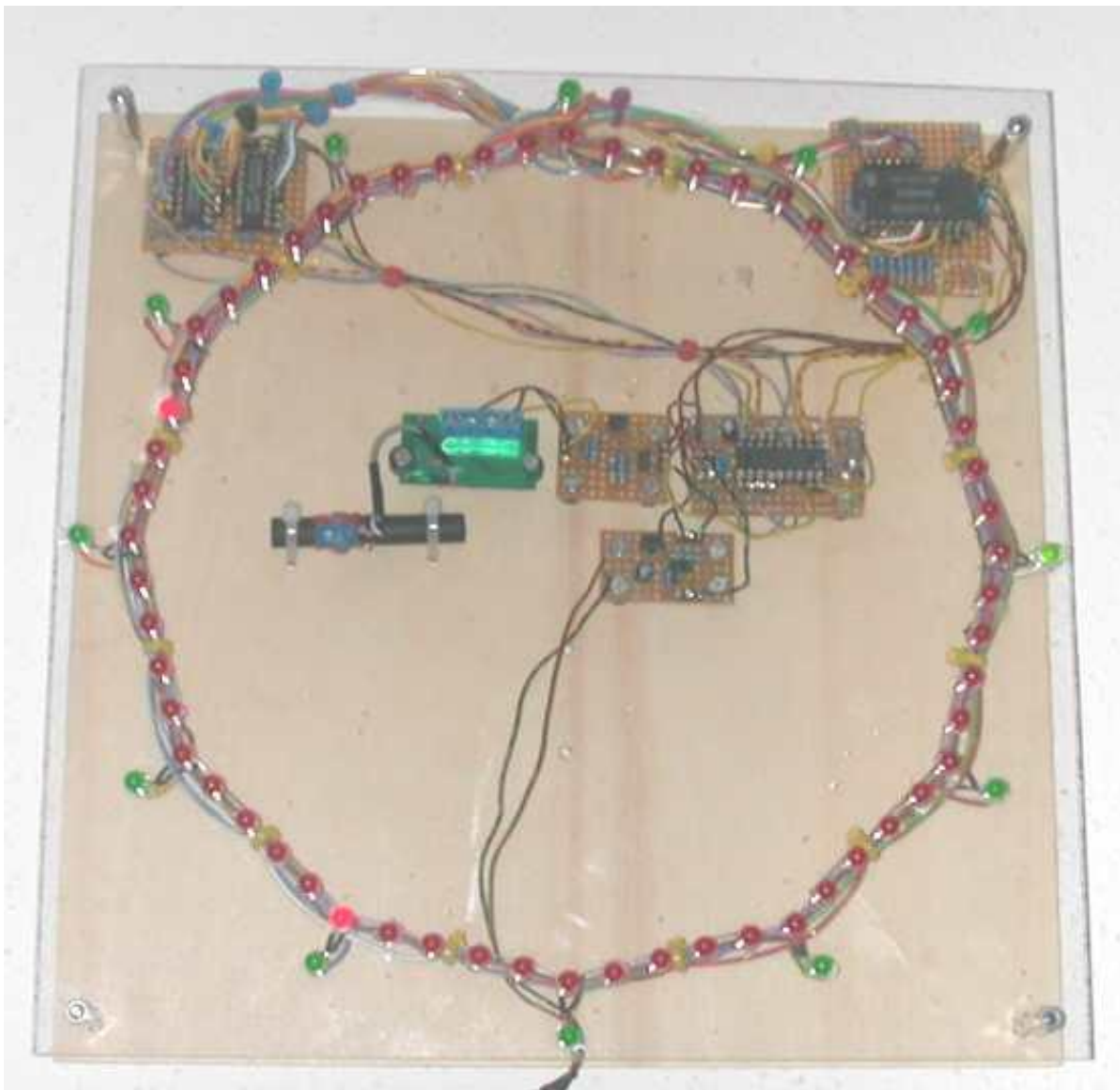


# Analog/Digitale Wanduhr 2



Autor:  
Letzte Bearbeitung:

Buchgeher Stefan  
15. Oktober 2006

# Inhaltsverzeichnis

<b>1. EINLEITUNG .....</b>	<b>3</b>
<b>2. GRUNDLEGENDES ZU DCF .....</b>	<b>4</b>
<b>3. SCHALTUNGSBESCHREIBUNG .....</b>	<b>6</b>
<b>4. SOFTWAREBESCHREIBUNG .....</b>	<b>10</b>
4.1. Das DCF-Dekodierverfahren .....	10
4.2. Benötigte Register, Konstanten, Portdefinition, Makros und Tabellen.....	11
4.3. Hauptprogramm.....	15
4.4. ISR (Timer 0).....	16
4.5. Unterprogramme.....	18
4.5.1. INIT .....	18
4.5.2. DCFROUTINE.....	19
4.5.3. DCFUPSEKUNDE.....	23
4.5.4. DCFUPMINUTE.....	24
4.5.5. INNEREUHR.....	28
4.5.6. ANZEIGE .....	29
4.5.7. BCDBIN2.....	29
<b>5. NACHBAUHINWEISE .....</b>	<b>31</b>
<b>ANHANG A: SCHALTPLAN .....</b>	<b>40</b>
<b>ANHANG B: LISTING DES PIC-MIKROCONTROLLER (WANDUHR2.ASM).....</b>	<b>41</b>
<b>ANHANG C: STÜCKLISTE.....</b>	<b>54</b>

# **1. Einleitung**

Eine Wanduhr ist heutzutage wohl nichts mehr Besonderes. Und eine rein elektronische Wanduhr im Selbstbau? Dazu kommt, dass diese Wanduhr dank DCF stets die korrekte Zeit anzeigt. Manchen ist der Begriff „Funkuhr“ geläufiger, denn genau darum handelt es sich bei diesem Projekt. Also um eine Funkuhr im Selbstbau. Ein angenehmer Nebeneffekt ist, dass dadurch das lästige Zeitumstellen zweimal im Jahr entfällt.

Die Zeiger werden hier durch Leuchtdioden ersetzt. Die Farben der Leuchtdioden spielen hier aber keine Rolle. Für die Anzeige der Stunde sollte allerdings nicht dieselbe Farbe wie für die Minuten/Sekunden verwendet werden. Es können prinzipiell natürlich auch blaue und/oder weiße Leuchtdioden verwendet werden. Es ist aber zu beachten, dass diese um einiges teurer sind als rote, gelbe oder grüne. Weiters sollten Leuchtdioden verwendet werden, die möglichst wenig Strom benötigen.

Es steht aber nicht der praktische Nutzen, sondern die Programmierung und der Umgang mit Mikrocontrollern im Vordergrund dieses Projektes! Hier geht es hauptsächlich um die Auswertung (Dekodierung) des DCF-Signals. Dies ist auch der Grund weshalb ich alle erarbeiteten Unterlagen (Schaltplan und Listing) zur freien Verfügung bereitstelle.

Da keine besonderen Forderungen an den Mikrocontroller gestellt werden, könnte diese Aufgabe wohl auch von jeder anderen Mikrocontrollerfamilie durchgeführt werden. Ich habe mich aber für die PIC-Familie entschieden.

Ich hoffe, dass dieses Projekt ausreichend dokumentiert ist, so dass auch ein nicht Elektronik-Profi mit diesem Projekt zurecht kommt.

Buchgeher Stefan

## 2. Grundlegendes zu DCF

Der Zeitzeichensender DCF77 befindet sich in Mainflingen, ca. 25 km südöstlich von Frankfurt am Main. Dieser Langwellensender hat, bedingt durch die niedrige Trägerfrequenz von 77,5 kHz eine Reichweite von ca. 1500 km bis 2000 km. (Bild 2.1)



Bild 2.1: Reichweite des DCF77-Senders (Quelle: www.dcf77.com/)

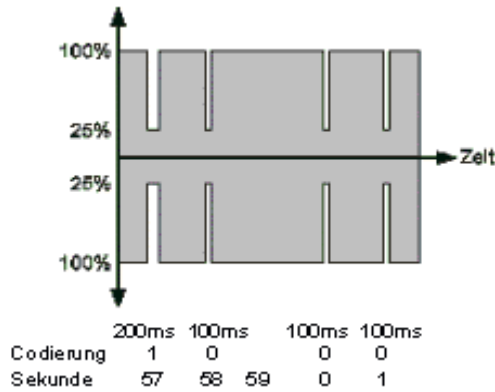


Bild 2.2: Amplitudenmodulation des 77,5kHz-Trägers

Der Dauerträger des DCF-Senders senkt im Sekundentakt für 100ms oder 200ms die Amplitude der Trägerfrequenz auf 25 % ab, was einer einfachen Amplitudenmodulation entspricht. Die Länge dieser so genannten Sekundenmarken überträgt in codierter Form das Zeittlegramm. Eine Absenkdauer des Trägers um 100ms (Toleranz:  $\pm 20\text{ms}$ ) entspricht dabei

einem logischen Low-Pegel, während ein logischer High-Pegel mit einer Absenkdauer von 200ms (Toleranz  $\pm 40\text{ms}$ ) codiert ist. In jeder 59.

Sekunde wird die Absenkung nicht vorgenommen, so dass damit eine eindeutige Zuordnung des Minutenanfangs möglich ist. Die neue Sekunde beginnt, mit Ausnahme der 59. Sekunde, jeweils mit dem Absenken des 77,5 kHz-Trägers.

Im jeweils einminütigem Zeittlegramm ist die Zeit (Stunde und Minute) der nächstfolgenden Minute sowie das komplette Datum und der jeweilige Wochentag codiert. Die folgende Tabelle zeigt die Bedeutung der 59 Bits, die pro Minute versandt werden.

Bit	Bedeutung	Wertigkeit	Bit	Bedeutung	Wertigkeit
0	Minutenbeginn Low		20	Telegrammbeginn High	
1	Reserve		21	Minuten Einer	1
2	Reserve		22	Minuten Einer	2
3	Reserve		23	Minuten Einer	4
4	Reserve		24	Minuten Einer	8
5	Reserve		25	Minuten Zehner	10
6	Reserve		26	Minuten Zehner	20
7	Reserve		27	Minuten Zehner	40
8	Reserve		28	Prüfbit 1	
9	Reserve		29	Stunden Einer	1
10	Reserve		30	Stunden Einer	2
11	Reserve		31	Stunden Einer	4
12	Reserve		32	Stunden Einer	8
13	Reserve		33	Stunden Zehner	10
14	Reserve		34	Stunden Zehner	20
15	Reserveantenne		35	Prüfbit 2	
16	Zeitumstellung Ankündigung		36	Kalendertag Einer	1
17	Zeitzonebit 1		37	Kalendertag Einer	2
18	Zeitzonebit 2		38	Kalendertag Einer	4
19	Schaltsekunde Ankündigung		39	Kalendertag Einer	8

Tabelle 2.1: Zeittlegramm

Bit	Bedeutung	Wertigkeit	Bit	Bedeutung	Wertigkeit
40	Kalendertag Zehner	10	50	Jahr Einer	1
41	Kalendertag Zehner	20	51	Jahr Einer	2
42	Wochentag	1	52	Jahr Einer	4
43	Wochentag	2	53	Jahr Einer	8
44	Wochentag	4	54	Jahr Zehner	10
45	Monat Einer	1	55	Jahr Zehner	20
46	Monat Einer	2	56	Jahr Zehner	40
47	Monat Einer	4	57	Jahr Zehner	80
48	Monat Einer	8	58	Prüfbit 3	
49	Monat Zehner	10	59	Keine Austastung	

Tabelle 2.1: Zeitlegramm (Fortsetzung)

Die Synchronisation des Sekundenzählers erfolgt mit Ausbleiben der Absenkung der 59. Sekunde. Die nächste Absenkung ist immer Low. Die Bits 1 bis 14 sind nicht belegt. Bit 15 zeigt durch einen High-Pegel an, dass zurzeit die Reserveantenne des DCF77-Senders aktiv ist. Im Normalfall ist dieses Bit auf „Low“ gesetzt. Bit 16 wird eine Stunde bevor die Zeitumstellung von Sommer- auf Winterzeit bzw. umgekehrt erfolgt auf „high“ gesetzt und mit der Zeitumstellung wieder zurückgesetzt. Die Zeitangaben beziehen sich auf die UTC-Zeit (**U**niversal **T**ime **C**oordinated). Bezogen auf die UTC-Zeit eilt die mitteleuropäische Zeit (MEZ) um eine Stunde vor, während die mitteleuropäische Sommerzeit (MESZ) um 2 Stunden voreilt. Diese Differenz wird in den Zeitzonenbits 17 und 18 ausgedrückt. Während der MEZ ist Bit 17 High und Bit 18 Low, während bei der Sommerzeit (MESZ) der Abstand zur UTC 2 Stunden beträgt und somit Bit 17 Low und Bit 18 High ist. Bit 19 kündigt eine bevorstehende Schaltsekunde an. Das eigentliche Zeit- und Datumstelegramm ist in den Bits 20 bis 58 codiert. Für die Einerstellen der Zeit und Datuminformationen sind jeweils 4 Bit, während für die Zehnerstellen nur 2 oder 3 Bit erforderlich sind. Die Zahlendarstellung der Zeit- und Datuminformation erfolgt im Binärformat (BCD-Code). Für die Jahreszahl werden nur die Einer- und Zehnerstelle übertragen. Die Bits 42 bis 44 geben in binärer Schreibweise den Wochentag an (der Wert 1 steht für den Montag, der Wert 7 für den Sonntag). Das Prüfbit 1 ergänzt die Bits 21 bis 27 auf gerade Parität, d.h. es werden die High-Bits 21 bis einschließlich 28 addiert, deren Ergebnis muss dann eine gerade Zahl ergeben. Das Prüfbit 2 ergänzt die Parität von Bit 29 bis 34, während Prüfbit 3 für die Parität der Bits 36 bis 57 zuständig ist. Diese Prüfbits sind ein erstes Überprüfungs-kriterium für ein DCF-Empfangsprogramm, welches damit zunächst auf einfache Weise die Konformität der empfangenen Daten überprüfen kann. Für eine fehlerfreie DCF-Decodierung sind allerdings noch weitere Maßnahmen notwendig. (Zum Beispiel ein Vergleich der soeben dekodierten Uhrzeit und des Datums mit der Uhrzeit und dem Datum welches mit der vorhergehenden Minute übertragen wurde und zusätzlich noch eine mitlaufende Softwareuhr).

In unregelmäßigen Zeitabständen muss eine Schaltsekunde eingefügt werden. Dies ist dadurch bedingt, dass sich die Erde nicht genau in 24 Stunden um sich selbst dreht. Auf die koordinierte Weltzeitskala UTC bezogen, wird diese Korrektur zum Ende der letzten Stunde des 31. Dezember oder 30. Juni vorgenommen. In Mitteleuropa muss die Schaltsekunde daher am 1. Januar um 1.00 Uhr MEZ oder am 1. Juli um 2.00 MESZ eingeschoben werden. Zu den genannten Zeiten werden daher 61 Sekunden gesendet.

### **3. Schaltungsbeschreibung**

Die gesamte Schaltung lässt sich grob in 5 Bereiche unterteilen:

- Mikrocontroller
- Anpassschaltung für das DCF-Empfangsmodul
- Stundenanzeige
- Minuten/Sekundenanzeige
- Stromversorgung

Die Schaltungsbeschreibung erfolgt hier entsprechend diesen 5 Bereichen. Anhang A zeigt den Schaltplan der gesamten Schaltung.

#### **Mikrocontroller:**

Der Mikrocontroller (IC1) zählt zu den wichtigsten Komponenten in diesem Projekt. Für diese Aufgabe wurde der Typ PIC16F84 der Fa. Microchip ausgewählt. Dieser verfügt über eine für dieses Projekt ausreichende Anzahl an I/O-Pins, über ausreichend Programmspeicher, Datenspeicher und auch über einen nichtflüchtigen Speicher<sup>1</sup>. Der größte Vorteil an diesem Mikrocontroller ist aber, dass er einen Flash-Programmspeicher besitzt und daher fast beliebig oft einfach neu programmiert werden kann.

#### Eigenschaften des Mikrocontroller PIC16F84:

- |                           |                  |
|---------------------------|------------------|
| • Flash-Programmspeicher: | 1k x 14-bit-Wort |
| • Datenspeicher:          | 68 Bytes         |
| • EEPROM-Datenspeicher:   | 64 Bytes         |
| • I/O-Ports:              | 13               |
| • Interruptquellen:       | 4                |
| • Stapelspeicher:         | 8 Ebenen         |
| • Timer:                  | 1                |
| • Taktfrequenz:           | 0 bis 20 MHz     |

#### Weitere Eigenschaften des PIC16F84:

- Direkte, indirekte und relative Adressierung
- Power-on Reset (POR)
- Power-up-Timer (PWRT)
- Interner Watchdog (WDT)
- In-Circuit Serial Programming™ (ICSP)
- Geringer Stromverbrauch

Für die Wanduhr wird jedoch nur ein Teil dieser Eigenschaften benötigt.

#### Grundbeschaltung des Mikrocontrollers:

Als Takterzeugung wird eine Standardapplikation bestehend aus einem Quarz (X1), zwei Keramikkondensatoren (C2, C3) und einem Widerstand (R3) verwendet. Bei X1 handelt es sich um einen 4,096 MHz-Quarz. Dieser recht „ungewöhnliche“ Wert ist notwendig, damit mit dem Controller eine genaue Uhr realisiert werden kann, welche auch ohne DCF-Synchronisierung oder bei einem gestörten DCF-Empfang eine genaue Zeit erzeugt.

---

<sup>1</sup> Unter einem nichtflüchtigen Speicher versteht man einen Speicher, welcher seine Werte beim Ausschalten des Gerätes beibehält. Diese Speicherart wird hauptsächlich für Geräteinstellungen verwendet

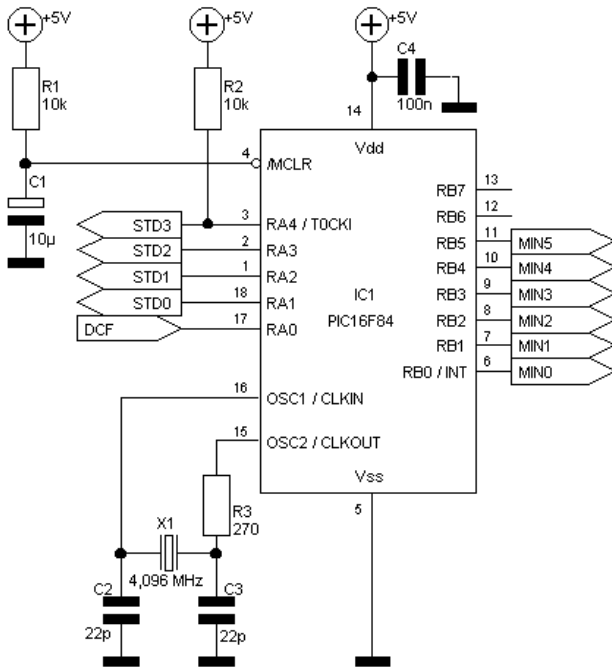


Bild 3.1: Beschaltung des Mikrocontroller

Zur Erzeugung des Reset wurde ebenfalls eine einfache Standardlösung bestehend aus einem Widerstand (R1) und einem Elektrolyt-Kondensator (C1) gewählt.

Der Kondensator C4 dient zur Entkoppelung der Betriebsspannung für den Mikrocontroller. Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser muss möglichst nahe an diesen IC angebracht werden.

Eine Besonderheit stellt der Portpin RA4 dar. Im Gegensatz zu den anderen Portpins besitzt dieser einen so genannten Open-Drain-Ausgang. Wird dieser Portpin, so wie hier bei der Wanduhr, als Ausgang verwendet, so ist ein Pull-Up-Widerstand (R2) notwendig.

Wanduhrspezifische Beschaltung:

Das DCF-Empfangsmodul ist mit einer zusätzlichen Anpassschaltung am Portpin RA0 angeschlossen. Die restlichen 4 Pins des Port A dienen der Stundenanzeige, während die Minuten-/Sekundenanzeige über den Port B (RB0 bis RB5) angesteuert werden. Die beiden übrig bleibenden Portpins RB6 und RB7 sind für mögliche Erweiterungen vorgesehen.

**Anpasschaltung für das DCF-Empfangsmodul:**



Eine weitere wichtige Komponente zur DCF-Dekodierung ist ein bei Conrad erhältliches DCF-Empfangsmodul (Bestell-Nr.: 641138). Dieses Modul enthält eine Empfangsantenne und einen Demodulator, so dass am Ausgang des Moduls das übertragene Zeitlegramm mit einem Mikrocontroller ausgewertet werden kann.

Bild 3.2: DCF-Empfangsmodul

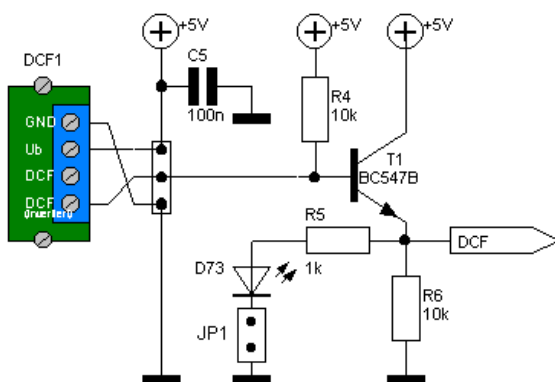


Bild 3.3: Anpassschaltung

Der Ausgangsstrom von nur einem Milliampere ist für die Kontroll-LED (D73) zu wenig. Die nachgeschaltete Transistorstufe (T1) mit den Widerständen R4 und R6 gleicht diesen Nachteil aus. Der Widerstand R5 dient als Vorwiderstand für die Leuchtdiode (D73). Diese Leuchtdiode signalisiert den empfangenen Datenstrom, wenn der Jumper JP1 gesteckt ist. Bei einem korrekten Empfang blinkt diese Leuchtdiode

sagen den Ausgangspegel der Anpassschaltung an. Das Puls-Pausen-Verhältnis (Leuchtzeit/Dunkelzeit der Leucht-diode D73) entspricht der im Abschnitt 2 genannten Zeiten. Bei etwas Übung kann der Unterschied zwischen Low (100ms Absenkung des Trägers, LED ist 100ms dunkel) und High (200ms Absenkung des Trägers, LED ist 200ms dunkel) optisch erkannt werden. Dieser Jumper sollte allerdings nicht dauerhaft gesteckt werden, da dadurch die Spannungsquelle (z.B. ein Steckernetzteil) unnötig belastet wird.

Der Kondensator C5 dient zur Entkoppelung der Betriebsspannung für das DCF-Modul. Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser muss möglichst nahe am DCF-Modul angebracht werden.

