

DCF

(Dekodierung mit
PIC-Mikrocontroller)

Autor: Buchgeher Stefan
Letzte Bearbeitung: 15. Oktober 2006

Inhaltsverzeichnis

1.	GRUNDLEGENDES ZU DCF	3
2.	HARDWARE	5
3.	SOFTWARE	5
3.1.	Das Dekodierverfahren	6
3.2.	Benötigte Register, Konstanten und Portdefinition.....	7
3.3.	Initialisierung (Unterprogramm „INIT“).....	10
3.4.	Unterprogramme für die DCF-Dekodierung	11
3.4.1.	Unterprogramm DCFROUTINE	11
3.4.2.	Unterprogramm DCFUPSEKUNDE	15
3.4.3.	Unterprogramm DCFUPMINUTE	16
3.4.4.	Unterprogramm INNEREUHR	20
3.5.	Änderungen wenn DCFROUTINE alle 10ms aufgerufen wird.....	20
4.	DEMONSTRATIONSBEISPIEL	21
4.1.	Hardware.....	21
4.2.	Software.....	22
4.3.	Anmerkungen zur Software	34

1. 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 1)



Bild 1: Reichweite des DCF77-Senders
(Quelle: www.dcf77.com/)

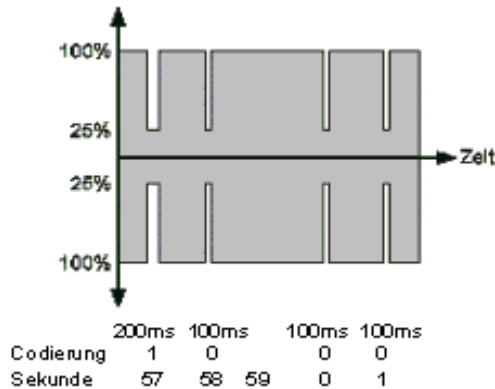


Bild 2: Amplitudenmodulation des 77,5kHz-Trägers

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: ± 20 ms) entspricht dabei

einem logischen Low-Pegel, während ein logischer High-Pegel mit einer Absenkdauer von 200ms (Toleranz ± 40 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 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 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 Zeitzonebits 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.

2. Hardware



Die Hauptkomponente 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: DCF-Empfangsmodul

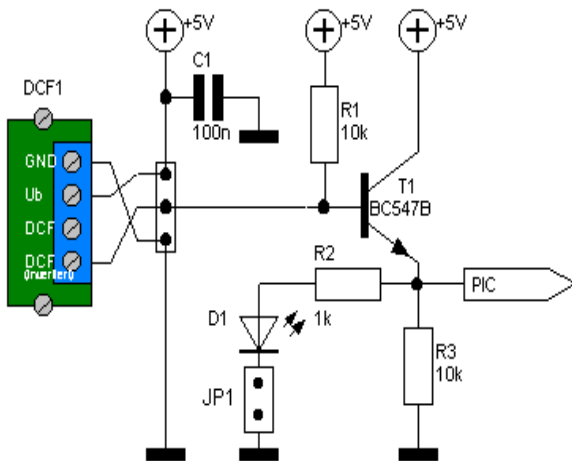


Bild 4: Anpass-Schaltung

Der Ausgangsstrom von nur einem Milliampere ist für die Kontroll-LED (D1) zu wenig. Die nachgeschaltete Transistorstufe (T1) mit den Widerständen R1 und R2 gleicht diesen Nachteil aus. Der Widerstand R2 dient als Vorwiderstand für die Leuchtdiode (D1). Diese Leuchtdiode signalisiert den empfangenen Datenstrom, wenn der Jumper JP1 gesteckt ist. Bei einem korrekten Empfang blinkt diese Leuchtdiode im Sekundentakt. Dieses Blinken zeigt sozusagen den Ausgangspegel der Anpass-Schaltung an. Das Puls-Pausen-Verhältnis (Leuchtzeit/Dunkelzeit der Leuchtdiode D1) entspricht der im Abschnitt 1 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 unnötig belastet wird.

Der Kondensator C1 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.

3. Software

Die Aufgabe der PIC-Software besteht bei der DCF-Dekodierung 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.

3.1. Das Dekodierverfahren

Der PIC-I/O-Eingang, an dem das DCF-Empfangsmodul (mit einer Anpass-Schaltung) angeschlossen 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 500µs oder alle 1ms einen Interrupt erzeugt und ein Zählregister zählt die notwendige Anzahl an Interruptaufrufen bis zu einer Zeitbasis von 4 ms. Hier in dieser Dokumentation 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 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 1 (Grundlegendes zu DCF) gewonnen Erkenntnisse zu dekodieren. Für diese Informationen befinden sich im DCF-Statusregister (DCFSTATUS) entsprechende Flags. (siehe Abschnitt 3.2. Benötigte Register, Konstanten und Portdefinition). 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 zwischenzuspeichern.

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 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

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 3.4.3)

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.

