC*-Projekt an.
   - Lassen ein Verzeichnis für das Projekt erzeugen, allerdings keine initiale Datei.
   - Wählen Sie *AVR Simulator* als *Debug Platform* und *ATmega8515* als *Device*.
   - Kopieren Sie das Rahmenprogramm sowie die vorgegebenen Dateien aus dem Verzeichnis `src` in das eben erstellte Verzeichnis.
   - Fügen Sie die Kopien der Dateien zum Projekt hinzu.
   - Passen Sie die Projektoptionen an (Rechtklick, *Edit Configuration Options...*):
     - *General* → *Frequency*: 8000000
     - *Include directories*: Wurzelverzeichnis ihres Projektes
2. Programmieren Sie den Mikrocontroller
   - *Tools* → *Program AVR...* → *Auto connect*
   - Unter *Main* muss *ATmega8515* als Gerät sowie *ISP mode* ausgewählt werden. Frequenz: 1,843 MHz (schnellere Programmierung).
   - Unter *Program* → *Flash* können Sie nun das erstellte *Input HEX file* anwählen und per *Program* auf den Controller übertragen.

- **Achtung:** Modifizieren Sie unter keinen Umständen die Einstellungen in den anderen Tabs, insbesondere die der Tabs *Fuses* und *LockBits*. Sie können den Mikrocontroller sehr leicht unwiderruflich sperren und dadurch unbrauchbar machen!

3. Stecken Sie das serielle Kabel von `RS232 CTRL` auf `RS232 SPARE` um (*STK500* vorher ausschalten!).
4. Öffnen Sie *Start* → *Programs* → *Accessories* → *Communications* → *HyperTerminal* und erstellen Sie eine Verbindung (Beliebiger Verbindungsname, 9600 Bit pro Sekunde, 8 Daten-Bits, 1 Stop-Bit, keine Parität, Flusskontrolle aus). Der Mikrocontroller ist vorprogrammiert, eine Testausgabe auf der seriellen Schnittstelle auszugeben.

Aufgabenstellung

a) Nehmen Sie den Mikrocontroller in Betrieb und testen Sie die serielle Schnittstelle mit Hilfe des Testprogramms.

b) Erweitern Sie das Testprogramm so, dass beim Drücken des ersten Tasters (`SW0`) `LED0` aufleuchtet.

c) Erweiteren Sie das Programm nochmals, so dass `LED1` im Sekundentakt aufleuchtet (`LED0` soll weiterhin durch `SW0` ansteuerbar sein).

Hinweis: Der Timer wird vom vorgegebenen Rahmenprogramm so initalisiert, dass alle $10\,ms$ ein Interrupt ausgelöst wird.

d) Erstellen Sie nun ein Programm, das das DFC77-Signal auswertet.

- Verbinden Sie hierzu den Ausgang des DFC77-Empfängers mit einem unbelegten Eingangs-Pin und verwenden Sie den Timer, um das Signal abzutasten.
- Die Auswertung muss gewisse Unregelmäßigkeiten tolerieren, die sich aus Emfangsfehlern und dem Versatz des lokalen Timers auf dem Mikrocontoller und den High- und Low-Zeiten des Signals ergeben.
- Speichern Sie die ermittelte Bitsequenz vor der Auswertung in einem Array zwischen, und führen sie eine Paritätsprüfung durch.
- Geben Sie bereits während des Empfangs das jeweils empfangene Bit sowie nach dem Erhalt der vollständigen Information Uhrzeit (Format: `hh::mm::ss`) und Datum (Format: `<Wochentag>, tt.mm.yyyy`) auf der seriellen Schnittstelle aus.
- Erweitern Sie das Programm so, dass nach der Ermittlung der aktuellen Uhrzeit im Sekundentakt die jeweils aktuelle Uhrzeit ausgegeben wird. Nach dem Ablauf einer vollen Minute soll sich das Programm wieder neu über die empfangene Bitsequenz synchronisieren.