### Stundenanzeige:

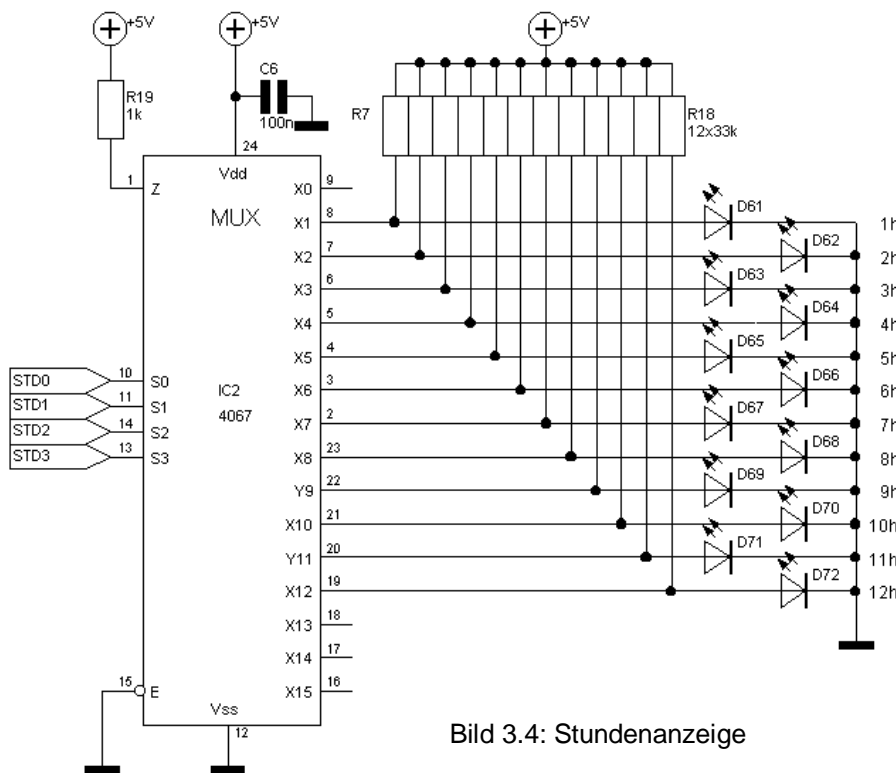


Bild 3.4: Stundenanzeige

Für die Anzeige der aktuellen Stunde dienen 12 Low-current Leuchtdioden (D61-D72). Diese 12 Leuchtdioden besitzen je einen relativ hochohmigen Vorwiderstand (R7 bis R18). Diese Vorwiderstände bewirken dass diese 12 LEDs sehr schwach leuchten, was vor allem in der Nacht die Orientierung wesentlich vereinfacht. Die aktuelle Stunde soll aber stärker leuchten, so dass man die aktuelle Stunde gut ablesen kann. Zu diesem Zweck dient

ein einfacher 1-aus-16-Dekoder (IC2). Hier wird der Typ 4067 verwendet, wobei nur die Ausgänge 1 bis 12 verwendet werden. Während der Synchronisierung wird auf den Ausgang 0 (Pin 9) geschaltet, so dass keine der 12 Leuchtdioden leuchtet. Nur bei der gültigen Uhrzeit leuchtet die entsprechende Leuchtdiode. Die Funktion des 1-aus-16-Dekoders (IC2) ist sehr einfach. Der binäre Wert der 4 Eingänge S0 bis S3 bestimmt, welcher Ausgang X0 bis X15 mit dem Eingang Z (Pin 1) verbunden wird. Als Folge wird der Widerstand R19 dem, an diesem Ausgang angeschlossenen Vorwiderstand (R7 bis R18) parallelgeschaltet. Die dazugehörige Leuchtdiode leuchtet dadurch stärker, da sich nun der Vorwiderstand aus einer Parallelschaltung zwischen dem 33-k-Vorwiderstand und R19 (1k) bildet. Bekanntlich ergibt sich daraus ein Wert, der in etwa R19 (1k) entspricht.

Der Kondensator C6 dient zur Entkoppelung der Betriebsspannung für IC2. Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser sollte möglichst nahe an IC2 angebracht werden



### Minuten-/Sekunden-Anzeige:

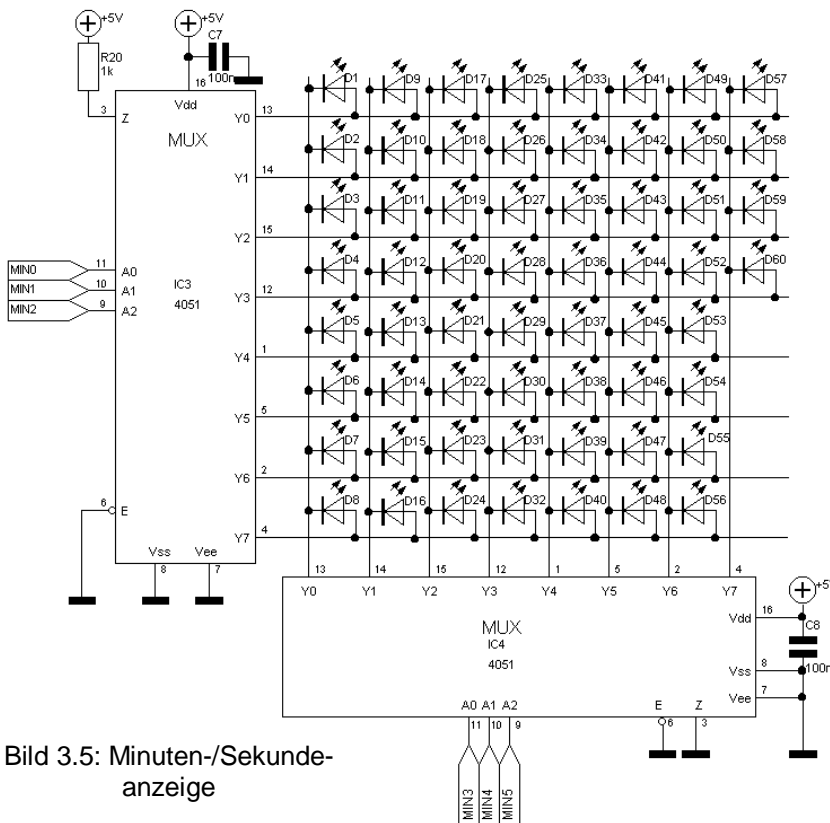


Bild 3.5: Minuten-/Sekunden-anzeige

Die 60 Low-current-Leuchtdioden (D1 bis D60) für die Anzeige der Minute und der Sekunde sind in Form einer 8 x 8 – Matrix angeordnet. Welche Leuchtdiode leuchten soll wird durch die beiden 1-aus-8-De-koder (IC3 und IC4) bestimmt. Der binäre Wert der 3 Eingänge A0 bis A2 bestimmt, welcher Ausgang Y0 bis Y7 mit dem Eingang Z (Pin 3) verbunden wird. Demnach schaltet IC3 den Widerstand R20 an den entsprechend ausgewählten Ausgang, während IC4 das

Massepotential an den ausgewählten Ausgang legt. Als Folge wird eine der 60 Leuchtdioden über den „Vorwiderstand“ R20 auf Betriebsspannung gelegt und erhält auch gleichzeitig ein Massepotential und leuchtet, während die anderen Leuchtdioden dunkel bleiben.

Die beiden Kondensatoren C7 und C8 dienen zur Entkoppelung der Betriebsspannung für IC3 und IC4. Für diese Koppelkondensatoren sollten Keramiktypen verwendet werden. Diese sollten möglichst nahe an diese ICs angebracht werden.

### Stromversorgung:

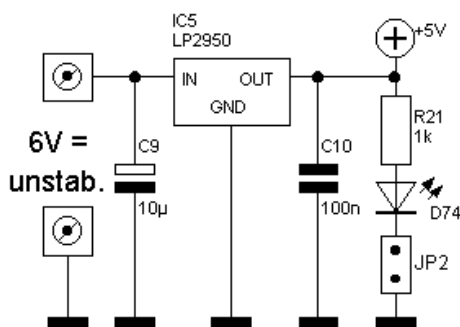


Bild 3.6: Stromversorgung

Zur Stromversorgung gibt es nicht viel zu sagen. Ein Low-Drop-Festspannungsregler (IC5) vom Typ LP2950 übernimmt mit den Kondensatoren C9 und C10 die Spannungsregelung. Als Spannungsquelle dient beispielsweise ein 6-V-Steckernetzteil.

Bei gestecktem Jumper JP2 dient die Leuchtdiode D74 zur Spannungskontrolle. Dieser Jumper sollte allerdings nicht dauerhaft gesteckt werden, da dadurch die Spannungsquelle unnötig zusätzlich belastet wird.

## **4. Softwarebeschreibung**

Die Aufgabe der PIC-Software besteht bei der Wanduhr hauptsächlich aus der DCF-Dekodierung. Diese wiederum besteht aus mehreren Teilaufgaben:

- In kurzen Zeitabständen (ca. alle 4ms) das vom DCF-Empfangsmodul erzeugte Signal abtasten. Daraus entweder ein Low, High oder eine neue Minute ermitteln. Kann aus den Abtastungen keines dieser drei Fälle ermittelt werden, so handelt es sich um einen Telegrammfehler.
- Die empfangenen Low- und High-Pegel zwischenspeichern
- Bei jeder neuen Minute aus den gespeicherten Werten entsprechend Tabelle 1 die Zeit- und Datumswerte zusammensetzen und in entsprechenden Registern sichern.
- Das soeben ermittelte Telegramm mit dem Telegramm der vorhergehenden Minute vergleichen. Die soeben ermittelte Minute muss dabei um 1 größer als die vorhergehende Minute sein, und die soeben ermittelten Werte für die Stunde, Tag, Monat, Jahr und Wochentag muss mit den Werten des vorhergehenden Telegramms übereinstimmen. ACHTUNG: Bei dieser einfachen Überprüfung wird der Übergang von z. B. 13:59 auf 14:00 als falsch gewertet, da sich hier die Stunde ändert, und auch die Minute um mehr als 1. Würden hier alle möglichen Übergänge berücksichtigt, dann würde dieses Unterprogramm noch umfangreicher als es ohnehin schon ist. Durch diese Vereinfachung dauert die Synchronisierung im schlimmsten Fall nur eine Minute länger.
- Parallel zur DCF-Uhr eine reine Softwareuhr erzeugen. Diese Softwareuhr ist gültig, wenn keine korrekte Zeit vom DCF-Empfangsmodul empfangen wird. Mit anderen Worten. Die Softwareuhr wird ständig mit einer gültig dekodierten „DCF-Zeit“ synchronisiert. Dies gilt natürlich auch für die Datumsinformationen.

Die zweite wichtige Aufgabe der PIC-Software ist die Anzeige der Uhrzeit. Für die Anzeige dient das gleichnamige Unterprogramm (ANZEIGE). Dieses Unterprogramm wird ebenfalls zyklisch alle 4ms aufgerufen. Dazu wird die gleiche 4-ms-Zeitbasis wie für die DCF-Dekodierung verwendet. Der Grund für diese Zeitbasis ist, dass das Unterprogramm ANZEIGE abwechselnd die Minute und die Sekunde ausgibt. (zur Erinnerung: Die Minute und die Sekunde werden ja auf einem gemeinsamen „Leuchtdioden-Kreis“ angezeigt). Man spricht hier also von einer einfachen Multiplexansteuerung. Für den Betrachter entsteht so der Eindruck, dass gleichzeitig die Sekunde und die Minute leuchten. Dazu eignet sich die 4-ms-Zeitbasis, und diese ist ja ohnehin schon vorhanden.

Für die Stundenanzeige stehen 12 eigene Leuchtdioden, welche über den Port A angesteuert werden, zur Verfügung. Daher ist hier keine Multiplexansteuerung notwendig.

### **4.1. Das DCF-Dekodierverfahren**

Der PIC-I/O-Eingang, an dem das DCF-Empfangsmodul (mit einer Anpassschaltung) angeschlossen ist (hier bei der Wanduhr der Portpin RA0), wird zyklisch (ca. alle 4 ms) vom Unterprogramm DCFROUTINE abgefragt. Als Zeitbasis für den 4-ms-Takt dient der Timer-0-Interrupt. Der Aufruf des Unterprogramms DCFROUTINE erfolgt hier aber nicht von der ISR (Interrupt-Service-Routine), sondern vom Hauptprogramm. Die ISR setzt nur alle 4 ms ein Flag zur Kennzeichnung, dass das Hauptprogramm das Unterprogramm DCFROUTINE aufrufen soll.

Die Aufgabe des Unterprogramms DCFROUTINE besteht darin, aus dem vom DCF-Empfangsmodul empfangenen Datenstrom die Informationen LOW, HIGH und den Minutenwechsel (also das Ausbleiben der 59. Sekunde) gemäß dem im Abschnitt 2 (Grundlegendes zu DCF) gewonnenen Erkenntnisse zu dekodieren. Für diese Informationen befinden sich im DCF-Statusregister (DCFSTATUS) entsprechende Flags. (Siehe Abschnitt 4.2. Benötigte Register, Konstanten, Portdefinition, Makros und Tabellen.)

Weiters beinhaltet das Register DCFSTATUS noch den Zustand des DCF-Eingangs 4 ms vor dem Aufruf des Unterprogramms DCFROUTINE. Mit Hilfe des aktuellen Zustands und des Zustands vor 4 ms kann nun erkannt werden, ob sich der Pegel am DCF-Eingang geändert hat.

Das Hauptprogramm prüft nun ständig die Flags DCFNEUESEK (wird zusätzlich zu DCFLOW bzw. DCFHIGH gesetzt, wenn eine gültige Sekunde empfangen wurde) und DCFNEUEMIN, und ruft die zugehörigen Unterprogramme *DCFUPSEKUNDE* bzw. *DCFUPMINUTE* auf. Die Aufgabe des Unterprogramms DCFUPSEKUNDE ist es, die empfangenen Low- und High-Pegel in den Register DCFTELEGRAMM1 bis DCFTELEGRAMM8 zwischenspeichern.

Die Aufgabe des Unterprogramms DCFUPMINUTE ist dagegen etwas umfangreicher: Zunächst die Zeit und das Datum aus den in den Registern DCFTELEGRAMM1 bis DCFTELEGRAMM8 zwischengespeicherten Werten entsprechend Tabelle 1 zusammensetzen. Anschließend die soeben gewonnenen Zeit- und Datumsinformationen mit den Zeit- und Datumsinformationen der vorhergehenden Minute vergleichen. Die Zeit- und Datumswerte der soeben empfangenen Minute sind nur dann gültig, wenn sich die Minute um maximal 1 von der vorhergehenden Minute unterscheidet, während sich die restlichen Zeit- und Datumsinformationen mit denen der vorhergehenden Minute nicht unterscheiden. Dies ist nur eine einfache Überprüfung, die aber manche richtigen Telegramme als falsch auswertet. (Siehe Abschnitt 4.5.4)

Parallel zur Dekodierung des DCF-Eingangs erzeugt das Unterprogramm *INNEREUHR* eine reine Softwareuhr. Für diesen Zweck ist eine genaue 1-Sekunden-Zeitbasis notwendig. Diese Zeitbasis wird von der schon erwähnten Timer-0-ISR (Interrupt Service Routine) zusätzlich zur 4ms-Zeitbasis erzeugt. Auch für diese Zeitbasis wird in der ISR ein entsprechendes Flag gesetzt, und das Hauptprogramm ruft das Unterprogramm *INNEREUHR* auf, wenn dieses Flag gesetzt ist. Diese zusätzliche Softwareuhr mag auf dem ersten Blick unnötig erscheinen. Es hat sich aber gezeigt, dass ein DCF-Empfang oft auch über eine längere Zeit nicht möglich ist. In diesem Fall würde die Uhr „stehen“. Diese Zeit wird mit einer zusätzlichen Softwareuhr so überbrückt, dass der Betrachter der Uhr davon nichts bemerkt.

## 4.2. Benötigte Register, Konstanten, Portdefinition, Makros und Tabellen

### Register:

Für die Wanduhr sind neben einigen internen Register (SFR, Spezielle Funktions-Register) noch eine ganze Menge eigener Register notwendig, wobei der Großteil für die DCF-Dekodierung notwendig sind:

- *ISR\_STAT\_TEMP*: Zwischenspeicher des Statusregister der ISR
- *ISR\_w\_TEMP*: Zwischenspeicher des Arbeitsregister der ISR

- **FLAGISRHP:** beinhaltet Botschaftsflags ISR -> HP, hier bei der Wanduhr die Flags für die 4-ms-Zeitbasis (Flag FLAG4MSEK) und für 1-Sekunden-Zeitbasis (Flag FLAG1SEK)
- **ZAEHLERISR1SEK:** Zählregister für 1-Sekunden-Zeitbasis
- **DCFSTATUS:** Statusregister der DCF-Dekodierung. Dieses Register beinhaltet folgende Bits:

Bit	Flagname	Funktion
0	DCFPORTNEU	Akt. Zustand DCF-Porteingang
1	DCFPORTALT	Vorhergehender Zustand am DCF-Porteingang
2	DCFNEUESEK	gesetzt, wenn neue Sekunde begonnen
3	DCFNEUEMIN	gesetzt, wenn neue Minute begonnen
4	DCFLOW	gesetzt, wenn Low-Signal erkannt
5	DCFHIGH	gesetzt, wenn High-Signal erkannt
6	DCFFEHLER	gesetzt, wenn ein Fehler erkannt
7	DCFSYNC	gesetzt, wenn Uhr mit DCF synchronisiert

Tabelle 4.1: Zusammensetzung von DCFSTATUS

- **DCFPULS:** Zählregister zur Ermittlung der Pulsdauer bei der DCF-Dekodierung
- **DCFPAUSE:** Zählregister zur Ermittlung der Pausedauer bei der DCF-Dekodierung.
- **DCFTELEGRAMM1 – DCFTELEGRAMM8:** In diese Register wird das empfangene, neue DCF-Telegramm im Sekundentakt eingelesen. Beginnt eine neue Minute, so werden diese Register ausgewertet und in die entsprechenden Register kopiert, z.B. in das Register DCFMINBCD.
- **DCFMINBCD:** Beinhaltet die dekodierte, aktuelle Minute im BCD-Format.
- **DCFSTDBCD:** Beinhaltet die dekodierte, aktuelle Stunde im BCD-Format.
- **DCFTAGBCD:** Beinhaltet den dekodierten, aktuellen Tag im BCD-Format.
- **DCFMONBCD:** Beinhaltet den dekodierten, aktuellen Monat im BCD-Format.
- **DCFJAHRBCD:** Beinhaltet das dekodierte, aktuelle Jahr im BCD-Format, wobei nur die Einer- und die Zehnerstelle im DCF-Telegramm enthalten sind. Die Hunderter- und die Tausenderstelle befinden sich nicht im DCF-Telegramm
- **DCFWOCHENTAG:** Beinhaltet den dekodierten, aktuellen Wochentag, wobei folgende Tabelle gilt:

Wert	Bedeutung
1	Montag
2	Dienstag
3	Mittwoch
4	Donnerstag
5	Freitag
6	Samstag
7	Sonntag

Tabelle 4.2: Wochentag

- **DCFZUSATZINFOS:** Beinhaltet weitere, folgende Informationen, die im DCF-Telegramm enthalten sind:

Bit	Flagname	Funktion
0	DCFRESANT	Ist dieses Bit gesetzt so wurde die Reserveantenne zum Versenden des DCF-Telegramms verwendet
1	DCFSOMWIN	Ist dieses Bit gesetzt, so kündigt es eine Umstellung zwischen Sommer- und Winterzeit an. Dieses Bit wird eine Stunde vor der Zeitumstellung gesetzt. Ab der Zeitumstellung enthält es wieder den Wert 0
2	DCFSOMMER	Während der Sommerzeit ist dieses Bit gesetzt
3	DCFWINTER	Während der Winterzeit ist dieses Bit gesetzt
4	DCFSCHALTSEK	Ist dieses Bit gesetzt, so kündigt es eine Schaltsekunde an

Tabelle 4.3: Zusammensetzung von DCFZUSATZINFOS

- DCFMINALT: Beinhaltet die empfangen Minute des DCF-Telegramms, welches mit der vorangegangenen Minute empfangen wurde. Diese wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- DCFSTDALT: Beinhaltet die empfangen Stunde des DCF-Telegramms, welches mit der vorangegangenen Minute empfangen wurde. Diese wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- DCFTAGALT: Beinhaltet den empfangenen Tag des DCF-Telegramms, welcher mit der vorangegangenen Minute empfangen wurde. Dieser wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- DCFMONALT: Beinhaltet den empfangen Monat des DCF-Telegramms, welcher mit der vorangegangenen Minute empfangen wurde. Dieser wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- DCFJAHRALT: Beinhaltet das empfangen Jahr des DCF-Telegramms, welches mit der vorangegangenen Minute empfangen wurde. Dieser wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- DCFWTAGALT: Beinhaltet den empfangen Wochentag des DCF-Telegramms, welcher mit der vorangegangenen Minute empfangen wurde. Dieser wird zur Überprüfung des neuen DCF-Telegramms verwendet.
- UHRSEKUNDE: Sekundenzähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- UHRMINUTE: Minutenzähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- UHRSTUNDE: Stundenzähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- DATUMTAG: Tageszähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- DATUMMONAT: Monatszähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- DATUMJAHR: Jahreszähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- DATUMWTAG: Wochentagszähler der inneren quartzesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert)
- FLAGS: Beinhaltet ein Hilfsflag für die Anzeige
- TEMP1, TEMP2: Hilfsregister

**Konstanten:**

Die Konstante *KONSTISR1SEK* gibt die Anzahl der notwendigen ISR-Aufrufe für die 1-Sekunden Zeitbasis an.

Hier wird die ISR alle 4ms aufgerufen, daher ergibt sich für diese Konstanten der Wert 250 (250 x 4ms = 1000ms = 1 Sekunde)

Bei einem LOW dauert die Absenkung des Trägers zwischen 80ms und 120ms, für ein High dauert die Absenkung des Trägers zwischen 160ms und 240ms (siehe auch Abschnitt 2. Grundlegendes zu DCF). Die Konstanten *DCFLOWMIN*, *DCFLOWMAX*, *DCFHIGHMIN* und *DCFHIGHMAX* dienen zur Ermittlung eines LOW oder eines HIGH. Das Unterprogramm DCFROUTINE wird alle 4ms aufgerufen, daher ergeben sich für die Konstanten folgende Werte:

- DCFLOWMIN: 20 (20 x 4ms = 80ms)
- DCFLOWMAX: 30 (30 x 4ms = 120ms)
- DCFHIGHMIN: 40 (40 x 4ms = 160ms)
- DCFHIGHMAX: 60 (60 x 4ms = 240ms)

### Portdefinition:

Im Allgemeinen wird bei jeder Anwendung der Eingangspin für die DCF-Dekodierung an einem anderen Portpin verwendet. Damit dies in der Software nur an einer Stelle berücksichtigt werden muss befindet sich in der Software eine Portdefinition für den DCF-Eingang. Diese besteht aus den folgenden 3 Parametern:

- DCFINPORT: Dieser Parameter gibt den Port an (z.B. Port A, Port B,...).
- DCFINTRIS: Dieser Parameter ist für die Initialisierung des Verwendeten Ports zuständig. Für die DCF-Dekodierung muss der verwendete Portpin als Eingang definiert werden.
- DCFIN: Dieser Parameter gibt den Portpin des Verwendeten Ports an (z.B. 0, 1, 2, usw.).

Achtung: Wird für DCFINPORT der Port A verwendet, so muss für DCFINTRIS das zum Port A zugehörige TRIS-Register definiert werden. Bei der hier beschriebenen Wanduhr ergeben sich aufgrund der Hardwarebeschaltung folgende Definitionen

```
DCFINPORT equ PORTA
DCFINTRIS equ TRISA
DCFIN equ 0
```

### Makros:

```
;Umschalten zu Registerbank 0
bank0 MACRO
    bcf STAT,RP0
ENDM

;Umschalten zu Registerbank 1
bank1 MACRO
    bsf STAT,RP0
ENDM

;Bit invertieren (entspricht dem Befehl btg bei PIC17/18)
btg MACRO FILE,BIT
    local SETBIT
    btfss FILE,BIT
    goto SETBIT
    bcf FILE,BIT
    btfsc FILE,BIT
SETBIT bsf FILE,BIT
ENDM
```

**Tabellen:**

Die bei der Wanduhr verwendeten Tabellen dienen zur Anzeige der Uhrzeit:

- Die Tabelle TABSTUNDEN wandelt den Stundenwert so um, dass die Stunde am Port A entsprechend dem vorgegebenen Schaltungsteil korrekt angezeigt wird.

Hier die Tabelle:

```
TABSTUNDEN      addwf PC,f
dt              .24
dt              .2,.4,.6,.8,.10,.12,.14,.16,.18,.20,.22,.24
dt              .2,.4,.6,.8,.10,.12,.14,.16,.18,.20,.22
dt              .0,.0,.0,.0,.0,.0,.0,.0,.0,.0
```

- Die Tabelle TABMINUTENSEKUNDEN wandelt sowohl den Minuten- als auch den Sekundenwert so um, dass diese am Port B entsprechend dem vorgegebenen Schaltungsteil korrekt angezeigt werden.

Hier die Tabelle:

```
TABMINUTENSEKUNDEN
      addwf PC,f
dt              .59
dt              .0,.1,.2,.3,.4,.5,.6,.7,.8,.9,.10,.11,.12
dt              .13,.14,.15,.16,.17,.18,.19,.20,.21,.22,.23,.24
dt              .25,.26,.27,.28,.29,.30,.31,.32,.33,.34,.35,.36
dt              .37,.38,.39,.40,.41,.42,.43,.44,.45,.46,.47,.48
dt              .49,.50,.51,.52,.53,.54,.55,.56,.57,.58
dt              .60,.60,.60,.60,.60
```

## 4.3. Hauptprogramm

**Aufgaben des Hauptprogramms:**

Zuerst wird der Mikrocontroller initialisiert. Diese Tätigkeit wird von einem Unterprogramm (INIT) ausgeführt. Danach wird der Timer0-Interrupt und der globale Interrupt freigegeben.<sup>2</sup> Dafür ist das Register INTCON zuständig. Je nach benötigten Interrupts werden die entsprechenden Freigabebits (im Englischen: Enable) gesetzt. Wird ein Interrupt verwendet so muss zusätzlich zum verwendeten Interrupt auch die globale Interruptfreigabe GIE (**G**eneral **I**nterrupt **E**nable) gesetzt werden. Er ist sozusagen der Hauptschalter, der Interrupts ermöglicht. Das INTCON-Register kann einfach mit den Befehlen `movlw b'10100000'` und `movwf INTCON` beschrieben werden. Der Timer0-Interrupt ist jetzt eingeschaltet. Er sorgt hier bei der Wanduhr für eine 4-ms-Zeitbasis und für eine 1-Sekunden-Zeitbasis.

Nun befindet sich die Software in einer Endlosschleife. Diese Schleife besitzt die Aufgabe ständig die so genannten Botschaftsflags abzufragen. Ist eines dieser Botschaftsflags gesetzt, so muss vom Hauptprogramm eine bestimmte Aufgabe ausgeführt werden. Diese Aufgaben sind in Form von Unterprogrammen vorhanden. Zwei dieser Botschaftsflags werden in der Timer-ISR gesetzt. Und zwar wird alle 4ms ein Flag gesetzt. Das zweite Flag von der ISR wird jede Sekunde gesetzt.

<sup>2</sup> Der PIC16F84 besitzt 4 Interruptquellen. Bei der Wanduhr wird aber nur der Timer0-Interrupt benötigt.

Hier Zusammenfassend die Tätigkeiten in der Endlosschleife, welche durch die Botschaftsflags ausgelöst werden:

- Tätigkeiten und/oder Unterprogramme, die alle 4ms durchgeführt werden müssen
  - Unterprogramm DCFROUTINE aufrufen (siehe Abschnitt 4.5.2)
  - Unterprogramm ANZEIGE aufrufen (siehe Abschnitt 4.5.6)
- Tätigkeiten und/oder Unterprogramme, die jede Sekunde durchgeführt werden
  - Unterprogramm INNEREUHR aufrufen (siehe Abschnitt 4.5.5)
- Bei einem Minutenwechsel (Flag DCFNEUEMIN ist gesetzt) das Unterprogramm DCFUPMINUTE abarbeiten (siehe Abschnitt 4.5.4)
- Bei einem Sekundenwechsel (Flag DCFNEUESEK ist gesetzt) das Unterprogramm DCFUPSEKUNDE abarbeiten (siehe Abschnitt 4.5.3)

**Hier das Hauptprogramm:**

```

Beginn      call  INIT                ;Initialisierungen
            movlw b'10100000'      ;TMR0 freigeben durch
            movwf INTCON           ;Setzen von GIE und T0IE im Register
                                   ; INTCON

HPSCHLEIFE  btfsc  FLAGISRHP,FLAG4MSEK
            goto  HPWEITER1

            btfsc  FLAGISRHP,FLAG1SEK
            goto  HPWEITER2

            btfsc  DCFSTATUS,DCFNEUEMIN ;Wenn Anforderungsflag DCFNEUEMIN
                                   ; gesetzt
            call  DCFUPMINUTE        ;Unterprogramm DCFUPMINUTE ausführen
            btfsc  DCFSTATUS,DCFNEUESEK ;Wenn Anforderungsflag DCFNEUESEK
                                   ; gesetzt
            call  DCFUPSEKUNDE      ;Unterprogramm DCFUPSEKUNDE ausführen

            goto  HPSCHLEIFE

HPWEITER1   call  DCFROUTINE        ;Tätigkeiten, die alle 4 ms durchgeführt
                                   ; werden müssen

            call  ANZEIGE
            bcf   FLAGISRHP,FLAG4MSEK ;Anforderungsflag (für 4ms
                                   ; Aktivitäten) zurücksetzen

            goto  HPSCHLEIFE

HPWEITER2   call  INNEREUHR        ;Tätigkeiten, die jede Sekunde
                                   ; durchgeführt werden müssen
            bcf   FLAGISRHP,FLAG1SEK ;Anforderungsflag (für 1-Sek.-
                                   ; Aktivitäten) zurücksetzen

            goto  HPSCHLEIFE
  
```

## 4.4. ISR (Timer 0)

Eine ISR (Interrupt Service Routine) ist im Prinzip ein Unterprogramm, welches aber im Gegensatz zu normalen Unterprogrammen, „unvorhergesehen“ aufgerufen wird. Hier, beim Timer 0-Interrupt jedes Mal, wenn der Timer 0 überläuft, also von 255 auf 0 wechselt. Würde zum Beispiel ein RB-Interrupt verwendet werden, so würde bei jeder Pegeländerung von RB4 bis RB7 ein Interrupt auftreten und die entsprechende ISR wird ausgeführt. Eine ISR sollte daher so kurz wie möglich sein.



Ein weiterer wichtiger Punkt bei einer ISR ist, dass das w-Register (Working- oder Arbeitsregister) und das STATUS-Register in andere Registern zwischengespeichert werden müssen, falls diese in der ISR ihren Registerinhalt verändern. Der Grund dafür ist, dass eine ISR eben unvorhergesehen aufgerufen wird, und die angesprochenen Register unter Umständen zu diesem Zeitpunkt gerade benötigte Werte enthalten. Nach Ausführung der ISR springt diese zwar wieder genau an die Stelle zurück, wo sie war, bevor der Interrupt auftauchte, aber mit einem möglicherweise falschen Wert im w-Register (bzw. STATUS-Register). Das Zwischenspeichern des w-Register bzw. des STATUS-Registers wird häufig auch als PUSH bezeichnet. Das Wiederherstellen von w-Register und STATUS-Register nennt man POP.

Woher weiß das Programm, dass ein Interrupt aufgerufen werden muss? Dazu gibt es für jede Interruptquelle ein Kontroll-Flag. Dies wird vom Controller gesetzt wenn dieser Interrupt auftritt. (Vorausgesetzt, dass diese Interruptquelle freigegeben ist). Damit aber die ISR nicht ständig aufgerufen wird, muss dieses Bit in der ISR wieder gelöscht werden.

Nun aber zur wanduhrspezifischen Timer 0-ISR. Diese hat lediglich die Aufgabe eine Zeitbasis für 4ms und eine Zeitbasis für 1 Sekunde zu erzeugen. Eine Zeitbasis für 4ms ist hier besonders einfach zu erzeugen, da ja diese ISR ohnehin schon zu genau diesen Zeitabständen aufgerufen wird. Es ist also nur notwendig ein Flag zu setzen. Dieses Flag trägt hier den Namen FLAG4MSEK und befindet sich im Register FLAGISRHP. Die Generierung für die 1-Sekunden-Zeitbasis ist dagegen schon etwas „umfangreicher“. Damit eine Zeit von einer Sekunde entsteht muss die ISR 250-mal aufgerufen werden ( $250 \times 4\text{ms} = 1000\text{ms} = 1 \text{ Sekunde}$ ). Bei jedem ISR-Aufruf muss also ein Zählregister um 1 vermindert werden. Besitzt es danach den Wert 0, so ist eine Sekunde vergangen. Nun wird das Botschaftsflag FLAG1SEK im Register FLAGISRHP gesetzt, und das Zählregister muss mit dem Wert 250 neu geladen werden. Der Wert 250 wird hier durch die Konstante KONSTISR1SEK ersetzt.

Die ISR wird, wie schon mehrmals erwähnt, alle 4ms aufgerufen Diese 4ms ergeben sich folgendermaßen: TMR0 wird mit dem Wert 0 geladen – es dauert also 256 Taktzyklen bis das Register wieder den Wert 0 besitzt, der Vorteiler besitzt den Wert 16 (vgl. Unterprogramm INIT, Abschnitt. 4.5.1). Der Taktzyklus ergibt sich aus dem verwendeten Quarz (X1). Dieser ist bei der PIC-Familie wie folgt definiert:

$$\frac{4}{f_{\text{Quarz}}}$$

Daraus ergibt sich folgender Zusammenhang:

$$ISRAUFRUF \ [\mu\text{s}] = \frac{4 \cdot 256 \cdot 16}{f_{\text{Quarz}} \ [\text{MHz}]} = \frac{4 \cdot 256 \cdot 16}{4,096} = 4000$$

Also ein ISR-Aufruf alle 4000 $\mu\text{s}$ , was gleichbedeutend mit 4ms ist.

Eine ISR muss mit dem Befehl *retfie* beendet werden.<sup>3</sup>

<sup>3</sup> Beim Aufruf der ISR wird automatisch das Bit GIE gelöscht, damit während der Ausführung der ISR kein weiterer Interrupt ausgelöst werden kann, was bei mehreren freigegebenen Interruptquellen durchaus möglich sein kann. Mit dem Befehl *retfie* wird zunächst an die Stelle im Programm zurückgesprungen, wo sich die Programmabarbeitung befand bevor die ISR aufgerufen wurde, und die verwendeten Interrupts werden wieder freigegeben (hier bei der Wanduhr der Timer0-Interrupt)

**Hier die gesamte ISR:**

```

ISR
PUSH    movwf  ISR_w_TEMP      ;w-Register retten
        swapf  STAT,w         ;Statusregister
        movwf  ISR_STAT_TEMP   ; retten

        ;Beginn der eigentlichen ISR-Routine
        bsf    FLAGSISRHP,FLAG4MSEK ;Botschaftsflag setzen

        decfsz ZAEHLERISR1SEK,f ;Zählregister für 1-Sekunden-Zeitbasis
        ; um 1 vermindern

        goto   ISRWEITER1
        bsf    FLAGSISRHP,FLAG1SEK ;Botschaftsflag setzen
        movlw  KONSTISR1SEK      ;Zählregister für den Sekundentakt mit
        movwf  ZAEHLERISR1SEK   ; der Konstanten KONSTISR1SEK laden

ISRWEITER1
        ;Ende der eigentlichen ISR-Routine
ISRFBERTIG bcf    INTCON,T0IF      ;T0-Interruptflag löschen

POP     swapf  ISR_STAT_TEMP,w   ;Status-Register
        movwf  STAT             ; und
        swapf  ISR_w_TEMP,f     ; w-Register
        swapf  ISR_w_TEMP,w     ; wieder herstellen

        retfie

```

## 4.5. Unterprogramme

Die insgesamt sieben Unterprogramme lassen sich folgendermaßen einteilen:

- Unterprogramm zur Initialisierung des Mikrocontroller (INIT)
- Unterprogramme zur DCF-Dekodierung (DCFROUTINE, DCFUPSEKUNDE und DCFUPMINUTE)
- Unterprogramm, für den Fall dass kein DCF-Empfang möglich ist (INNEREUHR)
- Unterprogramm zur Ausgabe der Uhrzeit (ANZEIGE)
- Unterprogramm für Berechnungen, Umwandlungen und dergleichen (BCDBIN2)

### 4.5.1. INIT

Dieses Unterprogramm dient zur Initialisierung des Mikrocontrollers. Der Portpin an dem der DCF-Empfänger angeschlossen ist muss als Eingang definiert werden, während die Portpins zur Anzeige der Stunde, Minute und Sekunde als Ausgang definiert werden.

In der Initialisierungsroutine muss für den erste Aufruf des Unterprogramms DCFROUTINE der aktuelle Zustand dieses Portpins gelesen, und im Register DCFSTATUS kopiert werden.

Da das Unterprogramm DCFROUTINE zyklisch (alle 4ms) aufgerufen wird, ist eine entsprechende Zeitbasis notwendig. Diese wird mit Hilfe eines Timer-Interrupts erzeugt. Für die Definition der Zeitbasis ist hier das mikrocontrollerinterne Funktions-Register OPTREG (in der Registerbank 1) zuständig. Damit bei einer PIC-Taktfrequenz von 4,096MHz eine Zeitbasis von 4ms erzeugt wird, muss das Register OPTREG mit dem

binären Wert b'00000011' geladen werden. Das Zählregister für dies Zeitbasis (Funktions-Register TMR0, in Registerseite 0) muss gelöscht werden.

Weiters müssen einige Register vorbelegt (gelöscht) werden.

### Hier das Unterprogramm:

```

INIT      clrf  TMR0                ;Timer0 auf 0 voreinstellen

          bank1                    ;Registerseite 1
          movlw b'00000011'        ;interner Takt, Vorteiler = 16 an TMR0
          movwf OPTREG
          clrf  TRISA              ;Port A: Ausgänge (Stundenanzeige)

          movlw b'00000000'        ;Port B: Ausgang
          movwf TRISB

          bsf   DCFINTRIS,DCFIN    ;DCFIN als Eingang definieren
          bank0                    ;Registerseite 0

          movlw KONSTISR1SEK      ;Zählregister für den Sekundentakt mit
          movwf ZAEHLERISR1SEK   ; der Konstanten KONSTISR1SEK laden

          clrf  DCFSTATUS         ;DCF-Statusregister (DCFSTATUS)
                                   ; initialisieren
          bsf   DCFSTATUS,DCFFEHLE ; Fehlerflag setzen, alle anderen Flags
                                   ; löschen

          btfs  DCFINPORT,DCFIN   ;Pegel von DCFIN einlesen und im Flag
          bsf   DCFSTATUS,DCFPORTNEU ; DCFPORTNEU des Register DCFSTATUS
                                   ; sichern

          movlw b'11110'          ;Anzeige löschen, dazu Werte am Port A
          movwf PORTA            ; und am Port B ausgeben, die keiner
          movlw b'00111111'      ; Uhrzeit entsprechen
          movwf PORTB

          clrf  UHRSEKUNDE        ;Sekundenzähler initialisieren
          clrf  UHRMINUTE         ;Minutenzähler initialisieren
          clrf  UHRSTUNDE        ;Stundenzähler initialisieren
          clrf  DATUMWTAG         ;Wochentagzähler initialisieren
          clrf  DATUMTAG         ;Tageszähler initialisieren
          clrf  DATUMMONAT       ;Monatszähler initialisieren
          clrf  DATUMJAHR        ;Jahreszähler initialisieren

          movlw .99
          movwf DCFMINALT         ;Vergleichsregister initialisieren
          movwf DCFSTDALT        ;Vergleichsregister initialisieren
          movwf DCF'TAGALT       ;Vergleichsregister initialisieren
          movwf DCFMONALT        ;Vergleichsregister initialisieren
          movwf DCFJAHRALT       ;Vergleichsregister initialisieren
          movwf DCFWTAGALT       ;Vergleichsregister initialisieren
          return

```

## 4.5.2. DCFROUTINE

Das Unterprogramm DCFROUTINE ist für die DCF-Dekodierung die wichtigste Komponente.

Diese Routine wird ca. alle 4 ms aufgerufen

**Aufgabe und Vorgehensweise:**

Die Aufgabe des Unterprogramms DCFROUTINE besteht darin aus den Abtastungen herauszufinden, ob ein Low, ein High oder ein Minutenwechsel gesendet wurde. Dazu wird bei jedem Aufruf des Unterprogramms DCFROUTINE entweder das Zählregister DCFPULS oder das Zählregister DCFPAUSE um 1 erhöht (inkrementiert), wenn sich der Pegel vom DCF-Eingang zum vorhergehenden Aufruf dieses Unterprogramms nicht verändert hat. Ist der DCF-Eingang Low (also logisch 0), so wird das Zählregister DCFPULS um 1 erhöht, ist der DCF-Eingang High (also logisch 1), so wird das Zählregister DCFPAUSE um 1 erhöht. Beide Zählregister können maximal den Wert 255 aufnehmen, was einer Zeit von 1020 ms (oder 1,02 Sekunden) entspricht ( $=255 \cdot 4\text{ms}$ ).

Unterscheidet sich der aktuelle Pegel des DCF-Eingangs mit dem Pegel vom vorhergehenden Aufruf, so spricht man von einer Flanke, wobei hier zwischen einer fallenden und einer steigenden Flanke unterschieden werden muss. Bei einer fallenden Flanke (der aktuelle Pegel des DCF-Eingangs ist logisch 0, während der Pegel beim vorhergehenden Aufruf noch logisch 1 war), erfolgt zunächst eine Auswertung des Zählregisters DCFPULS. Beinhaltet dieses Zählregister einen Wert zwischen den Konstanten DCFLOWMIN und DCFLOWMAX, so wurde ein LOW des DCF-Telegramms ermittelt. Im DCF-Statusregister (DCFSTATUS) werden daher die Flags DCFLOW und DCFNEUESEK gesetzt. Das Flag DCFNEUESEK dient als Zeichen einer neu empfangenen und gültigen Sekunde. Anschließend wird das Zählregister DCFPULS gelöscht. Beinhaltet das Zählregister DCFPULS einen Wert zwischen den Konstanten DCFHIGHMIN und DCFHIGHMAX, so wurde ein HIGH des DCF-Telegramms ermittelt. Im DCF-Statusregister (DCFSTATUS) werden daher die Flags DCFHIGH und DCFNEUESEK gesetzt. Das Flag DCFNEUESEK dient auch hier als Zeichen einer neu empfangenen und gültigen Sekunde, und auch das Zählregister DCFPULS wird anschließend gelöscht. Beinhaltet das Zählregister DCFPULS einen Wert der weder zwischen den Konstanten DCFLOWMIN und DCFLOWMAX noch zwischen den Konstanten DCFHIGHMIN und DCFHIGHMAX liegt, so handelt es sich um einen Übertragungsfehler. In diesem Fall wird das Fehlerflag (DCFFEHLER) im DCF-Statusregister (DCFSTATUS) gesetzt und auch das Zählregister DCFPULS wird anschließend gelöscht. Das Flag DCFNEUESEK wird hier aber nicht gesetzt, da es sich in diesem Fall um keine gültige Sekunde handelt. Das Fehlerflag wird erst bei der Erkennung einer neuen Minute wieder gelöscht. Achtung: Solange das Fehlerflag gesetzt ist erfolgt **keine** Auswertung des Zählregisters DCFPULS. Dieses Zählregister wird aber dennoch bei jeder fallenden Flanke zurückgesetzt.

Bei einer steigenden Flanke (der aktuelle Pegel des DCF-Eingangs ist logisch 1, während der Pegel beim vorhergehenden Aufruf noch logisch 1 war) wird das Zählregister DCFPAUSE gelöscht.

Entsprechend dem im Abschnitt 2 beschriebenen Protokoll erfolgt in der 59. Sekunde keine Absenkung des Trägers. Da es also in der 59. Sekunde keine fallende Flanke gibt, gibt es auch keine steigende Flanke. Das Zählregister DCFPAUSE „läuft über“, da es nur einen Zählbereich von 0 bis 255 besitzt. Dieser Überlauf wird hier ausgenützt. Zur Bestimmung einer neuen Minute bzw. zur Bestimmung des Telegrammbeginns. Ist das Fehlerflag gesetzt, so wird es nun gelöscht. War es nicht gesetzt, so wird das Flag DCFNEUEMIN im DCF-Statusregister (DCFSTATUS) gesetzt.

Das folgende Flussdiagramm soll das soeben beschriebene und umfangreiche Unterprogramm verdeutlichen. In den grau hinterlegten Bereichen erfolgt das Setzen oder Rücksetzen der zur weiteren Verwendung notwendigen Übergabeflags im DCF-Statusregister (DCFSTATUS). Die Unterprogramme DCFUPSEKUNDE und DCFUPMINUTE greifen auf diese Flags zu.

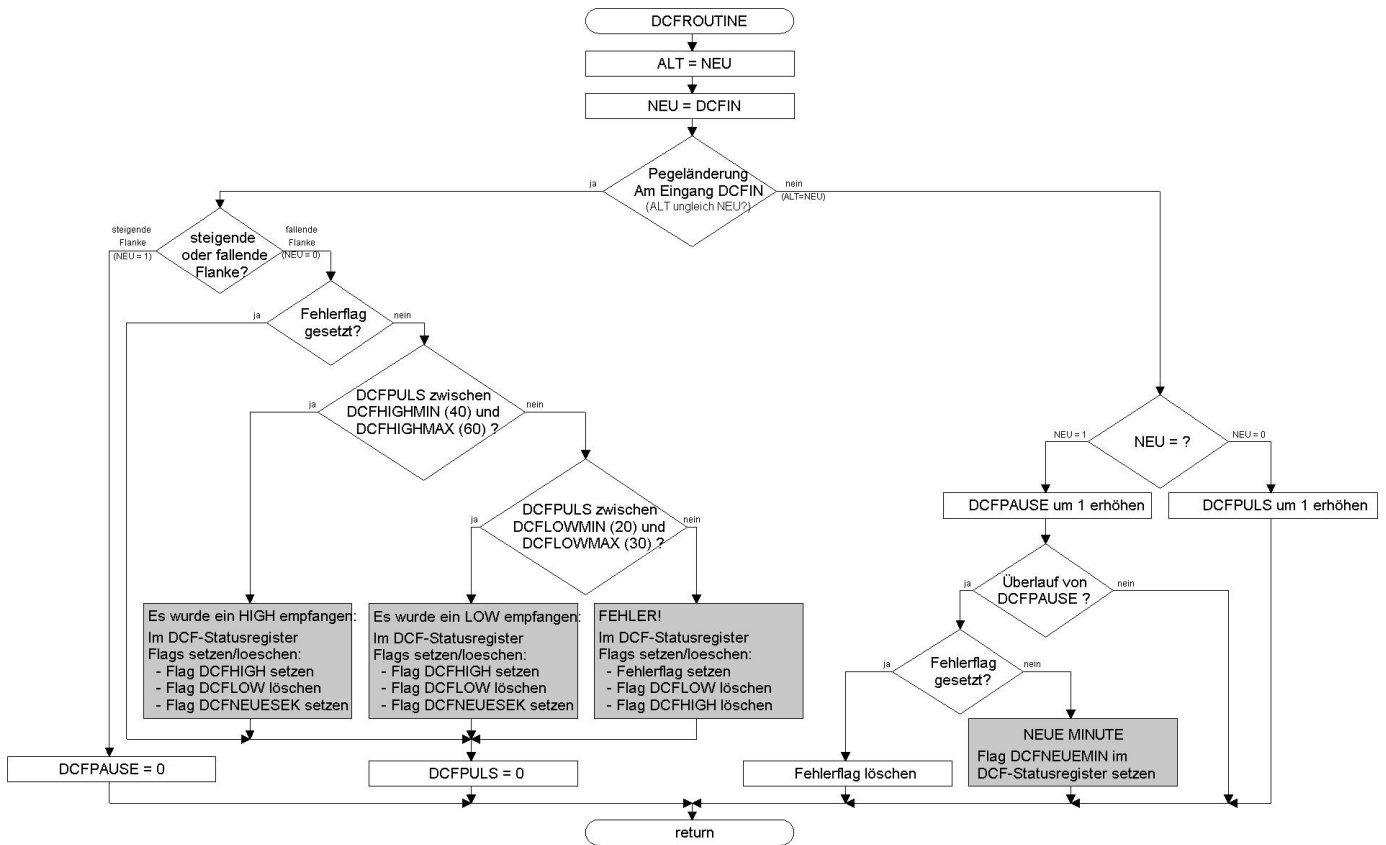


Bild 4.1: Flussdiagramm (DCFROUTINE)

**Hier das Unterprogramm:**

```

DCFROUTINE  bcf    DCFSTATUS,DCFPORTALT    ;DCFSTATUS,DCFPORTNEU ->
                                                    ; DCFSTATUS,DCFPORTALT

            btfsc  DCFSTATUS,DCFPORTNEU
            bsf    DCFSTATUS,DCFPORTALT

            movf   DCFINPORT,w
            movwf  TEMP1
            bcf    DCFSTATUS,DCFPORTNEU    ;DCFIN -> DCFSTATUS,DCFPORTNEU,
                                                    ; TEMP1,DCFPORTALT

            btfsc  TEMP1,DCFIN
            bsf    DCFSTATUS,DCFPORTNEU
            clrf   TEMP2
            btfsc  TEMP1,DCFIN
            bsf    TEMP2,DCFPORTALT

            movf   DCFSTATUS,w
            xorwf  TEMP2,f

            btfss  TEMP2,DCFPORTALT        ;Hat sich am Pin DCFIN der Pegel
                                                    ; geändert?
            goto   DCFINCPULSPAUSE        ;nein: Puls- bzw. Pausenzähler um 1
                                                    ; erhöhen
            btfss  DCFSTATUS,DCFPORTNEU   ;ja: -> steigende oder fallende
                                                    ; Flanke?
            goto   DCFHLFLANKE           ;fallende Flanke
DCFHLFLANKE clrf   DCFPAUSE              ;steigende Flanke: Zählregister
                                                    ; DCFPAUSE
            goto   DCFROUTINEFERTIG      ; löschen und UP verlassen

            ;Fallende Flanke
    
```

## Analog/Digitale Wanduhr 2

```

DCFHLFLANKE  btfss DCFSTATUS,DCFFEHLER    ;Fehlerflag gesetzt:
              goto  DCFPLAUSIBEL          ;nein: Ermittlung, ob die ermittelte
                                          ; Pulszeit einem HIGH oder einem
                                          ; LOW entspricht
                                          ; (Plausibilitätsprüfung)
              goto  DCFPULSLOESCHEN       ;ja: Puls-Zähler löschen

DCFPLAUSIBEL ;Prüfen, ob ein High empfangen wurde
DCFCECHKHIGH
              movlw DCFHIGHMIN             ;unteren High-Grenzwert laden
              subwf DCFPULS,w
              btfss STAT,C                 ;Grenzwert < DCFPULS ?
              goto  DCFCECHKLOW            ;nein: kein HIGH-Pegel, Prüfen, ob
                                          ; LOW
              movlw DCFHIGHMAX             ;ja: DCFPULS mit oberen HIGH-
                                          ; Grenzwert vergleichen

              subwf DCFPULS,w
              btfsc STAT,C                 ;DCFPULS < Grenzwert ?
              goto  DCFCECHKLOW            ;nein: es handelt sich um kein HIGH-
                                          ; Signal, Prüfen, ob ein LOW
                                          ; empfangen wurde
              bcf   DCFSTATUS,DCFLOW       ;ja: Es wurde ein High-Signal
                                          ; empfangen, das Flag DCFLOW
                                          ; löschen,
              bsf   DCFSTATUS,DCFHIGH      ; das Flag DCFHIGH setzen,
              bsf   DCFSTATUS,DCFNEUESEK  ; das Flag DCFNEUESEK setzen
              goto  DCFPULSLOESCHEN       ; und Puls-Zähler löschen

              ;Prüfen, ob ein Low empfangen wurde
DCFCECHKLOW  movlw DCFLOWMIN               ;unteren Low-Grenzwert laden
              subwf DCFPULS,w
              btfss STAT,C                 ;Grenzwert < DCFPULS ?
              goto  DCFSETFEHLER          ;nein: kein LOW-Pegel,
                                          ; Empfangsfehler
              movlw DCFLOWMAX              ;ja: DCFPULS mit oberen LOW-
                                          ; Grenzwert vergleichen

              subwf DCFPULS,w
              btfsc STAT,C                 ;DCFPULS < Grenzwert ?
              goto  DCFSETFEHLER          ;nein: es handelt sich um kein LOW-
                                          ; Signal -> Empfangsfehler
              bcf   DCFSTATUS,DCFHIGH      ;ja: Es wurde ein LOW-Signal
                                          ; empfangen, das Flag DCFLOW
                                          ; setzen,
              bsf   DCFSTATUS,DCFLOW       ; das Flag DCFHIGH löschen,
              bsf   DCFSTATUS,DCFNEUESEK  ; das Flag DCFNEUESEK setzen
              goto  DCFPULSLOESCHEN       ; und den Puls-Zähler löschen

              ;Empfangsfehler
DCFSETFEHLER
              bsf   DCFSTATUS,DCFFEHLER    ;Bei einem Empfangsfehler
              bcf   DCFSTATUS,DCFHIGH      ; Fehlerflag (DCFFEHLER) setzen,
              bcf   DCFSTATUS,DCFLOW       ; die Flags DCFLOW und DCFHIGH
                                          ; löschen,
              goto  DCFPULSLOESCHEN       ; und den Puls-Zähler löschen

              ;Puls- oder Pausenzähler erhöhen
DCFINCPULSPAUSE
              btfsc DCFSTATUS,DCFPORTNEU
              goto  DCFINCPAUSE

DCFINCPULS   incf  DCFPULS,f              ;Puls-Zähler um 1 erhöhen
              goto  DCFROUTINEFERTIG

```

```

DCFINCPAUSE incfsz DCFPAUSE,f           ;Pausezähler erhöhen und auf
                                           ; Überlauf Prüfen
      goto DCFROUTINEFERTIG           ;kein überlauf: Unterprogramm
                                           ; verlassen

      btfs c DCFSTATUS,DCFNEUEMIN      ;Überlauf: NEUE MINUTE: wenn
                                           ; Fehlerflag
      goto DCFROUTINEW1

      bsf DCFSTATUS,DCFNEUEMIN         ;nicht gesetzt, Flag DCFNEUEMIN
                                           ; setzen
      goto DCFROUTINEFERTIG           ; und Unterprogramm verlassen

DCFROUTINEW1
      bcf DCFSTATUS,DCFNEUEMIN         ;Fehlerflag gesetzt: Fehlerflag
                                           ; löschen
      goto DCFROUTINEFERTIG           ; und Unterprogramm verlassen (Flag
                                           ; DCFNEUEMIN wird in diesem Fall
                                           ; nicht gesetzt)

                                           ;Pulszähler löschen
DCFPULSLOESCHEN
      clrf DCFPULS

DCFROUTINEFERTIG
      return
    
```

**Anmerkung:**

Die temporären Register TEMP1 und TEMP2 werden hier nur als Hilfsregister benötigt. Sie können daher auch in anderen Unterprogrammen verwendet werden.

### 4.5.3. DCFUPSEKUNDE

**Aufgaben:**

- Je nachdem ob das Bit DCFLOW oder das Bit DCFHIGH gesetzt ist dieses Bit dem Telegramm (Register DCFTELEGRAMM1 bis DCFTELEGRAMM8) hinzufügen
- Anforderungsflag (Flag DCFNEUESEK) zurücksetzen

**Vorgehensweise:**

Ist das Flag DCFLOW (im Register DCFSTATUS) gesetzt das Carryflag (im SFR STAT) löschen, ist aber das Flag DCFHIGH (ebenfalls im Register DCFSTATUS) gesetzt das Carryflag setzen. Dieses Carryflag nun dem Telegramm hinzufügen. Dieser Vorgang erfolgt mit einem so genannten Schiebebefehl. Dabei werden alle Bits des Registers DCFTELEGRAMM1 auf die nächst höhere Position verschoben. (Bit 6 wandert ins Bit 7, Bit 5 wandert ins Bit 6 usw. Bit 0 wandert ins Bit 1). Der Inhalt vom Carryflag wandert ins Bit 0. Der Inhalt von Bit 7 wird ins Carryflag geschoben. Bei den Registern DCFTELEGRAMM2 bis DCFTELEGRAMM6 erfolgt derselbe Vorgang. (Der Inhalt von Bit 7 des Registers DCFTELEGRAMM1 wird ins Carry geschoben, und das Carry aber weiter in das Bit 0 des Registers DCFTELEGRAMM2)

**Anmerkung:**

Die Flags DCFLOW und DCFHIGH können nie gleichzeitig gesetzt sein. Es ist nur möglich, dass entweder nur DCFLOW oder nur DCFHIGH gesetzt ist, aber nie beide gleichzeitig.

**Hier das Unterprogramm:**

```

DCFUPSEKUNDE
    btfsc DCFSTATUS,DCFLOW ;Ist das Flag DCFLOW im Register DCFSTATUS
                          ; gesetzt?
    goto DCFSCHIEBELOW
    btfsc DCFSTATUS,DCFHIGH ;nein: Wenn das FLAG DCFHIGH im Register
DCFSCHIEBEHIGH
    bsf STAT,C ; DCFSTATUS gesetzt ist, dem Telegramm
    goto DCFSCHIEBE ; ein Low mit Hilfe des Carry hinzufügen
DCFSCHIEBELOW
    bcf STAT,C ;ja: Dem Telegramm ein High mit Hilfe des
                ; Carry hinzufügen

DCFSCHIEBE rlf DCFTELEGRAMM1,f
            rlf DCFTELEGRAMM2,f
            rlf DCFTELEGRAMM3,f
            rlf DCFTELEGRAMM4,f
            rlf DCFTELEGRAMM5,f
            rlf DCFTELEGRAMM6,f
            ; rlf DCFTELEGRAMM7,f
            ; rlf DCFTELEGRAMM8,f

            bcf DCFSTATUS,DCFNEUESEK ;Anforderungsflag löschen
            return

```

**4.5.4. DCFUPMINUTE****Aufgaben:**

- Fehlende 59. Sekunde "nachholen" (Unterprogramm DCFUPSEKUNDE aufrufen)
- Datum, Uhrzeit und die Zusatzinformationen aus den Registern DCFTELEGRAMM1 bis DCFTELEGRAMM8 zusammensetzen
- Das soeben ermittelte Telegramm mit dem Telegramm der vorhergehenden Minute vergleichen. Die soeben ermittelte Minute muss dabei um 1 größer als die vorhergehende Minute sein, und die soeben ermittelten Werte für die Stunde, Tag, Monat, Jahr und Wochentag muss mit den Werten des vorhergehenden Telegramms übereinstimmen. ACHTUNG: Bei dieser einfachen Überprüfung wird der Übergang von z. B. 13:59 auf 14:00 als falsch gewertet, da sich hier die Stunde ändert, und auch die Minute um mehr als 1. Würden hier alle möglichen Übergänge berücksichtigt, dann würde dieses Unterprogramm noch umfangreicher als es ohnehin schon ist. Durch diese Vereinfachung dauert die Synchronisierung im schlimmsten Fall nur eine Minute länger.
- Das alte Datum bzw. die alte Uhrzeit (von der vorhergehenden Minute) durch das neue Datum bzw. Uhrzeit ersetzen. Dies ist für die Überprüfung des nächsten Telegramms notwendig.
- Wenn das neu empfangene Datum bzw. die neu empfangene Uhrzeit gültig ist, die BCD-Kodierten Datums- bzw. Uhrzeitkomponenten in eine binäre Form umwandeln und damit die mitlaufende Uhr synchronisieren. Das Flag DCFSYNC im Register DCFSTATUS setzen. Dieses gesetzte Flag kennzeichnet, dass die mitlaufende Uhr mit der DCF-Uhr synchronisiert ist.
- Anforderungsflag (Flag DCFNEUEMIN im Register DCFSTATUS) zurücksetzen

**Hier das Unterprogramm:**

```

DCFUPMINUTE call DCFUPSEKUNDE ;Fehlende Sekunde nachholen

            clr DCFJAHRBCD ;Jahr zusammensetzen (BCD-Format)
            btfsc DCFTELEGRAMM1,1

```



## Analog/Digitale Wanduhr 2

```
bsf   DCFJAHRBCD, 7
btfsc DCFTELEGRAMM1, 2
bsf   DCFJAHRBCD, 6
btfsc DCFTELEGRAMM1, 3
bsf   DCFJAHRBCD, 5
btfsc DCFTELEGRAMM1, 4
bsf   DCFJAHRBCD, 4
btfsc DCFTELEGRAMM1, 5
bsf   DCFJAHRBCD, 3
btfsc DCFTELEGRAMM1, 6
bsf   DCFJAHRBCD, 2
btfsc DCFTELEGRAMM1, 7
bsf   DCFJAHRBCD, 1
btfsc DCFTELEGRAMM2, 0
bsf   DCFJAHRBCD, 0

clrf  DCFMONBCD           ;Monat zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM2, 1
bsf   DCFMONBCD, 4
btfsc DCFTELEGRAMM2, 2
bsf   DCFMONBCD, 3
btfsc DCFTELEGRAMM2, 3
bsf   DCFMONBCD, 2
btfsc DCFTELEGRAMM2, 4
bsf   DCFMONBCD, 1
btfsc DCFTELEGRAMM2, 5
bsf   DCFMONBCD, 0

clrf  DCFWOCHENTAG       ;Wochentag zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM2, 6
bsf   DCFWOCHENTAG, 2
btfsc DCFTELEGRAMM2, 7
bsf   DCFWOCHENTAG, 1
btfsc DCFTELEGRAMM3, 0
bsf   DCFWOCHENTAG, 0

clrf  DCFTAGBCD          ;Tag zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM3, 1
bsf   DCFTAGBCD, 5
btfsc DCFTELEGRAMM3, 2
bsf   DCFTAGBCD, 4
btfsc DCFTELEGRAMM3, 3
bsf   DCFTAGBCD, 3
btfsc DCFTELEGRAMM3, 4
bsf   DCFTAGBCD, 2
btfsc DCFTELEGRAMM3, 5
bsf   DCFTAGBCD, 1
btfsc DCFTELEGRAMM3, 6
bsf   DCFTAGBCD, 0

clrf  DCFSTDBCD          ;Stunde zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM4, 0
bsf   DCFSTDBCD, 5
btfsc DCFTELEGRAMM4, 1
bsf   DCFSTDBCD, 4
btfsc DCFTELEGRAMM4, 2
bsf   DCFSTDBCD, 3
btfsc DCFTELEGRAMM4, 3
bsf   DCFSTDBCD, 2
btfsc DCFTELEGRAMM4, 4
bsf   DCFSTDBCD, 1
btfsc DCFTELEGRAMM4, 5
bsf   DCFSTDBCD, 0
```

## Analog/Digitale Wanduhr 2

```

clrf DCFMINBCD          ;Minute zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM4,7
bsf DCFMINBCD,6
btfsc DCFTELEGRAMM5,0
bsf DCFMINBCD,5
btfsc DCFTELEGRAMM5,1
bsf DCFMINBCD,4
btfsc DCFTELEGRAMM5,2
bsf DCFMINBCD,3
btfsc DCFTELEGRAMM5,3
bsf DCFMINBCD,2
btfsc DCFTELEGRAMM5,4
bsf DCFMINBCD,1
btfsc DCFTELEGRAMM5,5
bsf DCFMINBCD,0

;
; clrf DCFZUSATZINFOS          ;Zusätzliche Informationen
; btfsc DCFTELEGRAMM5,7          ; Zusätzliche Schaltsekunde
; bsf DCFZUSATZINFOS,DCFSCHALTSEK
; btfsc DCFTELEGRAMM6,0          ; Sommerzeit
; bsf DCFZUSATZINFOS,DCFSOMMER
; btfsc DCFTELEGRAMM6,1          ; Winterzeit
; bsf DCFZUSATZINFOS,DCFWINTER
; btfsc DCFTELEGRAMM6,2          ; Vorankündigung: Wechsel
; bsf DCFZUSATZINFOS,DCFSOMWIN ; Sommerzeit <-> Winterzeit
; btfsc DCFTELEGRAMM6,3          ; Reserveantenne
; bsf DCFZUSATZINFOS,DCFRESANT

;Neues Telegramm mit dem alten Vergleichen
bcf DCFSTATUS,DCFFEHLER ;zuerst Fehlerflag löschen

movf DCFMINALT,w          ;Minuten prüfen
subwf DCFMINBCD,w
movwf TEMP1              ;TEMP1 = DCFMINBCD - DCFMINALT
decfsz TEMP1,w           ;TEMP1 - 1 = 0 ?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf DCFSTDALT,w          ;Stunden prüfen
subwf DCFSTDBCD,w
btfss STAT,Z             ;DCFSTDBCD - DCFSTDALT = 0?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf DCFTAGALT,w          ;Tag prüfen
subwf DCFTAGBCD,w
btfss STAT,Z             ;DCFTAGBCD - DCFTAGBCD = 0?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf DCFMONALT,w          ;Monat prüfen
subwf DCFMONBCD,w
btfss STAT,Z             ;DCFMONBCD - DCFMONALT = 0?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf DCFJAHRALT,w          ;Jahr prüfen
subwf DCFJAHRBCD,w
btfss STAT,Z             ;DCFJAHRBCD - DCFJAHRALT = 0?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf DCFWTAGALT,w          ;Wochentag prüfen
subwf DCFWOCHENTAG,w
btfss STAT,Z             ;DCFWOCHENTAG - DCFWTAGALT = 0?
bsf DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

```

## Analog/Digitale Wanduhr 2

```
;Altes Telegramm durch das neue Telegramm ersetzen
movf DCFMINBCD,w
movwf DCFMINALT

movf DCFSTDBCD,w
movwf DCFSTDALT

movf DCF'TAGBCD,w
movwf DCF'TAGALT

movf DCFMONBCD,w
movwf DCFMONALT

movf DCFJAHRBCD,w
movwf DCFJAHRALT

movf DCFWOCHENTAG,w
movwf DCFWTAGALT

btfsc DCFSTATUS,DCFFEHLER ;DCF-Telegramm korrekt?
goto DCFUPMINUTEENDE ;nein: UP beenden
movf DCFSTDBCD,w ;ja: DCFSTDBCD
call BCDBIN2 ;von BCD nach binär umwandeln
movwf UHRSTUNDE ;und die Stunde der mitlaufenden Uhr
;überschreiben

movf DCFMINBCD,w ;DCFMINBCD
call BCDBIN2 ;von BCD nach binär umwandeln
movwf UHRMINUTE ;und die Minute der mitlaufenden Uhr
;überschreiben

clrf UHRSEKUNDE ;Sekunde löschen

movf DCF'TAGBCD,w ;DCF'TAGBCD
call BCDBIN2 ;von BCD nach binär umwandeln
movwf DATUMTAG ;und den Tag des mitlaufenden Datums
;überschreiben

movf DCFMONBCD,w ;DCFMONBCD
call BCDBIN2 ;von BCD nach binär umwandeln
movwf DATUMMONAT ;und den Monat des mitlaufenden
;Datums überschreiben

movf DCFJAHRBCD,w ;DCFJAHRBCD
call BCDBIN2 ;von BCD nach binär umwandeln
movwf DATUMJAHR ;und das Jahr des mitlaufenden
;Datums überschreiben

movf DCFWOCHENTAG,w ;DCFWOCHENTAG
movwf DATUMWTAG

bsf DCFSTATUS,DCFSYNC

DCFUPMINUTEENDE bcf DCFSTATUS,DCFFEHLER
bcf DCFSTATUS,DCFNEUEMIN ;Anforderungsflag löschen

return
```

### Anmerkung:

Das temporäre Register TEMP1 wird hier nur als Hilfsregister benötigt, und kann daher auch in anderen Unterprogrammen verwendet werden.

## 4.5.5. INNEREUHR

### Aufgabe:

Für den Fall dass kein gültiges DCF-Telegramm empfangen werden kann sorgt dieses Unterprogramm dafür, dass die Uhrzeit trotzdem jede Sekunde aktualisiert wird. Ist für eine längere Zeit kein korrekter DCF-Empfang möglich, so würden die Minuten, Stunden, Tage usw. stehen bleiben, da ja im Unterprogramm DCFUPMINUTE nur gültige DCF-Telegramme übernommen werden.

Dieses Unterprogramm wird also parallel zu den Unterprogrammen für die DCF-Dekodierung aufgerufen.

### Vorgehensweise:

Die Sekunden (Register UHRSEKUNDE) um 1 erhöhen. Beinhaltet dieses Register nun den Wert 60, so beginnt eine neue Minute. Daher die Sekunden löschen und die Minuten (Register UHRMINUTE) um 1 erhöhen. Beinhaltet dieses Register nun den Wert 60, so beginnt eine neue Stunde. Daher die Minuten löschen und die Stunden (Register UHRSTUNDE) um 1 erhöhen. Beinhaltet dieses Register nun den Wert 24, so beginnt ein neuer Tag. Daher die Stunden löschen. Ist das Mitzählen des Datums notwendig, so muss nun ein Register für den Tag (z.B. DATUMTAG) um 1 erhöht werden und dieses überprüft werden, wobei diese Prüfung nun nicht mehr so einfach ist, da ja jeder Monat unterschiedlich viele Tage besitzt. Erschwerend kommt auch noch hinzu, dass auch die Schaltjahre miteinbezogen werden müssen!

### Achtung:

Dieses Unterprogramm muss daher jede Sekunde aufgerufen werden.

### Hier das Unterprogramm:

```
INNEREUHR    incf  UHRSEKUNDE,f      ;Sekundenzähler um 1 erhöhen
             movf  UHRSEKUNDE,w
             sublw .60
             btfss STAT,Z          ;Prüfen, ob Sekunde=60
             goto  INNERERUHRFERTIG
             clrf  UHRSEKUNDE      ;Wenn Sekunde=60, Sekunde löschen
             incf  UHRMINUTE,f      ; und Minute um 1 erhöhen
             movf  UHRMINUTE,w
             sublw .60
             btfss STAT,Z          ;Prüfen, ob Minute=60
             goto  INNERERUHRFERTIG
             clrf  UHRMINUTE      ;Wenn Minute=60, Minute löschen
             incf  UHRSTUNDE,f      ; und Stunde erhöhen
             movf  UHRSTUNDE,w
             sublw .24
             btfsc STAT,Z          ;Prüfen, ob Stunde=24
             clrf  UHRSTUNDE      ;Wenn Stunde=24, Stunden löschen
INNERERUHRFERTIG
             return
```

## 4.5.6. ANZEIGE

Dieses Unterprogramm wird vom Hauptprogramm alle 4ms aufgerufen.

### Aufgabe und Vorgehensweise:

Wenn die mitlaufende Uhr mit der DCF-Uhr synchronisiert ist (Flag DCFSYNC im Register DCFSTATUS gesetzt) die Uhrzeit wie folgt ausgeben. Andernfalls dieses Unterprogramm vorzeitig beenden: Die Stunde am Port A und abwechselnd Minute und Sekunde am Port B ausgeben. Abwechselnd bedeutet hier, dass beim Aufruf dieses Unterprogramms entweder die Minuten am Port B ausgegeben werden oder die Sekunden. Diese Methode entspricht einer einfachen Multiplex-Ansteuerung. Die Unterscheidung, ob die Minuten oder die Sekunden ausgegeben werden hängt vom Bit 0 des Registers FLAGS ab. Dieses Bit wird bei jedem Aufruf invertiert (getoggelt). Besitzt es danach den Wert 1, so werden die Minuten angezeigt. Beim nächsten Aufruf dieses Unterprogramms (in ca. 4ms) werden dann die Sekunden angezeigt, da dieses Bit ja wieder invertiert wurde und daher den Wert 0 besitzt.

Mit Hilfe der beiden Tabellen TABSTUNDEN und TABMINUTENSEKUNDEN erfolgt eine Anpassung an die Hardware (Umwandlung von 24 Stunden auf 12 Stunden, LED-Matrix am Port B)

### Hier das Unterprogramm:

```

ANZEIGE      btfss DCFSTATUS,DCFSYNC ;Ist die mitlaufende Uhr mit der DCF-Uhr
              ; synchronisiert?
              goto ANZEIGEENDE ;nein: Unterprogramm vorzeitig beenden

              movf UHRSTUNDE,w ;ja: Stunde mit Hilfe der Tabelle
TABSTUNDE
              andlw b'00011111'
              call TABSTUNDEN ; von 0-23 auf 0-11 umwandeln und um eine
              ; Stelle verschieben
              movwf PORTA ; und am Port A ausgeben

              btg FLAGS,0
              btfss FLAGS,0 ;Abwechselnd
              goto ANZEIGESEK

ANZEIGEMIN   movf UHRMINUTE,w ; Minute
              goto ANZEIGEW1

ANZEIGESEK   movf UHRSEKUNDE,w ; oder Sekunde

ANZEIGEW1    andlw b'00111111'
              call TABMINUTENSEKUNDEN; so anpassen, dass die richtige
              movwf PORTB ; Leuchtdiode am Port B leuchtet

ANZEIGEENDE return

```

## 4.5.7. BCDBIN2

### Aufgabe:

Die im w-Register stehende 2-stellige-BCD-Zahl nach Binär umwandeln. Das Ergebnis befindet sich wieder im w-Register.

**Vorgehensweise:**

- Die umzuwandelnde BCD-Zahl im temporären Register TEMP1 zwischenspeichern
- w-Register (Arbeitsregister) löschen.
- Die BCD-Zahl bitweise überprüfen.

z.B.: BCD-Zahl:      0010 0110    (= 26)

		-		0	( = 0 x 1)
		---	+	2	( = 1 x 2)
		-----	+	4	( = 1 x 4)
		-----	+	0	( = 0 x 8)
		-----	+	0	( = 0 x 10)
		-----	+	20	( = 1 x 20)
		-----	+	0	( = 0 x 40)
		-----	+	0	( = 0 x 80)
				=	26

- Ergebnis in w

**Hier das Unterprogramm:**

```
BCDBIN2    movwf TEMP1            ;umzuwandelnde BCD-Zahl zwischenspeichern
          clrw                    ;Arbeitsregister (w-Register) löschen
          btfsc TEMP1,0          ;Ist Bit0 der BCD-Zahl gesetzt,
          addlw .1                ; zum w-Register den Wert 1 addieren
          btfsc TEMP1,1          ;Ist Bit1 der BCD-Zahl gesetzt,
          addlw .2                ; zum w-Register den Wert 2 addieren
          btfsc TEMP1,2          ;Ist Bit2 der BCD-Zahl gesetzt,
          addlw .4                ; zum w-Register den Wert 4 addieren
          btfsc TEMP1,3          ;Ist Bit3 der BCD-Zahl gesetzt,
          addlw .8                ; zum w-Register den Wert 8 addieren
          btfsc TEMP1,4          ;Ist Bit4 der BCD-Zahl gesetzt,
          addlw .10               ; zum w-Register den Wert 10 addieren
          btfsc TEMP1,5          ;Ist Bit5 der BCD-Zahl gesetzt,
          addlw .20               ; zum w-Register den Wert 20 addieren
          btfsc TEMP1,6          ;Ist Bit6 der BCD-Zahl gesetzt,
          addlw .40               ; zum w-Register den Wert 40 addieren
          btfsc TEMP1,7          ;Ist Bit7 der BCD-Zahl gesetzt,
          addlw .80               ; zum w-Register den Wert 80 addieren, im w-
                                  ; Register befindet sich nun das Ergebnis
          return
```

**Anmerkung:**

Das temporäre Register TEMP1 wird hier nur zum Zwischenspeichern benötigt, und kann daher auch in anderen Unterprogrammen verwendet werden.

## 5. Nachbauhinweise

### Schritt 1: (Lochraster)-Platinen bestücken

Die Elektronik dieses Projektes wurde auf mehrere kleine Lochrasterplatinen aufgeteilt. Diese entsprechen in etwa der Einteilung bei der Schaltungsbeschreibung. Nur die Leuchtdioden für die Stunden- und Minuten-Sekunden-Anzeige (also das Ziffernblatt) befinden sich nicht auf den Lochrasterplatinen, sondern werden (bei mir) auf einer 25 x 25cm großen durchsichtigen Platte montiert. (z.B. Bastelglas).

Selbstverständlich kann eine eigene Platine entworfen werden, auf der die gesamte Elektronik inklusive „Ziffernblatt“ untergebracht ist.

Das Bestücken einer Platine ist erst dann sinnvoll, wenn alle für diese Platine benötigten Bauteile vorhanden sind. Es sollten generell nur erstklassige und neuwertige Bauteile verwendet werden. Auf Bauteile aus ausgeschlachteten Geräten sollte grundsätzlich verzichtet werden, da ihre Funktionalität nicht gewährleistet ist, und eine unnötige Fehlersuche dadurch vermieden werden kann.

Weiters sollte ausreichend Platz und vor allem ausreichend Zeit für die Bestückung und Verdrahtung der Platinen vorhanden sein.

#### **Controllerplatine:**

Die Bauelemente entsprechend Bild 5.1, der folgenden Reihenfolge, der Stückliste (Anhang C) und dem Schaltplan (Anhang A) bestücken wobei zunächst nur die Bauteile angelötet werden. Diese jedoch noch nicht verdrahten. Das Verdrahten erfolgt erst wenn alle Bauteile angelötet sind. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bearbeitung und Bestückung der Controllerplatine:

- Platine auf eine Größe von 43 x 26 mm zuschneiden und die geschnittenen Seiten abschleifen
- 3,5-mm-Bohrungen für die Montage der Platine auf einer Bodenplatte (4 Bohrungen)
- Drahtbrücke nicht vergessen!
- Widerstände R1, R2 (je 10k) und R3 (270 Ohm): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw.
- IC-Fassung für IC1 (18polig): **Tipp 1:** Obwohl es bei den Fassungen elektrisch gesehen egal ist wie die Fassungen eingelötet sind, sollte man doch die Fassung so einlöten, dass die Kerbe in die richtige Richtung zeigt. Dies erleichtert das spätere Einsetzen der ICs bzw. erleichtert die Arbeit bei einem IC-Tausch. **Tipp 2:** Beim Einlöten der Fassungen sollte man wie folgt vorgehen: Fassung an der einzusetzenden Stelle lagerichtig einsetzen und zunächst nur einen beliebigen Eckpin anlöten. Fassung kontrollieren und eventuell Nachlöten. Sitzt die Fassung ordentlich, den gegenüberliegenden Pin anlöten. Fassung wieder kontrollieren und eventuell nachlöten. Erst wenn Sie mit der Lage der Fassung zufrieden sind, die restlichen Pins anlöten.
- Buchsenleisten: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den

letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.

- Keramikkondensatoren C2 und C3 (je 22pF)
- Keramikkondensator C4 (100nF)
- Elko C1 (10µF/25V): **Achtung:** Polarität beachten!
- Quarz X1 (4,096 MHz)

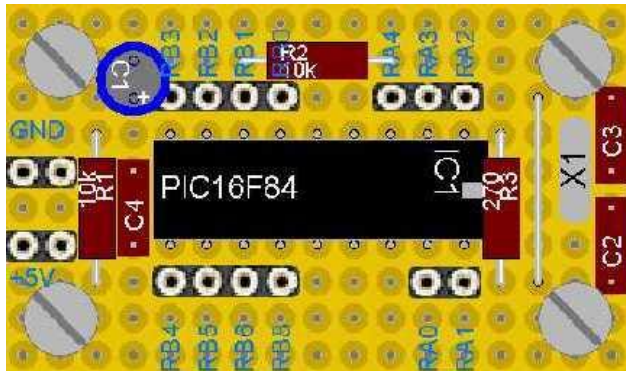


Bild 5.1: Controllerplatine (Bauteilseite)

Nachdem alle Bauteile eingelötet wurden, die Verbindungen auf der Lötseite entsprechend Bild 9 mit einem dünnen Draht bzw. mit isolierten Drähten herstellen. Dabei sollte sorgfältigst gearbeitet werden, und man sollte sich dafür ausreichend Zeit nehmen. Befinden sich Verbindungen direkt nebeneinander, so können diese auch nur mit Lötzinn verbunden werden. **Tipp:** Bild 5.2 ausdrucken, und während dem Verlöten die soeben durchgeführte Verbindung im Ausdruck kennzeichnen. Auf diese Weise können Fehler durch fehlende Verbindungen vermieden werden.

Zum Schluss alle Verbindungen und Lötstellen noch einmal sorgfältig überprüfen. (Auch hierfür sollte man sich ausreichend Zeit nehmen!)

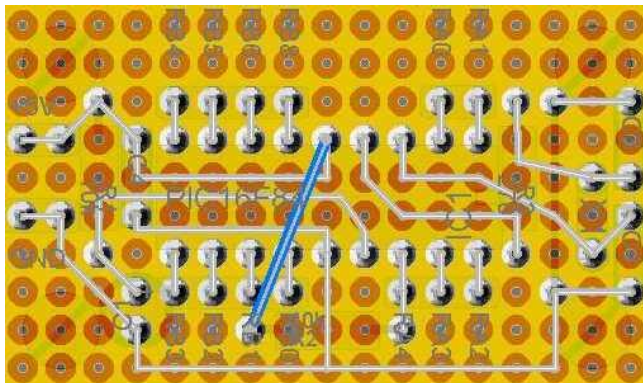


Bild 5.2: Controllerplatine (Lötseite)

### Platine für Anpassschaltung:

Die Bauelemente entsprechend Bild 5.3, der folgenden Reihenfolge, der Stückliste (Anhang C) und dem Schaltplan (Anhang A) bestücken wobei zunächst nur die Bauteile angelötet werden. Diese jedoch noch nicht verdrahten. Das Verdrahten erfolgt erst wenn alle Bauteile angelötet sind. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bearbeitung und Bestückung der Anpassplatine:

- Platine auf eine Größe von 28 x 23 mm zuschneiden und die geschnittenen Seiten abschleifen
- 3,5-mm-Bohrungen für die Montage der Platine auf einer Bodenplatte (4 Bohrungen)



- Widerstände R4, R6 (je 10k) und R5 (1k): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw..
- Buchsenleisten: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.
- Keramikkondensator C5 (100nF)
- Leuchtdiode D73 (3mm, low-current, gelb): **Achtung:** Polarität beachten! Der längere Anschluss ist die Kathode (plus), der kürzere demnach die Anode (minus)
- Transistor T1 (BC547B): **Achtung:** Polarität beachten!
- Stifteleiste für Jumper JP1

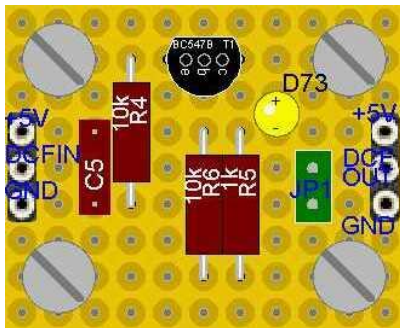


Bild 5.3: Anpassplatine (Bauteilseite)

Nachdem alle Bauteile eingelötet wurden, die Verbindungen auf der Lötseite entsprechend Bild 5.4 mit einem dünnen Draht herstellen. Dabei sollte sorgfältig gearbeitet werden, und man sollte sich dafür ausreichend Zeit nehmen. Befinden sich Verbindungen direkt nebeneinander, so können diese auch nur mit Lötzinn verbunden werden. **Tipp:** Bild 5.4 ausdrucken, und während dem Verlöten die soeben durchgeführte Verbindung im Ausdruck kennzeichnen. Auf diese Weise können Fehler durch fehlende Verbindungen vermieden werden.

Zum Schluss alle Verbindungen und Lötstellen noch einmal sorgfältig überprüfen. (Auch hierfür sollte man sich ausreichend Zeit nehmen!)



Bild 5.4: Anpassplatine (Lötseite)

### Platine für Stundenanzeige:

Die Bauelemente entsprechend Bild 5.5, der folgenden Reihenfolge, der Stückliste (Anhang C) und dem Schaltplan (Anhang A) bestücken wobei zunächst nur die Bauteile angelötet werden. Diese jedoch noch nicht verdrahten. Das Verdrahten erfolgt erst

wenn alle Bauteile angelötet sind. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bearbeitung und Bestückung der Stundenanzeigeplatine:

- Platine auf eine Größe von 49 x 41 mm zuschneiden und die geschnittenen Seiten abschleifen.
- 3,5-mm-Bohrungen für die Montage der Platine auf einer Bodenplatte (4 Bohrungen)
- Widerstände R7 bis R18 (je 33k) und R19 (1k): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw.
- IC-Fassung für IC2 (24polig): **Tipp 1:** Obwohl es bei den Fassungen elektrisch gesehen egal ist wie die Fassungen eingelötet sind, sollte man doch die Fassung so einlöten, dass die Kerbe in die richtige Richtung zeigt. Dies erleichtert das spätere Einsetzen der ICs bzw. erleichtert die Arbeit bei einem IC-Tausch. **Tipp 2:** Beim Einlöten der Fassungen sollte man wie folgt vorgehen: Fassung an der einzusetzenden Stelle lagerichtig einsetzen und zunächst nur einen beliebigen Eckpin anlöten. Fassung kontrollieren und eventuell Nachlöten. Sitzt die Fassung ordentlich, den gegenüberliegenden Pin anlöten. Fassung wieder kontrollieren und eventuell nachlöten. Erst wenn Sie mit der Lage der Fassung zufrieden sind, die restlichen Pins anlöten.
- Buchsenleisten: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.
- Keramik Kondensator C6 (100nF)

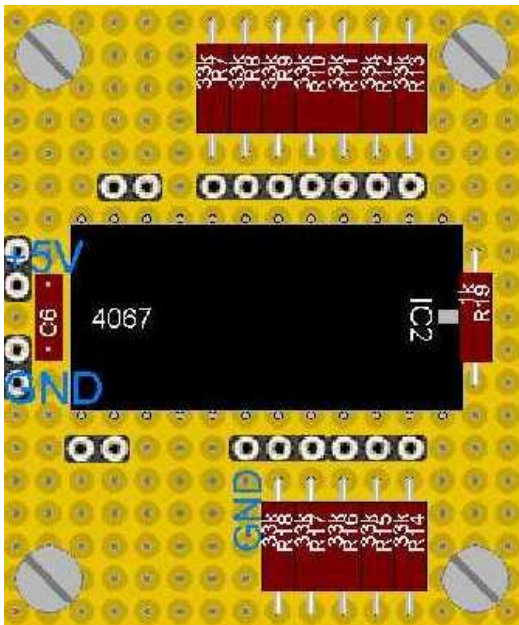


Bild 5.5: Stundenanzeigeplatine (Bauteilseite)

Nachdem alle Bauteile eingelötet wurden, die Verbindungen auf der Lötseite entsprechend Bild 5.6 mit einem dünnen Draht herstellen. Dabei sollte sorgfältig gearbeitet werden, und man sollte sich dafür ausreichend Zeit nehmen. Befinden sich Verbindungen direkt nebeneinander, so können diese auch nur mit Lötzinn verbunden werden. **Tipp:** Bild 5.6 ausdrucken, und während dem Verlöten die soeben

durchgeführte Verbindung im Ausdruck kennzeichnen. Auf diese Weise können Fehler durch fehlende Verbindungen vermieden werden.  
Zum Schluss alle Verbindungen und Lötstellen noch einmal sorgfältig überprüfen. (Auch hierfür sollte man sich ausreichend Zeit nehmen!)



Bild 5.6: Stundenanzeigeplatine (Lötseite)

#### **Platine für Minuten/Sekundenanzeige:**

Die Bauelemente entsprechend Bild 5.7, der folgenden Reihenfolge, der Stückliste (Anhang C) und dem Schaltplan (Anhang A) bestücken wobei zunächst nur die Bauteile angelötet werden. Diese jedoch noch nicht verdrahten. Das Verdrahten erfolgt erst wenn alle Bauteile angelötet sind. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bearbeitung und Bestückung der Minuten-/Sekundenanzeigeplatine:

- Platine auf eine Größe von 41 x 31 mm zuschneiden und die geschnittenen Seiten abschleifen
- 3,5-mm-Bohrungen für die Montage der Platine auf einer Bodenplatte (4 Bohrungen)
- Widerstand R20 (1k): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw.
- IC-Fassungen für IC3 und IC4 (je 16polig): **Tipp 1:** Obwohl es bei den Fassungen elektrisch gesehen egal ist wie die Fassungen eingelötet sind, sollte man doch die Fassung so einlöten, dass die Kerbe in die richtige Richtung zeigt. Dies erleichtert das spätere Einsetzen der ICs bzw. erleichtert die Arbeit bei einem IC-Tausch. **Tipp 2:** Beim Einlöten der Fassungen sollte man wie folgt vorgehen: Fassung an der einzusetzenden Stelle lagerichtig einsetzen und zunächst nur einen beliebigen Eckpin anlöten. Fassung kontrollieren und eventuell Nachlöten. Sitzt die Fassung ordentlich, den gegenüberliegenden Pin anlöten. Fassung wieder kontrollieren und eventuell nachlöten. Erst wenn Sie mit der Lage der Fassung zufrieden sind, die restlichen Pins anlöten.
- Buchsenleisten: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den

letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.

- Keramikkondensatoren C7 und C8 (je 100nF)

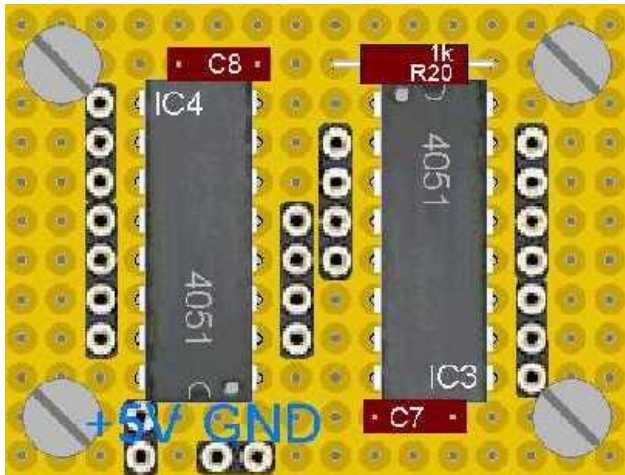


Bild 5.7: Minuten-/Sekundezeigeplatine (Bauteilseite)

Nachdem alle Bauteile eingelötet wurden, die Verbindungen auf der Lötseite entsprechend Bild 5.8 mit einem dünnen Draht herstellen. Dabei sollte sorgfältig gearbeitet werden, und man sollte sich dafür ausreichend Zeit nehmen. Befinden sich Verbindungen direkt nebeneinander, so können diese auch nur mit Lötzinn verbunden werden. **Tipp:** Bild 5.8 ausdrucken, und während dem Verlöten die soeben durchgeführte Verbindung im Ausdruck kennzeichnen. Auf diese Weise können Fehler durch fehlende Verbindungen vermieden werden.

Zum Schluss alle Verbindungen und Lötstellen noch einmal sorgfältig überprüfen. (Auch hierfür sollte man sich ausreichend Zeit nehmen!)

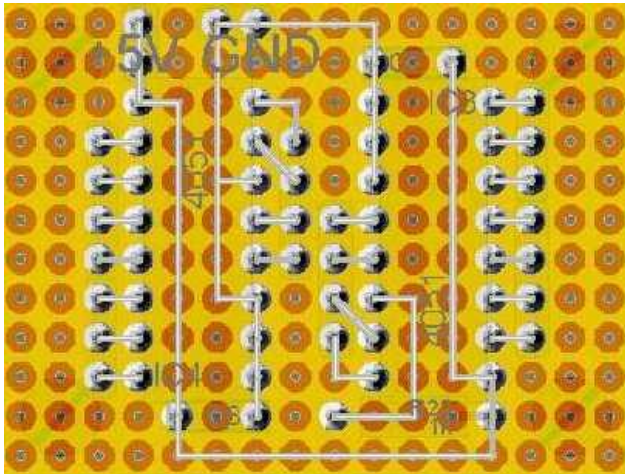


Bild 5.8: Minuten-/Sekundezeigeplatine (Lötseite)

### Stromversorgungsplatine:

Die Bauelemente entsprechend Bild 5.9, der folgenden Reihenfolge, der Stückliste (Anhang C) und dem Schaltplan (Anhang A) bestücken wobei zunächst nur die Bauteile angelötet werden. Diese jedoch noch nicht verdrahten. Das Verdrahten erfolgt erst wenn alle Bauteile angelötet sind. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bearbeitung und Bestückung der Stromversorgungsplatine:

- Platine auf eine Größe von 33 x 18 mm zuschneiden und die geschnittenen Seiten abschleifen
- 3,5-mm-Bohrungen für die Montage der Platine auf einer Bodenplatte (4 Bohrungen)
- Widerstand R21 (1k): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw.
- Buchsenleisten: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.
- Keramikkondensator C10 (100nF)
- Leuchtdiode D74 (3mm, low-current, gelb): **Achtung:** Polarität beachten! (der längere Anschluss ist die Kathode (plus), der kürzere demnach die Anode (minus))
- Spannungsregler IC5 (LP2950): **Achtung:** Polarität beachten!
- Elko C9 (10 $\mu$ F/25V): **Achtung:** Polarität beachten!
- Stiftleiste für Jumper JP2

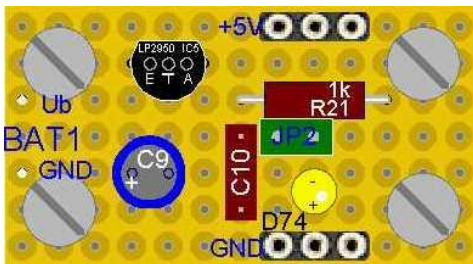


Bild 5.9: Stromversorgungsplatine (Bauteilseite)

Nachdem alle Bauteile eingelötet wurden, die Verbindungen auf der Lötseite entsprechend Bild 5.10 mit einem dünnen Draht herstellen. Dabei sollte sorgfältig gearbeitet werden, und man sollte sich dafür ausreichend Zeit nehmen. Befinden sich Verbindungen direkt nebeneinander, so können diese auch nur mit Lötzinn verbunden werden. **Tipp:** Bild 5.10 ausdrucken, und während dem Verlöten die soeben durchgeführte Verbindung im Ausdruck kennzeichnen. Auf diese Weise können Fehler durch fehlende Verbindungen vermieden werden. Zum Schluss alle Verbindungen und Lötstellen noch einmal sorgfältig überprüfen. (Auch hierfür sollte man sich ausreichend Zeit nehmen!)

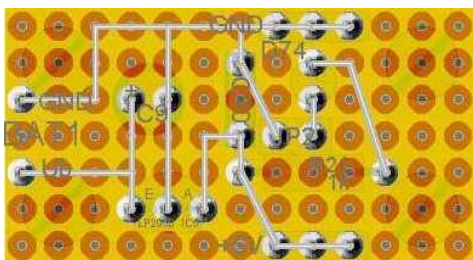


Bild 5.10: Stromversorgungsplatine (Lötseite)

## Schritt 2: Vorbereitung des Ziffernblatts

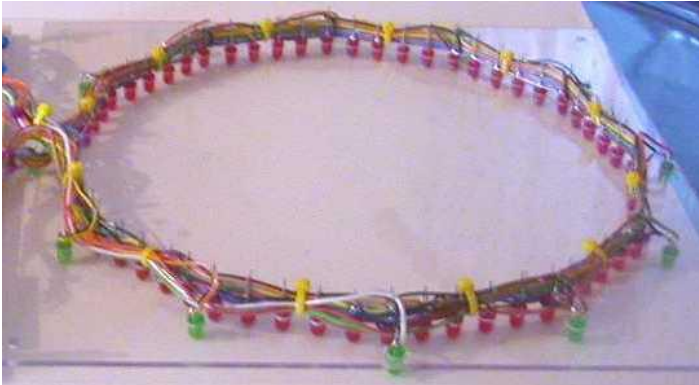


Bild 5.11: Ziffernblatt

Das Ziffernblatt ist der wohl arbeitsintensivste Arbeitsschritt bei diesem Projekt. Hier kann man den eigenen Bedürfnissen freien Lauf lassen. Wem es nicht stört, dass er die Elektronik und die Verdrahtung sieht, kann, so wie ich eine durchsichtige Platte verwenden. Auf eine richtige Glasplatte sollte dabei aber verzichtet werden. Ich habe bei meinem Prototyp ein 25 x 25 cm

großes und 2 mm dicke Bastelglas verwendet. Dieses habe ich gleich bei einem Baumarkt, wo ich es gekauft habe, auf diese Größe zuschneiden lassen. Die 60 Leuchtdioden für die Minuten bzw. Sekundenanzeige bilden bei mir ein regelmäßiges 12-Eck. Hierfür habe ich rot 5mm-Low-current-Typen verwendet. Das 12-Eck hat einen „Durchmesser“ von ca. 21 cm. Für die Stundenanzeige verwendete ich 12 grüne, ebenfalls 5mm-Low-current-Typen. Diese 12 Leuchtdioden befinden sich an den 12 Ecken, etwa 1,2cm von den Roten entfernt. Für die Befestigung der Leuchtdioden genügt es, wenn man 5mm-Löcher in das Bastelglas bohrt, und die Leuchtdioden in diese Löcher steckt bzw. drückt. Ein zusätzliches ankleben der Leuchtdioden war bei meinem Prototyp nicht notwendig.

Für das Verdrahten der Leuchtdioden sollte wirklich ausreichend Zeit vorhanden sein, denn dies ist der arbeitsintensivste Schritt. Es sollte daher auch sehr gründlich gearbeitet werden. Als Verbindungsdrähte habe ich die Drähte eines Flachbandkabels verwendet. Ich habe dazu, grob geschätzt, etwa einen halben Meter eines 40poligen Flachbandkabel benötigt. Die 60 Leuchtdioden für die Minuten-Sekundenanzeige sind im Schaltplan so nummeriert, wie sie auch am Ziffernblatt angeordnet sind. D1 entspricht also der LED, die 1 Minute anzeigt, die weiteren Leuchtdioden folgen im Uhrzeigersinn. D60 zeigt schließlich Minute 60 bzw. Minute 0 an. Für die 12 Leuchtdioden zur Stundenanzeige gilt die im Schaltplan angegebene „Position“ (D61 leuchtet bei 1 Uhr und 13 Uhr, D62 bei 2 Uhr und 14 Uhr usw.).

Für die Verbindung zur Stundenanzeigeplatine bzw. für die Verbindung zur Minuten-/Sekundenanzeigeplatine habe ich zu den Buchsenleisten passende Kontaktstreifen verwendet.

## Schritt 3: Montage und Verdrahtung der (Lochraster)-Platinen

Die im Schritt 1 vorbereiteten Platinen auf eine etwa. 25 x 25 cm große, etwa 4mm dicken (Holz)-Platte montieren, und entsprechend dem Schaltplan (Anhang A) mit isolierten Drähten verdrahten.

## Schritt 4: Test

- Die ICs (IC1 bis IC4) noch **nicht** in die Fassungen einsetzen.
- Zuerst mit einem Multimeter prüfen, ob zwischen Betriebsspannung und Masse kein Kurzschluss herrscht. (Multimeter im Mode „Durchgangstester“ an den Verbindungsleitungen zwischen den einzelnen Platinen). Ist kein Kurzschluss feststellbar, als nächstes an den IC-Sockeln gemäß dem Schaltplan ebenfalls diesen Test durchführen. Eventuell festgestellte Kurzschlüsse müssen natürlich aufgespürt und entfernt werden!
- Als nächstes (mit dem Multimeter) prüfen, ob die +5V und die Masseleitung an allen Platinen vorhanden ist. (Multimeter noch immer im Mode „Durchgangstester“)
- Eine 6V-Spannungsquelle (zum Beispiel ein Steckernetzteil, diese kann auch Spannung größer als 6 Volt liefern) anschließen. **Achtung:** Auf die Polarität achten.
- Spannung an den ICs an den jeweiligen Pins messen. Wird keine oder eine grob abweichende Spannung als 5V gemessen, so liegt ein Bestückungs- und/oder Verdrahtungsfehler vor, welcher unbedingt aufgespürt und beseitigt werden muss.
- Spannung von der Wanduhr nehmen
- **Programmierten** Mikrocontroller PIC16F84 (IC1), sowie IC2 (4067) und IC3 und IC4 (je 4051) im **ausgeschalteten** Zustand einsetzen. **Achtung:** Auf die Polarität achten!
- Spannung wieder anlegen, und die Jumper JP1 und JP2 stecken. Die Leuchtdiode D73 muss leuchten. Nach einer Weile muss auch die Leuchtdiode D74 zu „blinken“ beginnen, wobei die Leuchtdiode länger leuchtet, und nur sehr kurz erlischt.
- Nach einigen Minuten sollte dann die richtige Zeit angezeigt werden, und der „Sekundenzeiger“ sollte im Sekundentakt im „Kreis“ wandern.
- Die beiden Jumper JP1 und JP2 werden nun nicht mehr benötigt und können entfernt werden

Geschafft! Das folgende Bild zeigt die fertige Wanduhr im Betrieb.

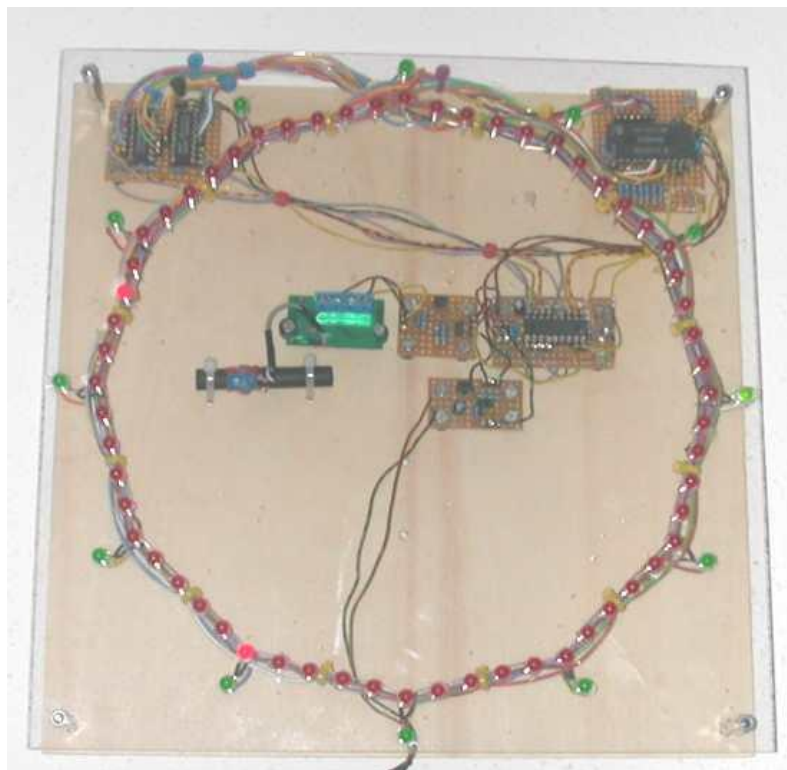
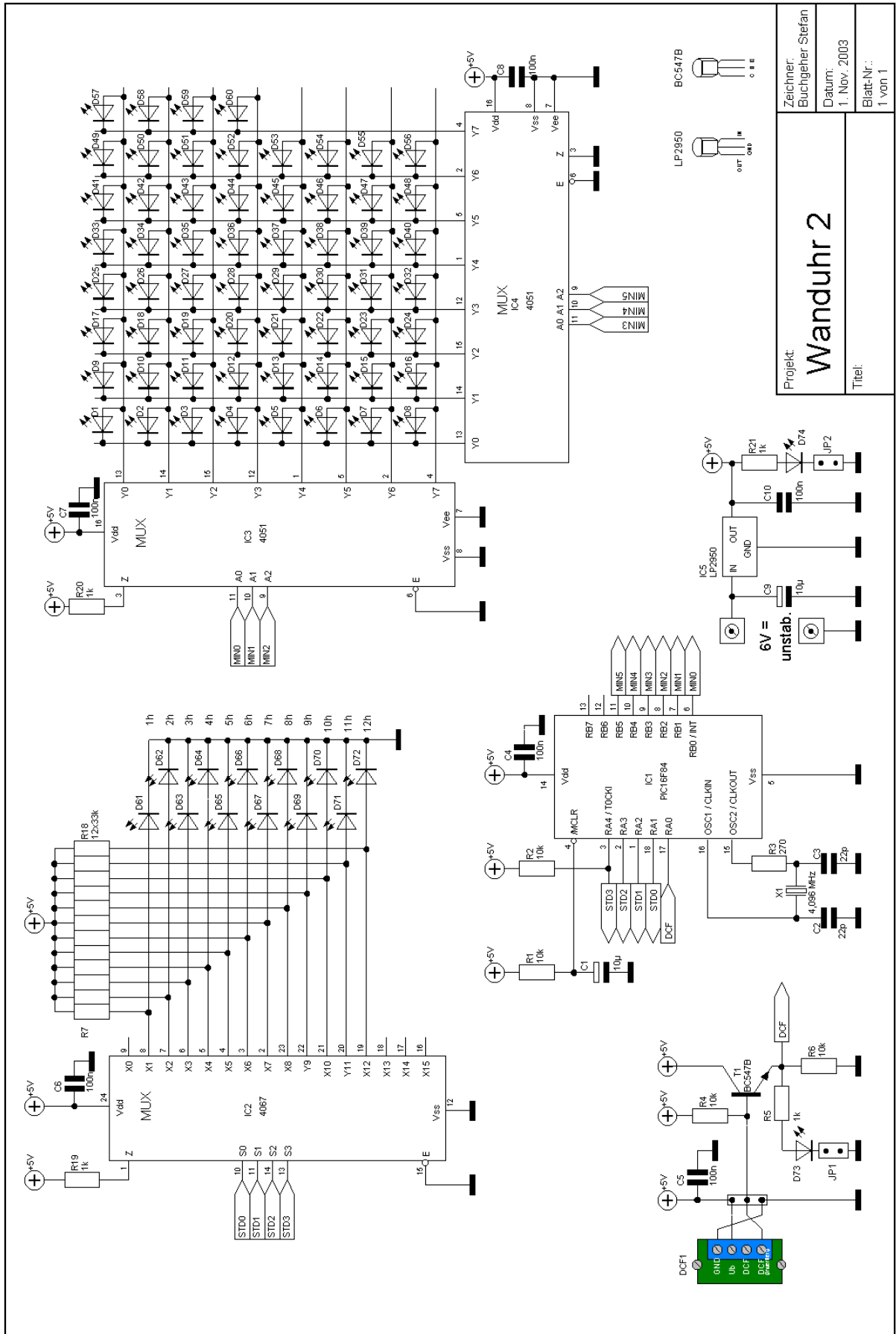


Bild 5.12: Wanduhr in Betrieb

# Anhang A: Schaltplan



Zeichner: Buchgeiger-Stefan	
Datum: 1. Nov. 2003	
Blatt-Nr.: 1 von 1	
<h2>Wanduhr 2</h2>	
Titel	



## Anhang B: Listing des PIC-Mikrocontroller (wanduhr2.asm)

```

;*****
; ** Analog/Digitale-Wanduhr 2 **
; **
; ** Wanduhr mit PIC16F84 und 72 LEDs, wobei 60 LEDs in Form eines regelm. 12-Ecks angeordnet **
; ** sind. Diese zeigen die Minuten bzw. die Sekunden an. Die restlichen 12 LEDs sind auf die **
; ** 12 Ecken aufgeteilt und zeigen die aktuelle Stunde an. **
; **
; ** Hardware: **
; ** + Mikrocontroller: PIC16F84, Takt (Quarz 4,096 MHz) **
; ** + DCF-Modul (mit Anpassschaltung) **
; ** + Ansteuerelektronik für die LEDs **
; **
; ** Die Aufgabe der PIC-Software besteht bei der DCF-Dekodierung aus mehreren Teilaufgaben: **
; ** + In kurzen Zeitabständen (hier alle 4 ms) das vom DCF-Empfangsmodul erzeugte Signal **
; ** abtasten. Daraus entweder ein Low, High oder eine neue Minuten ermitteln. Kann aus **
; ** den Abtastungen keines dieser drei Fälle ermittelt werden, so handelt es sich um **
; ** einen Telegrammfehler. **
; ** + Die empfangenen Low- und High-Pegel zwischenspeichern **
; ** + Bei jeder neuen Minute aus den gespeicherten Werten entsprechend die Zeit- und **
; ** Datumswerte zusammensetzen und in entsprechenden Registern sichern. **
; ** + Die soeben gewonnen Zeit- und Datumsinformationen mit den Zeit- und Datums- **
; ** informationen der vorhergehenden Minute vergleichen. Die Zeit- und Datumswerte der **
; ** soeben empfangenen Minute sind nur dann gültig, wenn sie sich um maximal 1 von den **
; ** Zeit- und Datumsinformationen der vorhergehenden Minute unterscheiden. **
; ** + Parallel zur DCF-Uhr eine reine Softwareuhr erzeugen. Diese Softwareuhr ist **
; ** gültig, wenn keine korrekte Zeit vom DCF-Empfangsmodul empfangen wird. Mit anderen **
; ** Worten. Die Softwareuhr wird ständig mit einer gültig dekodierten "DCF-Zeit" **
; ** synchronisiert. Dies gilt natürlich auch für die Datumsinformationen. **
; **
; ** Das DCF-Dekodierverfahren: **
; ** Der DCF-I/O-Eingang, an dem das DCF-Empfangsmodul (mit einer Anpassschaltung) ange- **
; ** schlossen ist, wird zyklisch (ca. alle 4 ms) vom Unterprogramm DCFROUTINE abgefragt. **
; ** Als Zeitbasis für den 4-ms-Takt dient der Timer-0-Interrupt. Dieser kann je nach **
; ** Anwendung entweder so eingestellt werden, dass er alle 4 ms einen Interrupt auslöst, **
; ** oder er wird so eingestellt dass er z. B. alle 500us oder alle 1ms einen Interrupt **
; ** erzeugt und ein Zählregister zählt die notwendige Anzahl an Interruptaufrufen bis **
; ** zu einer Zeitbasis von 4 ms. Hier wird der Einfachheit halber die erstere Variante **
; ** verwendet. Der Aufruf des Unterprogramms DCFROUTINE erfolgt hier aber nicht von der **
; ** ISR (Interrupt-Service-Routine), sondern vom Hauptprogramm. Die ISR setzt nur alle **
; ** 4ms ein Flag zur Kennzeichnung, dass das Hauptprogramm das Unterprogramm DCFROUTINE **
; ** aufrufen soll. **
; **
; ** Die Aufgabe des Unterprogramms DCFROUTINE besteht darin, aus dem vom DCF-Empfangs- **
; ** modul empfangenen Datenstrom die Informationen LOW, HIGH und den Minutenwechsel (also **
; ** das Ausbleiben der 59. Sekunde) zu dekodieren. für diese Informationen befinden sich **
; ** im DCF-Statusregister (DCFSTATUS) entsprechende Flags. **
; ** Weiters beinhaltet das Register DCFSTATUS noch den Zustand des DCF-Eingangs 4 ms vor **
; ** dem Aufruf des Unterprogramms DCFROUTINE. Mit Hilfe des aktuellen Zustands und des **
; ** Zustands vor 4 ms kann nun erkannt werden, ob sich der Pegel am DCF-Eingang geändert **
; ** hat. **
; ** Das Hauptprogramm prüft nun ständig die Flags DCFNEUESEK (wird zusätzlich zu **
; ** DCFLOW bzw. DCFHIGH gesetzt, wenn eine gültige Sekunde empfangen wurde) und **
; ** DCFNEUEMIN, und ruft die zugehörigen Unterprogramme DCFUPSEKUNDE bzw. DCFUPMINUTE **
; ** auf. Die Aufgabe des Unterprogramms DCFUPSEKUNDE ist es, die empfangenen Low- und **
; ** High-Pegel in den Register DCFTELEGRAMM1 bis DCFTELEGRAMM8 zwischenspeichern. **
; ** Die Aufgabe des Unterprogramms DCFUPMINUTE ist dagegen etwas umfangreicher: Zunächst **
; ** die Zeit und das Datum aus den in den Registern DCFTELEGRAMM1 bis DCFTELEGRAMM8 **
; ** zwischengespeicherten Werten zusammensetzen. Anschließend die soeben gewonnen Zeit- **
; ** und Datumsinformationen mit den Zeit- und Datumsinformationen der vorhergehenden **
; ** Minute vergleichen. Die Zeit- und Datumswerte der soeben empfangenen Minute sind nur **
; ** dann gültig, wenn sie sich um maximal 1 von den Zeit- und Datumsinformationen der **
; ** vorhergehenden Minute unterscheiden. **
; **
; ** Parallel zur Dekodierung des DCF-Eingangs erzeugt das Unterprogramm INNEREUHR eine **
; ** reine Softwareuhr. Für diesen Zweck ist eine genaue 1-Sekunden-Zeitbasis notwendig. **
; ** Diese Zeitbasis wird von der schon erwähnten Timer-0-ISR (Interrupt Service Routine) **
; ** zusätzlich zur 4ms-Zeitbasis erzeugt. Auch für diese Zeitbasis wird in der ISR ein **
; ** entsprechendes Flag gesetzt, und das Hauptprogramm ruft das Unterprogramm INNEREUHR **
; ** auf, wenn dieses Flag gesetzt ist. Diese zusätzliche Softwareuhr mag auf dem ersten **
; ** Blick unnötig erscheinen. Es hat sich aber gezeigt, dass ein DCF-Empfang oft auch **
; ** über eine längere Zeit nicht möglich ist. In diesem Fall würde die Uhr "stehen". **
; ** Diese Zeit wird mit einer zusätzlichen Softwareuhr so überbrückt, dass der **
; ** Betrachter der Uhr davon nichts bemerkt. **
; **

```

## Analog/Digitale Wanduhr 2

```

** Entwickler: Buchgeher Stefan **
** Entwicklungsbeginn der Software: 21. Oktober 2003 **
** Funktionsfähig seit: 23. Oktober 2003 **
** Letzte Bearbeitung: 26. Februar 2004 **
*****

```

List    p=PIC16F84

```

***** Register (in Registerseite 0) *****
TMRO        equ     1            ;Timer0-Register
PC          equ     2            ;Programmzähler
STAT        equ     3            ;Statusregister
PORTA       equ     5            ;PortA-Register
PORTB       equ     6            ;PortB-Register
INTCON      equ     0B          ;Interrupt-Register

***** Register (in Registerseite 1) *****
OPTREG      equ     1            ;OPTION-Register
TRISA       equ     5            ;Richtungsregister PortA
TRISB       equ     6            ;Richtungsregister PortB

***** Eigene Register (in Registerseite 0) *****
ISR_STAT_TEMP equ     0C            ;Zwischenspeicher des Statusregister der ISR
ISR_w_TEMP    equ     0D            ;Zwischenspeicher des Arbeitsregister der ISR
FLAGISRHP    equ     0E            ;beinhaltet Botschaftsflags ISR -> HP
ZAEHLERISR1SEK equ     0F            ;Zählregister für 1-Sekunden-Zeitbasis
DCFSTATUS    equ     10            ;DCF-Statusregister
DCFPULS      equ     11            ;Impulszeit des DCF-Impulses
DCFPAUSE     equ     12            ;Zeit zwischen 2 DCF-Impulsen
DCFTELEGRAMM1 equ     13            ;in diese Register wird das DCF-Telegramm
DCFTELEGRAMM2 equ     14            ; im Sekunden-Takt eingelesen. Beginnt eine
DCFTELEGRAMM3 equ     15            ; neue Minute, so werden diese Register
DCFTELEGRAMM4 equ     16            ; ausgewertet, und in die entsprechenden
DCFTELEGRAMM5 equ     17            ; DCF-Zeit-Datums-Register kopiert (z.B. in das
DCFTELEGRAMM6 equ     18            ; Register DCFMINBCD)
DCFTELEGRAMM7 equ     19            ;Reserve
DCFTELEGRAMM8 equ     1A            ;Reserve
DCFMINBCD    equ     1B            ;Beinhaltet die aktuelle dekodierte Minute (BCD-Format)
DCFSTDBCD    equ     1C            ;Beinhaltet die aktuelle dekodierte Stunde (BCD-Format)
DCFTAGBCD    equ     1D            ;Beinhaltet den aktuell dekodierten Tag (BCD-Format)
DCFMONBCD    equ     1E            ;Beinhaltet den aktuell dekodierten Monat (BCD-Format)
DCFJAHRBCD   equ     1F            ;Beinhaltet das aktuell dekodierte Jahr (BCD-Format)
DCFWOCHENTAG equ     20            ;Beinhaltet den aktuell dekodierten Wochentag
              ; (1=Montag, 2=Dienstag, usw., 7=Sonntag)
DCFZUSATZINFOS equ     21            ;Beinhaltet Zusatzinformationen (Sommer/Winterzeit,
              ; Reserveantenne usw.)
DCFMINALT    equ     22            ;In den Registern DCFxxxALT befinden sich die Uhrzeit-
DCFSTDALT    equ     23            ; und Datumsinfos der vorangegangenen Minute. Diese
DCFTAGALT    equ     24            ; werden zur Überprüfung des neuen DCF-Telegramms
DCFMONALT    equ     25            ; benötigt
DCFJAHRALT   equ     26
DCFWTAGALT   equ     27
UHRSEKUNDE   equ     28            ;Sekundenzähler der inneren quartzgesteuerten Uhr
UHRMINUTE    equ     29            ;Minutenzähler der inneren quartzgesteuerten Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
UHRSTUNDE    equ     2A            ;Stundenzähler der inneren quartzgesteuerten Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
DATUMWTTAG   equ     2B            ;Wochentagszähler der inneren quartzgest. Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
DATUMTAG     equ     2C            ;Tageszähler der inneren quartzgesteuerten Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
DATUMMONAT   equ     2D            ;Monatszähler der inneren quartzgesteuerten Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
DATUMJAHR    equ     2E            ;Jahreszähler der inneren quartzgesteuerten Uhr, wird
              ; mit jedem gültigen DCF-dekod. Wert synchronisiert
FLAGS        equ     30
TEMP1        equ     4A            ;allgemeines Hilfsregister 1
TEMP2        equ     4B            ;allgemeines Hilfsregister 2

***** Bits in Registern der Registerbank 0 *****
;Register STAT
C            equ     0            ;Carrybit im Statuswort-Register
Z            equ     2            ;Zerobit im Statuswort-Register
RP0          equ     5            ;Seitenauswahlbit im Statuswort-Register

```

## Analog/Digitale Wanduhr 2

```

;Register INTCON
TOIF      equ    2          ;TMR0-Interruptflag im INTCON-Register
GIE       equ    7          ;Interruptfreigabeflag

;***** Bits in den eigenen Registern der Registerbank 0 *****
;Register FLAGISRHP
FLAG4MSEK equ    0
FLAG1SEK  equ    1

;Register DCFSTATUS
DCFPORTNEU equ    0          ;Akt. Zustand DCF-Porteingang
DCFPORTALT equ    1          ;Vorhergehender Zustand am DCF-Porteingang
DCFNEUESEK equ    2          ;gesetzt, wenn neue Sekunde begonnen
DCFNEUEMIN equ    3          ;gesetzt, wenn neue Minute begonnen
DCFLOW     equ    4          ;gesetzt, wenn Low-Signal erkannt
DCFHIGH    equ    5          ;gesetzt, wenn High-Signal erkannt
DCFFEHLER equ    6          ;gesetzt, wenn ein Fehler erkannt
DCFSYNC    equ    7          ;gesetzt, wenn Uhr mit DCF synchronisiert

;Register DCFZUSATZINFOS
DCFRESANT  equ    0          ;Reserveantenne
DCFSOMWIN  equ    1          ;Vorankündigung der Sommer/Winterzeit-Umstellung
DCFSOMMER  equ    2          ;Sommerzeit
DCFWINTER  equ    3          ;Winterzeit
DCFSCHALTSEK equ    4       ;Vorankündigung einer zusätzliche Schaltsekunde

;***** Portbelegung *****
;Port A
DCFINPORT  equ    PORTA
DCFINTRIS  equ    TRISA
DCFIN      equ    0

;***** Konstanten *****
KONSTISR1SEK equ    .250
DCFLOWMIN    equ    .20
DCFLOWMAX    equ    .30
DCFHIGHMIN   equ    .40
DCFHIGHMAX   equ    .60

;***** Ziele der Registeroperationen *****
w      equ    0          ;w ist Zielregister
f      equ    1          ;f ist Zielregister

;***** Makros *****
;Umschalten zu Registerbank 0
bank0  MACRO
        bcf    STAT,RP0
        ENDM

;Umschalten zu Registerbank 1
bank1  MACRO
        bsf    STAT,RP0
        ENDM

;Bit invertieren (entspricht dem Befehl btg bei PIC17/18)
btg    MACRO FILE,BIT
        local SETBIT
        btfs  FILE,BIT
        goto SETBIT
        bcf  FILE,BIT
        btfs  FILE,BIT
        bsf  FILE,BIT
SETBIT ENDM

;***** Konfigurations-Bits *****
_lp_osc equ    h'3FFC'
_xt_osc equ    h'3FFD'
_hs_osc equ    h'3FFE'
_rc_osc equ    h'3FFF'

_wdt_off equ    h'3FFB'
_wdt_on  equ    h'3FFF'

```

## Analog/Digitale Wanduhr 2

```

_pwrtoff equ h'3FFF'
_pwrton equ h'3FF7'

_cpoff equ h'3FFF'
_cp_on equ h'000F'

__config _hs_osc & _wdt_off & _pwrtoff & _cp_off

ORG 0x000
goto Beginn
ORG 0x004
goto ISR

;***** Tabellen *****

;*****
;** Tabelle TABSTUNDEN: **
;** Ausgabetabelle der Stunde am Port A (Bits 1-4) **
;*****
TABSTUNDEN addwf PC,f
dt .24
dt .2,.4,.6,.8,.10,.12,.14,.16,.18,.20,.22,.24
dt .2,.4,.6,.8,.10,.12,.14,.16,.18,.20,.22
dt .0,.0,.0,.0,.0,.0,.0,.0,.0,.0

;*****
;** Tabelle TABMINUTENSEKUNDEN: **
;** Ausgabetabelle der Minuten und Sekunden am Port B (Bits 0-5) **
;*****
TABMINUTENSEKUNDEN addwf PC,f
dt .59
dt .0,.1,.2,.3,.4,.5,.6,.7,.8,.9,.10,.11,.12
dt .13,.14,.15,.16,.17,.18,.19,.20,.21,.22,.23,.24
dt .25,.26,.27,.28,.29,.30,.31,.32,.33,.34,.35,.36
dt .37,.38,.39,.40,.41,.42,.43,.44,.45,.46,.47,.48
dt .49,.50,.51,.52,.53,.54,.55,.56,.57,.58
dt .60,.60,.60,.60,.60

;***** ISR - Timer0 *****

;*****
;** Interrupt Service Routine: **
;** ** **
;** Aufruf: **
;** alle 4 ms **
;** ** **
;** Aufgaben: **
;** + w-Register (=Arbeitsregister) und Status-Register zwischenspeichern (PUSH) **
;** + Zeitbasis für 4ms und 1 Sekunde erzeugen **
;** + Das Timer-Interrupt-Flag T0IF wieder löschen **
;** + w-Register (=Arbeitsregister) und Statusregister wiederherstellen (POP). **
;*****
ISR
PUSH movwf ISR_w_TEMP ;w-Register retten
swapf STAT,w ;Statusregister
movwf ISR_STAT_TEMP ; retten

;Beginn der eigentlichen ISR-Routine
bsf FLAGISRHP,FLAG4MSEK ;Botschaftsflag setzen

decfsz ZAEHLERISR1SEK,f ;Zählregister für 1-Sekunden-Zeitbasis um 1
; vermindern

goto ISRWEITER1
bsf FLAGISRHP,FLAG1SEK ;Botschaftsflag setzen
movlw KONSTISR1SEK ;Zählregister für den Sekundentakt mit
movwf ZAEHLERISR1SEK ; der Konstanten KONSTISR1SEK laden

ISRWEITER1
;Ende der eigentlichen ISR-Routine
ISRfertig bcf INTCON,T0IF ;T0-Interruptflag löschen

POP swapf ISR_STAT_TEMP,w ;Status-Register
movwf STAT ; und

```

## Analog/Digitale Wanduhr 2

```

swapf   ISR_w_TEMP,f           ; w-Register
swapf   ISR_w_TEMP,w           ; wieder herstellen

retfie

;***** Unterprogramme *****

;***** Initialisierung des Prozessor: *****
;**   + TMR0-ISR soll alle 4 ms aufgerufen werden, dazu Vorteiler mit 16 laden (Bei einem   **
;**   4,096-MHz-Quarz)                                               **
;**   + Ports: Port A: Ausgänge (Stundenanzeige)                       **
;**   Port B: RB0 bis RA5: Ausgänge (Minutenanzeige)                 **
;**   RB6, RB7: Reserve                                               **
;**   (Der DCF-Eingang wird separat mit dem Befehl bsf DCFINTRIS,DCFIN als   **
;**   Eingang definiert)                                             **
;**   + Zählregister für den Sekundentakt vorbelegen                   **
;**   + DCF-Statusregister (DCFSTATUS) initialisieren (Fehlerflag setzen, alle anderen   **
;**   Flags löschen)                                                 **
;**   + Pegel von DCFIN einlesen und im Flag DCFPORTNEU des Register DCFSTATUS sichern   **
;**   + Anzeige löschen                                              **
;**   + diverse Register vorbelegen (initialisieren)                 **
;*****
INIT     clrf     TMR0           ;Timer0 auf 0 voreinstellen

        bank1          ;Registerseite 1
        movlw   b'00000011'    ;interner Takt, Vorteiler = 16 an TMR0
        movwf   OPTREG
        clrf    TRISA         ;Port A: Ausgänge (Stundenanzeige)

        movlw   b'00000000'    ;Port B: Ausgang
        movwf   TRISB

        bsf     DCFINTRIS,DCFIN ;DCFIN als Eingang definieren
        bank0          ;Registerseite 0

        movlw   KONSTISR1SEK   ;Zählregister für den Sekundentakt mit
        movwf   ZAEHLERISR1SEK ; der Konstanten KONSTISR1SEK laden

        clrf    DCFSTATUS      ;DCF-Statusregister (DCFSTATUS) initialisieren
        bsf     DCFSTATUS,DCFHEHLER ; Fehlerflag setzen, alle anderen Flags
        ; löschen

        btfsc   DCFINPORT,DCFIN ;Pegel von DCFIN einlesen und im Flag
        bsf     DCFSTATUS,DCFPORTNEU ; DCFPORTNEU des Register DCFSTATUS sichern

        movlw   b'11110'      ;Anzeige löschen, dazu Werte am Port A und
        movwf   PORTA         ; am Port B ausgeben, die keiner Uhrzeit
        movlw   b'00111111'   ; entsprechen
        movwf   PORTB

        clrf    UHRSEKUNDE    ;Sekundenzähler initialisieren (mit 0)
        clrf    UHRMINUTE     ;Minutenzähler initialisieren (mit 0)
        clrf    UHRSTUNDE     ;Stundenzähler initialisieren (mit 0)
        clrf    DATUMWTTAG    ;Wochentagzähler initialisieren (mit 0)
        clrf    DATUMTTAG     ;Tageszähler initialisieren (mit 0)
        clrf    DATUMMONAT    ;Monatszähler initialisieren (mit 0)
        clrf    DATUMJAHR     ;Jahreszähler initialisieren (mit 0)

        movlw   .99
        movwf   DCFMINALT     ;Vergleichsregister initialisieren (mit 99)
        movwf   DCFSTDALT     ;Vergleichsregister initialisieren (mit 99)
        movwf   DCFWTAGALT    ;Vergleichsregister initialisieren (mit 99)
        movwf   DCFMONALT     ;Vergleichsregister initialisieren (mit 99)
        movwf   DCFJAHRALT    ;Vergleichsregister initialisieren (mit 99)
        movwf   DCFWTAGALT    ;Vergleichsregister initialisieren (mit 99)
        return

;***** DCF Routinen *****

;*****
;** DCFROUTINE: **
;** **
;** Das Unterprogramm DCFROUTINE ist für die DCF-Dekodierung die wichtigste Komponente. **
;** **
;** Diese Routine wird ca. alle 4 ms aufgerufen **
;** **

```

## Analog/Digitale Wanduhr 2

```

** Aufgabe und Vorgehensweise: **
** Die Aufgabe des Unterprogramms DCFROUTINE besteht darin aus den Abtastungen heraus- **
** zufinden, ob ein Low, ein High oder ein Minutenwechsel gesendet wurde. Dazu wird bei **
** jedem Aufruf des Unterprogramms DCFROUTINE entweder das Zählregister DCFPULS oder **
** das Zählregister DCFPAUSE um 1 erhöht (inkrementiert), wenn sich der Pegel vom DCF- **
** Eingang zum vorhergehenden Aufruf dieses Unterprogramms nicht verändert hat. Ist **
** der DCF-Eingang Low (also logisch 0), so wird das Zählregister DCFPULS um 1 erhöht, **
** ist der DCF-Eingang High (also logisch 1), so wird das Zählregister DCFPAUSE um 1 **
** erhöht. Beide Zählregister können maximal den Wert 255 aufnehmen, was einer Zeit **
** von 1020 ms (oder 1,02 Sekunden) entspricht (=255 * 4ms). **
** Unterscheidet sich der aktuelle Pegel des DCF-Eingangs mit dem Pegel vom vorher- **
** gehenden Aufruf, so spricht man von einer Flanke, wobei hier zwischen einer fallenden **
** und einer steigenden Flanke unterschieden werden muss. Bei einer fallenden Flanke **
** (der aktuelle Pegel des DCF-Eingangs ist logisch 0), erfolgt Zunächst eine Aus- **
** wertung des Zählregisters DCFPULS. Beinhaltet dieses Zählregister einen Wert **
** zwischen den Konstanten DCFLOWMIN und DCFLOWMAX, so wurde ein LOW des DCF-Telegramms **
** ermittelt. Im DCF-Statusregister (DCFSTATUS) werden daher die Flags DCFLOW und **
** DCFNEUESEK gesetzt. Das Flag DCFNEUESEK dient als Zeichen einer neu empfangenen und **
** gültigen Sekunde. Anschließend wird das Zählregister DCFPULS gelöscht. Beinhaltet **
** das Zählregister DCFPULS einen Wert zwischen den Konstanten DCFHIGHMIN und **
** DCFHIGHMAX, so wurde ein HIGH des DCF-Telegramms ermittelt. Im DCF-Statusregister **
** (DCFSTATUS) werden daher die Flags DCFHIGH und DCFNEUESEK gesetzt. Das Flag **
** DCFNEUESEK dient auch hier als Zeichen einer neu empfangenen und gültigen Sekunde, **
** und auch das Zählregister DCFPULS wird Anschließend gelöscht. Beinhaltet das **
** Zählregister DCFPULS einen Wert der weder zwischen den Konstanten DCFLOWMIN und **
** DCFLOWMAX noch zwischen den Konstanten DCFHIGHMIN und DCFHIGHMAX liegt, so handelt es **
** sich um einen Übertragungsfehler. In diesem Fall wird das Fehlerflag (DCFFEHLER) im **
** DCF-Statusregister (DCFSTATUS) gesetzt und auch das Zählregister DCFPULS wird an- **
** schließend gelöscht. Das Flag DCFNEUESEK wird hier aber nicht gesetzt, da es sich in **
** diesem Fall um keine gültige Sekunde handelt. Das Fehlerflag wird erst bei der **
** Erkennung einer neuen Minute wieder gelöscht. Achtung: Solange das Fehlerflag ge- **
** setzt ist erfolgt keine Auswertung des Zählregisters DCFPULS. Dieses Zählregister **
** wird aber dennoch bei jeder fallenden Flanke zurückgesetzt. **
** Bei einer steigenden Flanke (der aktuelle Pegel des DCF-Eingangs ist logisch 1) wird **
** das Zählregister DCFPAUSE gelöscht. **
** In der 59. Sekunde erfolgt keine Absenkung des Trägers. Da es also in der 59. **
** Sekunde keine fallende Flanke gibt, gibt es auch keine steigende Flanke. Das Zähl- **
** register DCFPAUSE läuft über, da es nur einen Zählbereich von 0 bis 255 besitzt. **
** Dieser Überlauf wird hier ausgenützt. Zur Bestimmung einer neuen Minute bzw. zur **
** Bestimmung des Telegrammbeginns. Ist das Fehlerflag gesetzt, so wird es nun ge- **
** löscht. War es nicht gesetzt, so wird das Flag DCFNEUEMIN im DCF-Statusregister **
** (DCFSTATUS) gesetzt. **
** **
** Anmerkung: **
** Die temporären Register TEMP1 und TEMP2 werden hier nur als Hilfsregister benötigt. **
** Sie können daher auch in anderen Unterprogrammen verwendet werden. **
** *****
DCFROUTINE    bcf      DCFSTATUS,DCFPORTALT ;DCFSTATUS,DCFPORTNEU -> DCFSTATUS,DCFPORTALT
              btfsc   DCFSTATUS,DCFPORTNEU
              bsf     DCFSTATUS,DCFPORTALT

              movf   DCFINPORT,w
              movwf  TEMP1
              bcf    DCFSTATUS,DCFPORTNEU ;DCFIN -> DCFSTATUS,DCFPORTNEU, TEMP1,DCFPORTALT
              btfsc  TEMP1,DCFIN
              bsf    DCFSTATUS,DCFPORTNEU
              clrf   TEMP2
              btfsc  TEMP1,DCFIN
              bsf    TEMP2,DCFPORTALT

              movf   DCFSTATUS,w
              xorwf  TEMP2,f

              btfss  TEMP2,DCFPORTALT ;Hat sich am Pin DCFIN der Pegel geändert?
              goto   DCFINCPULSPAUSE ;nein: Puls- bzw. Pausenzähler um 1 erhöhen
              btfss  DCFSTATUS,DCFPORTNEU ;ja: -> steigende oder fallende Flanke?
              goto   DCFHLFLANKE ;fallende Flanke
DCFHLFLANKE  clrf    DCFPAUSE ;steigende Flanke: Zählregister DCFPAUSE
              goto   DCFROUTINEFERTIG ; löschen und UP verlassen

              ;Fallende Flanke
DCFHLFLANKE  btfss  DCFSTATUS,DCFFEHLER ;Fehlerflag gesetzt:
              goto   DCFPLAUSIBEL ;nein: Ermittlung, ob die ermittelte Pulszeit
              ; einem HIGH oder einem LOW entspricht
              ; (Plausibilitätsprüfung)
              goto   DCFPULSLOESCHEN ;ja: Puls-Zähler löschen

DCFPLAUSIBEL
              ;Prüfen, ob ein High empfangen wurde
DCFCHKHIGH  movlw   DCFHIGHMIN ;unteren High-Grenzwert laden

```

## Analog/Digitale Wanduhr 2

```

subwf DCFPULS,w
btfss STAT,C ;Grenzwert < DCFPULS ?
goto DCFCHECKLOW ;nein: kein HIGH-Pegel, Prüfen, ob LOW
movlw DCFHIGHMAX ;ja: DCFPULS mit oberen HIGH-Grenzwert
; vergleichen

subwf DCFPULS,w
btfsc STAT,C ;DCF PULS < Grenzwert ?
goto DCFCHECKLOW ;nein: es handelt sich um kein HIGH-Signal
; Prüfen, ob ein LOW empfangen wurde
bcf DCFSTATUS,DCFLOW ;ja: Es wurde ein High-Signal empfangen,
; das Flag DCFLOW löschen,
bsf DCFSTATUS,DCFHIGH ; das Flag DCFHIGH setzen,
bsf DCFSTATUS,DCFNEUESEK ; das Flag DCFNEUESEK setzen
goto DCFPULSLOESCHEN ; und Puls-Zähler löschen

;Prüfen, ob ein Low empfangen wurde
DCFCHECKLOW movlw DCFLOWMIN ;unteren Low-Grenzwert laden
subwf DCFPULS,w
btfss STAT,C ;Grenzwert < DCFPULS ?
goto DCFSETFEHLER ;nein: kein LOW-Pegel, Empfangsfehler
movlw DCFLOWMAX ;ja: DCFPULS mit oberen LOW-Grenzwert
; vergleichen

subwf DCFPULS,w
btfsc STAT,C ;DCF PULS < Grenzwert ?
goto DCFSETFEHLER ;nein: es handelt sich um kein LOW-Signal
; -> Empfangsfehler
bcf DCFSTATUS,DCFHIGH ;ja: Es wurde ein LOW-Signal empfangen,
; das Flag DCFLOW setzen,
bsf DCFSTATUS,DCFLOW ; das Flag DCFHIGH löschen,
bsf DCFSTATUS,DCFNEUESEK ; das Flag DCFNEUESEK setzen
goto DCFPULSLOESCHEN ; und den Puls-Zähler löschen

;Empfangsfehler
DCFSETFEHLER bsf DCFSTATUS,DCFNEUESEK ;Bei einem Empfangsfehler
bcf DCFSTATUS,DCFHIGH ; Fehlerflag (DCFNEUESEK) setzen,
bcf DCFSTATUS,DCFLOW ; die Flags DCFLOW und DCFHIGH löschen,
goto DCFPULSLOESCHEN ; und den Puls-Zähler löschen

;Puls- oder Pausenzähler erhöhen
DCFINCPULSPAUSE btfsc DCFSTATUS,DCFPORTNEU
goto DCFINCPAUSE

DCFINCPULS incf DCFPULS,f ;Puls-Zähler um 1 erhöhen
goto DCFROUTINEFERTIG

DCFINCPAUSE incfsz DCFPAUSE,f ;Pausezähler erhöhen und auf Überlauf
; Prüfen
goto DCFROUTINEFERTIG ;kein Überlauf: Unterprogramm verlassen

btfsc DCFSTATUS,DCFNEUESEK ;Überlauf: NEUE MINUTE: wenn Fehlerflag
goto DCFROUTINEW1

bsf DCFSTATUS,DCFNEUEMIN ;nicht gesetzt, Flag DCFNEUEMIN setzen
goto DCFROUTINEFERTIG ; und Unterprogramm verlassen

DCFROUTINEW1 bcf DCFSTATUS,DCFNEUESEK ;Fehlerflag gesetzt: Fehlerflag löschen
goto DCFROUTINEFERTIG ; und Unterprogramm verlassen (Flag DCFNEUEMIN
; wird in diesem Fall nicht gesetzt)

;Pulszähler löschen
DCF PULSLOESCHEN clrf DCF PULS

DCFROUTINEFERTIG
return

;*****
;** Neue Sekunde: **
;** Aufgaben: **
;** + je nachdem ob das Bit DCFLOW oder das Bit DCFHIGH gesetzt ist dieses Bit dem **
;** Telegramm (Register DCFTELEGRAMM1 bis DCFTELEGRAMM8) hinzufügen **
;** + Anforderungsflag (Flag DCFNEUESEK) zurücksetzen **
;** **
;** Vorgehensweise: **
;** Ist das Flag DCFLOW (im Register DCFSTATUS) gesetzt das Carryflag (im SFR STAT) **
;** löschen, ist aber das Flag DCFHIGH (ebenfalls im Register DCFSTATUS) gesetzt das **
;** Carryflag setzten. Dieses Carryflag nun dem Telegramm hinzufügen. Dieser Vorgang **
;** erfolgt mit einem so genannten Schiebepfehl. Dabei werden alle Bits des Registers **
;** DCFTELEGRAMM1 auf die nächst höhere Position verschoben. (Bit 6 wandert ins Bit 7, **

```

## Analog/Digitale Wanduhr 2

```

; ** Bit 5 wandert ins Bit 6 usw. Bit 0 wandert ins Bit 1). Der Inhalt vom Carryflag **
; ** wandert ins Bit 0. Der Inhalt von Bit 7 wird ins Carryflag geschoben. Bei den **
; ** Register DCFTELEGRAMM2 bis DCFTELEGRAMM6 erfolgt derselbe Vorgang. (Der Inhalt von **
; ** Bit 7 des Registers DCFTELEGRAMM1 wird ins Carry geschoben, und das Carry aber weiter **
; ** in das Bit 0 des Registers DCFTELEGRAMM2) **
; ** **
; ** Anmerkung: **
; ** Die Flags DCFLOW und DCFHIGH können nie gleichzeitig gesetzt sein. Es ist nur **
; ** möglich, dass entweder nur DCFLOW oder nur DCFHIGH gesetzt ist, aber nie beide **
; ** gleichzeitig. **
; *****
DCFUPSEKUNDE   bt fsc   DCFSTATUS, DCFLOW       ;Ist das Flag DCFLOW im Register DCFSTATUS
                                           ; gesetzt?
               goto    DCFSCHIEBELOW
               bt fsc   DCFSTATUS, DCFHIGH      ;nein: Wenn das FLAG DCFHIGH im Register
DCFSCHIEBEHIGH bsf     STAT, C                 ; DCFSTATUS gesetzt ist, dem Telegramm ein
               goto    DCFSCHIEBE           ; Low mit Hilfe des Carry hinzufügen
DCFSCHIEBELOW bcf     STAT, C                 ;ja: Dem Telegramm ein High mit Hilfe des Carry
                                           ; hinzufügen

DCFSCHIEBE    rlf     DCFTELEGRAMM1, f
               rlf     DCFTELEGRAMM2, f
               rlf     DCFTELEGRAMM3, f
               rlf     DCFTELEGRAMM4, f
               rlf     DCFTELEGRAMM5, f
               rlf     DCFTELEGRAMM6, f
;               rlf     DCFTELEGRAMM7, f
;               rlf     DCFTELEGRAMM8, f

               bcf     DCFSTATUS, DCFNEUESEK   ;Anforderungsflag löschen
               return

; *****
; ** Neue Minute: **
; ** Aufgaben: **
; ** + Fehlende 59. Sekunde "nachholen" (Unterprogramm DCFUPSEKUNDE aufrufen) **
; ** + Datum, Uhrzeit und die Zusatzinformationen aus den Registern DCFTELEGRAMM1 bis **
; ** DCFTELEGRAMM8 zusammensetzen **
; ** + Das soeben ermittelte Telegramm mit dem Telegramm der vorhergehenden Minute ver- **
; ** gleichen. Die soeben ermittelte Minute muss dabei um 1 größer als die vorher- **
; ** gehende Minute sein, und die soeben ermittelten Werte für die Stunde, Tag, Monat, **
; ** Jahr und Wochentag muss mit den Werten des vorhergehenden Telegramms überein- **
; ** stimmen. ACHTUNG: Bei dieser einfachen Überprüfung wird der Übergang von z. B. **
; ** 13:59 auf 14:00 als falsch gewertet, da sich hier die Stunde ändert, und auch die **
; ** Minute um mehr als 1. Würden hier alle möglichen Übergänge berücksichtigt, **
; ** dann würde dieses Unterprogramm noch umfangreicher als es ohnehin schon ist. Durch **
; ** diese Vereinfachung dauert die Synchronisierung im schlimmsten Fall nur eine Minute **
; ** länger. **
; ** + Das alte Datum bzw. die alte Uhrzeit (von der vorhergehenden Minute) durch das neue **
; ** Datum bzw. Uhrzeit ersetzen. Dies ist für die Überprüfung des nächsten Telegramms **
; ** notwendig. **
; ** + Wenn das neu empfangene Datum bzw. die neu empfangene Uhrzeit gültig ist, die BCD- **
; ** kodierten Datums- bzw. Uhrzeitkomponenten in eine binäre Form umwandeln und damit **
; ** die mitlaufende Uhr synchronisieren. Das Flag DCFSYNC im Register DCFSTATUS setzen. **
; ** Dieses gesetzte Flag kennzeichnet, dass die mitlaufende Uhr mit der DCF-Uhr **
; ** synchronisiert ist **
; ** + Anforderungsflag (Flag DCFNEUEMIN im Register DCFSTATUS) zurücksetzen **
; ** **
; ** Anmerkung: **
; ** Das temporäre Register TEMP1 wird hier nur als Hilfsregister benötigt, und kann **
; ** daher auch in anderen Unterprogrammen verwendet werden. **
; *****
DCFUPMINUTE    call    DCFUPSEKUNDE           ;Fehlende Sekunde nachholen

               clr f   DCFJAHRBCD           ;Jahr zusammensetzen (BCD-Format)
               bt fsc  DCFTELEGRAMM1, 1
               bsf     DCFJAHRBCD, 7
               bt fsc  DCFTELEGRAMM1, 2
               bsf     DCFJAHRBCD, 6
               bt fsc  DCFTELEGRAMM1, 3
               bsf     DCFJAHRBCD, 5
               bt fsc  DCFTELEGRAMM1, 4
               bsf     DCFJAHRBCD, 4
               bt fsc  DCFTELEGRAMM1, 5
               bsf     DCFJAHRBCD, 3
               bt fsc  DCFTELEGRAMM1, 6
               bsf     DCFJAHRBCD, 2
               bt fsc  DCFTELEGRAMM1, 7
               bsf     DCFJAHRBCD, 1

```



## Analog/Digitale Wanduhr 2

```

btfsc DCFTELEGRAMM2,0
bsf DCFJAHRBCD,0

clrf DCFMONBCD ;Monat zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM2,1
bsf DCFMONBCD,4
btfsc DCFTELEGRAMM2,2
bsf DCFMONBCD,3
btfsc DCFTELEGRAMM2,3
bsf DCFMONBCD,2
btfsc DCFTELEGRAMM2,4
bsf DCFMONBCD,1
btfsc DCFTELEGRAMM2,5
bsf DCFMONBCD,0

clrf DCFWOCHENTAG ;Wochentag zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM2,6
bsf DCFWOCHENTAG,2
btfsc DCFTELEGRAMM2,7
bsf DCFWOCHENTAG,1
btfsc DCFTELEGRAMM3,0
bsf DCFWOCHENTAG,0

clrf DCFTAGBCD ;Tag zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM3,1
bsf DCFTAGBCD,5
btfsc DCFTELEGRAMM3,2
bsf DCFTAGBCD,4
btfsc DCFTELEGRAMM3,3
bsf DCFTAGBCD,3
btfsc DCFTELEGRAMM3,4
bsf DCFTAGBCD,2
btfsc DCFTELEGRAMM3,5
bsf DCFTAGBCD,1
btfsc DCFTELEGRAMM3,6
bsf DCFTAGBCD,0

clrf DCFSTDBCD ;Stunde zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM4,0
bsf DCFSTDBCD,5
btfsc DCFTELEGRAMM4,1
bsf DCFSTDBCD,4
btfsc DCFTELEGRAMM4,2
bsf DCFSTDBCD,3
btfsc DCFTELEGRAMM4,3
bsf DCFSTDBCD,2
btfsc DCFTELEGRAMM4,4
bsf DCFSTDBCD,1
btfsc DCFTELEGRAMM4,5
bsf DCFSTDBCD,0

clrf DCFMINBCD ;Minute zusammensetzen (BCD-Format)
btfsc DCFTELEGRAMM4,7
bsf DCFMINBCD,6
btfsc DCFTELEGRAMM5,0
bsf DCFMINBCD,5
btfsc DCFTELEGRAMM5,1
bsf DCFMINBCD,4
btfsc DCFTELEGRAMM5,2
bsf DCFMINBCD,3
btfsc DCFTELEGRAMM5,3
bsf DCFMINBCD,2
btfsc DCFTELEGRAMM5,4
bsf DCFMINBCD,1
btfsc DCFTELEGRAMM5,5
bsf DCFMINBCD,0

; clrf DCFZUSATZINFOS ;zusätzliche Informationen
; btfsc DCFTELEGRAMM5,7 ; zusätzliche Schaltsekunde
; bsf DCFZUSATZINFOS,DCFSCHALTSEK
; btfsc DCFTELEGRAMM6,0 ; Sommerzeit
; bsf DCFZUSATZINFOS,DCFSOMMER
; btfsc DCFTELEGRAMM6,1 ; Winterzeit
; bsf DCFZUSATZINFOS,DCFWINTER
; btfsc DCFTELEGRAMM6,2 ; Vorankündigung: Wechsel
; bsf DCFZUSATZINFOS,DCFSOMWIN ; Sommerzeit <-> Winterzeit
; btfsc DCFTELEGRAMM6,3 ; Reserveantenne
; bsf DCFZUSATZINFOS,DCFRESANT

```

## Analog/Digitale Wanduhr 2

```

;Neues Telegramm mit dem alten Vergleichen
bcf     DCFSTATUS,DCFFEHLER ;zuerst Fehlerflag löschen

movf   DCFMINALT,w           ;Minuten Prüfen
subwf  DCFMINBCD,w
movwf  TEMP1                 ;TEMP1 = DCFMINBCD - DCFMINALT
decfsz TEMP1,w              ;TEMP1 - 1 = 0 ?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf   DCFSTDALT,w          ;Stunden Prüfen
subwf  DCFSTDBCD,w
btfss  STAT,Z               ;DCFSTDBCD - DCFSTDALT = 0?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf   DCFTAGALT,w         ;Tag Prüfen
subwf  DCFTAGBCD,w
btfss  STAT,Z               ;DCFTAGBCD - DCFTAGBCD = 0?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf   DCFMONALT,w         ;Monat Prüfen
subwf  DCFMONBCD,w
btfss  STAT,Z               ;DCFMONBCD - DCFMONALT = 0?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf   DCFJAHRALT,w        ;Jahr Prüfen
subwf  DCFJAHRBCD,w
btfss  STAT,Z               ;DCFJAHRBCD - DCFJAHRALT = 0?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

movf   DCFWTAGALT,w        ;Wochentag Prüfen
subwf  DCFWOCHENTAG,w
btfss  STAT,Z               ;DCFWOCHENTAG - DCFWTAGALT = 0?
bsf    DCFSTATUS,DCFFEHLER ;nein: Fehlerflag setzen

;Altes Telegramm durch das neue Telegramm ersetzen
movf   DCFMINBCD,w
movwf  DCFMINALT

movf   DCFSTDBCD,w
movwf  DCFSTDALT

movf   DCFTAGBCD,w
movwf  DCFTAGALT

movf   DCFMONBCD,w
movwf  DCFMONALT

movf   DCFJAHRBCD,w
movwf  DCFJAHRALT

movf   DCFWOCHENTAG,w
movwf  DCFWTAGALT

btfsc  DCFSTATUS,DCFFEHLER ;DCF-Telegramm korrekt?
goto   DCFUPMINUTEENDE     ;nein: UP beenden
movf   DCFSTDBCD,w         ;ja: DCFSTDBCD
call   BCDBIN2             ;von BCD nach binär umwandeln
movwf  UHRSTUNDE           ;und die Stunde der mitlaufenden Uhr
;überschreiben

movf   DCFMINBCD,w         ;DCFMINBCD
call   BCDBIN2             ;von BCD nach binär umwandeln
movwf  UHRMINUTE           ;und die Minute der mitlaufenden Uhr
;überschreiben

clrf   UHRSEKUNDE         ;Sekunde löschen

movf   DCFTAGBCD,w         ;DCFTAGBCD
call   BCDBIN2             ;von BCD nach binär umwandeln
movwf  DATUMTAG            ;und den Tag des mitlaufenden Datums
;überschreiben

movf   DCFMONBCD,w         ;DCFMONBCD
call   BCDBIN2             ;von BCD nach binär umwandeln
movwf  DATUMMONAT          ;und den Monat des mitlaufenden Datums
;überschreiben

movf   DCFJAHRBCD,w        ;DCFJAHRBCD
call   BCDBIN2             ;von BCD nach binär umwandeln

```

## Analog/Digitale Wanduhr 2

```

movwf  DATUMJAHR          ;und das Jahr des mitlaufenden Datums
                           ;überschreiben

movf   DCFWOCHENTAG,w    ;DCFWOCHENTAG
movwf  DATUMWTAG

bsf    DCFSTATUS,DCFSYNC

DCFUPMINUTEENDE bcf  DCFSTATUS,DCFFEHLER
bcf    DCFSTATUS,DCFNEUEMIN ;Anforderungsflag löschen

return

;*****
; ** INNERE UHR **
; **
; ** Aufgabe:
; ** für den Fall dass kein gültiges DCF-Telegramm empfangen werden kann sorgt dieses
; ** Unterprogramm dafür, dass die Uhrzeit trotzdem jede Sekunde aktualisiert wird. Ist
; ** für eine längere Zeit kein korrekter DCF-Empfang möglich, so Würden die Minuten,
; ** Stunden, Tage usw. stehen bleiben, da ja im Unterprogramm DCFUPMINUTE nur gültige
; ** DCF-Telegramme übernommen werden.
; ** Dieses Unterprogramm wird also parallel zu den Unterprogrammen für die DCF-Dekod-
; ** ierung aufgerufen.
; **
; ** Vorgehensweise:
; ** Die Sekunden (Register UHRSEKUNDE) um 1 erhöhen. Beinhalte dieses Register nun den
; ** Wert 60, so beginnt eine neue Minute. Daher die Sekunden löschen und die Minuten
; ** (Register UHRMINUTE) um 1 erhöhen. Beinhaltet dieses Register nun den Wert 60, so
; ** beginnt eine neue Stunde. Daher die Minuten löschen und die Stunden (Register
; ** UHRSTUNDE) um 1 erhöhen. Beinhaltet dieses Register nun den Wert 24, so beginnt ein
; ** neuer Tag. Daher die Stunden löschen. Ist das Mitzählen des Datums notwendig, so
; ** muss nun ein Register für den Tag (z. B. DATUMTAG) um 1 erhöht werden und dieses
; ** überprüft werden, wobei diese Prüfung nun nicht mehr so einfach ist, da ja jeder
; ** Monat unterschiedlich viele Tage besitzt. Erschwerend kommt auch noch hinzu, dass
; ** auch die Schaltjahre miteinbezogen werden müssen!
; **
; ** Achtung:
; ** Dieses Unterprogramm muss daher jede Sekunde aufgerufen werden
; *****
INNEREUHR   incf  UHRSEKUNDE,f      ;Sekundenzähler um 1 erhöhen
            movf  UHRSEKUNDE,w
            sublw .60
            btfss STAT,Z          ;Prüfen, ob Sekunde=60
            goto  INNERERUHRFERTIG
            clrf  UHRSEKUNDE      ;Wenn Sekunde=60, Sekunde löschen
            incf  UHRMINUTE,f      ; und Minute um 1 erhöhen
            movf  UHRMINUTE,w
            sublw .60
            btfss STAT,Z          ;Prüfen, ob Minute=60
            goto  INNERERUHRFERTIG
            clrf  UHRMINUTE      ;Wenn Minute=60, Minute löschen
            incf  UHRSTUNDE,f     ; und Stunde erhöhen
            movf  UHRSTUNDE,w
            sublw .24
            btfsc STAT,Z          ;Prüfen, ob Stunde=24
            clrf  UHRSTUNDE      ;Wenn Stunde=24, Stunden löschen
INNERERUHRFERTIG
            return

;***** Wanduhrspezifisches Unterprogramm *****
;*****
; ** ANZEIGE **
; **
; ** Dieses Unterprogramm wird vom Hauptprogramm alle 4ms aufgerufen
; **
; ** Aufgabe und Vorgehensweise:
; ** Wenn die mitlaufende Uhr mit der DCF-Uhr synchronisiert ist (Flag DCFSYNC im Register
; ** DCFSTATUS gesetzt) die Uhrzeit wie folgt ausgeben. Andenfalls dieses Unterprogramm
; ** vorzeitig beenden: Stunde am Port A und abwechselnd Minute und Sekunde am Port B
; ** ausgeben. Abwechselnd bedeutet hier, dass beim Aufruf dieses Unterprogramms entweder
; ** die Minuten am Port B ausgegeben werden oder die Sekunden. Diese Methode entspricht
; ** einer einfachen Multiplex-Ansteuerung. Die Unterscheidung, ob die Minuten oder die
; ** Sekunden ausgegeben werden hängt vom Bit 0 des Registers FLAGS ab. Dieses Bit wird
; ** bei jedem Aufruf invertiert (getoggelt). Besitzt es danach den Wert 1, so werden die
; ** Minuten angezeigt. Beim nächsten Aufruf dieses Unterprogramms (in ca. 4ms) werden

```

## Analog/Digitale Wanduhr 2

```

**      dann die Sekunden angezeigt, da dieses Bit ja wieder invertiert wurde und daher den      **
**      Wert 0 besitzt.                                                                    **
**      Mit Hilfe der beiden Tabellen TABSTUNDEN und TABMINUTENSEKUNDEN erfolgt eine An-      **
**      passung an die Hardware (Umwandlung von 24 Stunden auf 12 Stunden, LED-Matrix am      **
**      Port B).                                                                            **
*****
ANZEIGE      btfss   DCFSTATUS,DCFSYNC      ;Ist die mitlaufende Uhr mit der DCF-Uhr
                                                    ; synchronisiert?
              goto   ANZEIGEENDE          ;nein: Unterprogramm vorzeitig beenden

              movf   UHRSTUNDE,w          ;ja: Stunde mit Hilfe der Tabelle TABSTUNDE
              andlw  b'00011111'
              call   TABSTUNDEN          ; von 0-23 auf 0-11 umwandeln und um eine
                                                    ; Stelle verschieben
              movwf  PORTA                ; und am Port A ausgeben

              btg    FLAGS,0
              btfss  FLAGS,0              ;Abwechselnd
              goto   ANZEIGESEK

ANZEIGEMIN   movf   UHRMINUTE,w          ; Minute
              goto   ANZEIGEW1

ANZEIGESEK   movf   UHRSEKUNDE,w        ; oder Sekunde

ANZEIGEW1    andlw  b'00111111'
              call   TABMINUTENSEKUNDEN ; so anpassen, dass die richtige
              movwf  PORTB              ; Leuchtdiode am Port B leuchtet

ANZEIGEENDE return

***** Weitere, allgemeine Routinen *****

*****
**      Umwandlung von BCD nach binär:                                                    **
**      Aufgabe:                                                                            **
**      + Die im w-Register stehende 2-stellige-BCD-Zahl nach binär umwandeln            **
**      **                                                                                   **
**      Vorgehensweise:                                                                    **
**      + Die umzuwandelnde BCD-Zahl im temporären Register TEMP1 zwischenspeichern      **
**      + w-Register (Arbeitsregister) löschen.                                           **
**      + Die BCD-Zahl bitweise überprüfen.                                               **
**      z .B: BCD-Zahl: 0010 0110 (= 26)                                                 **
**      **                                                                                   **
**      | | | | | | | | - 0 (= 0 x 1)                                                    **
**      | | | | | | | | -- + 2 (= 1 x 2)                                                 **
**      | | | | | | | | --- + 4 (= 1 x 4)                                                 **
**      | | | | | | | | ---- + 0 (= 0 x 8)                                               **
**      | | | | | | | | ----- + 0 (= 0 x 10)                                           **
**      | | | | | | | | ----- +20 (= 1 x 20)                                           **
**      | | | | | | | | ----- + 0 (= 0 x 40)                                           **
**      | | | | | | | | ----- + 0 (= 0 x 80)                                           **
**      **                                                                                   **
**      ----- + 0 (= 0 x 80)                                                           **
**      **                                                                                   **
**      ----- =26                                                                      **
**      **                                                                                   **
**      + Ergebnis in w                                                                    **
**      **                                                                                   **
**      Anmerkung: Das temporäre Register TEMP1 wird hier nur zum Zwischenspeichern benötigt **
**      und kann daher auch wo anders verwendet werden.                                  **
*****
BCDBIN2      movwf  TEMP1                ;umzuwandelnde BCD-Zahl zwischenspeichern
              clrw   ;Arbeitsregister (w-Register) löschen
              btfsc  TEMP1,0              ;Ist Bit0 der BCD-Zahl gesetzt,
              addlw  .1                  ; zum w-Register den Wert 1 addieren
              btfsc  TEMP1,1              ;Ist Bit1 der BCD-Zahl gesetzt,
              addlw  .2                  ; zum w-Register den Wert 2 addieren
              btfsc  TEMP1,2              ;Ist Bit2 der BCD-Zahl gesetzt,
              addlw  .4                  ; zum w-Register den Wert 4 addieren
              btfsc  TEMP1,3              ;Ist Bit3 der BCD-Zahl gesetzt,
              addlw  .8                  ; zum w-Register den Wert 8 addieren
              btfsc  TEMP1,4              ;Ist Bit4 der BCD-Zahl gesetzt,
              addlw  .10                 ; zum w-Register den Wert 10 addieren
              btfsc  TEMP1,5              ;Ist Bit5 der BCD-Zahl gesetzt,
              addlw  .20                 ; zum w-Register den Wert 20 addieren
              btfsc  TEMP1,6              ;Ist Bit6 der BCD-Zahl gesetzt,
              addlw  .40                 ; zum w-Register den Wert 40 addieren
              btfsc  TEMP1,7              ;Ist Bit7 der BCD-Zahl gesetzt,
              addlw  .80                 ; zum w-Register den Wert 80 addieren, im w-Register
                                                    ; befindet sich nun das Ergebnis

```

## Analog/Digitale Wanduhr 2

```

return

;***** Hauptprogramm *****
;*****
;** Aufgaben des Hauptprogramms: **
;** + Aufrufen von INIT (Initialisierungen) **
;** + Interrupt freigeben **
;** + Tätigkeiten und/oder Unterprogramme, die alle 4ms durchgeführt werden müssen **
;** + Unterprogramm DCFROUTINE aufrufen **
;** + Unterprogramm ANZEIGE aufrufen **
;** + Tätigkeiten und/oder Unterprogramme, die jede Sek. durchgeführt werden müssen **
;** + Unterprogramm INNEREUHR aufrufen **
;** + Bei einem Minutenwechsel (Flag DCFNEUEMIN ist gesetzt) das Unterprogramm **
;** DCFUPMINUTE abarbeiten **
;** + Bei einem Sekundenwechsel (Flag DCFNEUESEK ist gesetzt) das Unterprogramm **
;** DCFUPSEKUNDE abarbeiten **
;*****
Beginn      call    INIT                ;Initialisierungen
            movlw  b'10100000'        ;TMR0 freigeben durch
            movwf  INTCON              ;Setzen von GIE und T0IE im Register INTCON

HPSCHLEIFE  btfsc  FLAGISRHP,FLAG4MSEK
            goto   HPWEITER1

            btfsc  FLAGISRHP,FLAG1SEK
            goto   HPWEITER2

            btfsc  DCFSTATUS,DCFNEUEMIN ;Wenn Anforderungsflag DCFNEUEMIN gesetzt
            call   DCFUPMINUTE          ;Unterprogramm DCFUPMINUTE ausführen
            btfsc  DCFSTATUS,DCFNEUESEK ;Wenn Anforderungsflag DCFNEUESEK gesetzt
            call   DCFUPSEKUNDE        ;Unterprogramm DCFUPSEKUNDE ausführen

            goto   HPSCHLEIFE

HPWEITER1   call   DCFROUTINE          ;Tätigkeiten, die alle 4 ms durchgeführt
            ; werden müssen

            call   ANZEIGE
            bcf    FLAGISRHP,FLAG4MSEK ;Anforderungsflag (für 4ms-Aktivitaeten)
            ; zurücksetzen

            goto   HPSCHLEIFE

HPWEITER2   call   INNEREUHR          ;Tätigkeiten, die jede Sekunde durchgeführt
            ; werden müssen

            bcf    FLAGISRHP,FLAG1SEK  ;Anforderungsflag (für 1-Sek.-Aktivitaeten)
            ; zurücksetzen

            goto   HPSCHLEIFE

end

```

**Anhang C: Stückliste**

Nr.	Bezeichnung	St.	Lieferant	Bestell. Nr.	E-Preis	Bemerkungen
R3	Widerstand 270 Ohm	1	Conrad	418188	0,14	
R5, R19-R21	Widerstand 1k	4	Conrad	418250	0,14	
R1, R2, R4, R6	Widerstand 10k	4	Conrad	418374	0,14	
R7-R18	Widerstand 33k	12	Conrad	418439	0,11	
C2, C3	Keramikkondensator 22pF	2	Conrad	457167	0,14	
C4-C8, C10	Keramikkondensator 100nF	6	Conrad	453358	0,25	
C1, C9	Elko 10µF/25V	2	Conrad	460524	0,14	
D1-D60	LED 5mm rot, low current	60	Conrad	186902	0,33	
D61-D72	LED 5mm grün, low current	12	Conrad	186929	0,33	
D73,D74	LED 3mm gelb, low current	2	Conrad	145980	0,25	
T1	Transistor BC547B	1	Conrad	155012	0,21	
IC1	PIC16F84	1	Conrad	146773	10,80	Programmiert mit wanduhr2.hex
IC2	4067	1	Conrad	173320	2,30	
IC3, IC4	4051	2	Conrad	172901	0,85	
IC5	LP2950	1	Conrad	175676	2,89	
X1	Quarz 4,096 MHz	1	Conrad	168629	1,40	
DCF1	DCF-Empfangsmodul	1	Conrad	641138	11,50	
	IC-Präzisions-Sockel 16polig	2	Conrad	189626	0,57	für IC3 und IC4
	IC-Präzisions-Sockel 18polig	1	Conrad	189634	0,57	für IC1
	IC-Präzisions-Sockel 24polig	1	Conrad	189650	0,85	für IC2
	Buchsenleiste 32polig	3	Conrad	740438	1,15	
	Kontaktstreifen	2	Conrad	739073	4,28	
	Steckernetzteil PA300, unstabilisiert	1	Conrad	518305	6,47	
	Zum Steckernetzteil passendes Gegenstück	1				
	Lochraster-Platine	2	Conrad	527769	3,25	
	Sperrholzplatte, 4mm dick (25 x 25 cm)	1				
	Bastelglas, 2mm dick (25 x 25 cm)	1				
	div. Montagematerial					