3.2. Benötigte Register, Konstanten und Portdefinition

Register:

Für die DCF-Dekodierung sind neben einigen internen Register (SFR, **S**pezielle **F**unktions-**R**egister) noch eine ganze Menge eigener Register notwendig:

- **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 2: 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 3: 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: 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. Dieses 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 quartzgesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert).
- UHRMINUTE: Minutenzähler der inneren quartzgesteuerten Uhr. (wird mit jedem gültigen DCF-dekod. Wert synchronisiert).
- UHRSTUNDE: Stundenzähler der inneren quartzgesteuerten 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).
- 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 ($250 \times 4\text{ms} = 1000\text{ms} = 1 \text{ Sekunde}$).

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 1. Grundlegendes zu DCF). Die Konstanten *KONSTDCFLMIN*, *KONSTDCFLMAX*, *KONSTDCFHMIN* und *KONSTDCFHMAX* dienen zur Ermittlung eines LOW oder eines HIGH.

Das Unterprogramm DCFROUTINE wird alle 4ms aufgerufen, daher ergeben sich für die Konstanten folgende Werte:

- KONSTDCFLMIN: 20 ($20 \times 4\text{ms} = 80\text{ms}$)
- KONSTDCFLMAX: 30 ($30 \times 4\text{ms} = 120\text{ms}$)
- KONSTDCFHMIN: 40 ($40 \times 4\text{ms} = 160\text{ms}$)
- KONSTDCFHMAX: 60 ($60 \times 4\text{ms} = 240\text{ms}$)

Es kann erforderlich sein, dass bei einer Anwendung die PIC-Taktfrequenz erhöht werden muss. In diesem Fall kann es notwendig sein, diese Konstanten anzupassen.

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 B verwendet, so muss für DCFINTRIS das zum Port B zugehörige TRIS-Register definiert werden. Eine mögliche Portdefinition ist:

```
DCFINPORT equ PORTB
DCFINTRIS equ TRISB
DCFIN equ 0
```

3.3. Initialisierung (Unterprogramm „INIT“)

Dieses Unterprogramm dient zur Initialisierung des Mikrocontrollers. Der Portpin an dem der DCF-Empfänger angeschlossen ist muss als Eingang definiert werden. In der Initialisierungsroutine muss für den ersten 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-Interrupt 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.

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine. Die schwarz hervorgehobenen Stellen sind für die DCF-Dekodierung notwendig.

```
INIT      clrf  TMR0                ;Timer0 auf 0 voreinstellen

          clrf  PORTA
          movlw 0x07                ;Alle Comparatoreingänge
          movwf CMCON              ; auf digital I/O umschalten

          bsf   STAT,RP0            ;Registerseite 1
          movlw b'00000011'        ;interner Takt, Vorteiler = 16 an TMR0
          movwf OPTREG

          bsf   DCFINTRIS,DCFIN    ;DCFIN als Eingang definieren
          bcf   STAT,RP0            ;Registerseite 0

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

          clrf  DCFSTATUS          ;DCF-Statusregister initialisieren
          bsf   DCFSTATUS,DCFFEHLER ; Fehlerflag setzen, alle anderen Flags
          ; loeschen

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

          clrf  UHRSEKUNDE        ;Sekundenzaehler initialisieren
          clrf  UHRMINUTE         ;Minutenzaehler initialisieren
          clrf  UHRSTUNDE        ;Stundenzaehler initialisieren
          clrf  DATUMWTTAG        ;Wochentagzaehler initialisieren
          clrf  DATUMTTAG        ;Tageszaehler initialisieren
          clrf  DATUMMONAT        ;Monatszaehler initialisieren
          clrf  DATUMJAHR        ;Jahreszaehler initialisieren

          movlw .99
          movwf DCFMINALT         ;Vergleichsregister initialisieren
          movwf DCFSTDALT         ;Vergleichsregister initialisieren
          movwf DCFRTAGALT        ;Vergleichsregister initialisieren
          movwf DCFMONALT         ;Vergleichsregister initialisieren
```

```

movwf DCFJAHRALT      ;Vergleichsregister initialisieren
movwf DCFWTAGALT      ;Vergleichsregister initialisieren
return

```

3.4. Unterprogramme für die DCF-Dekodierung

Zur DCF-Dekodierung sind 3 Unterprogramme notwendig:

- *DCFROUTINE*
- *DCFUPSEKUNDE*
- *DCFUPMINUTE*

Weiters das Unterprogramm *INNEREUHR*. Dieses Unterprogramm sorgt dafür, dass, wenn kein gültiges DCF-Telegramm empfangen werden kann, 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 im Unterprogramm *DCFUPMINUTE* nur gültige DCF-Telegramme übernommen werden.

Dazu kommt noch ein „allgemeines“ Unterprogramm zur Umwandlung einer 2stelligen BCD-Zahl in eine Binärzahl. Dieses Unterprogramm wird hier allerdings nicht näher beschrieben.

3.4.1. Unterprogramm 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 ($=255 * 4ms$).

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 *KONSTDCFLMIN* und *KONSTDCFLMAX*, 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

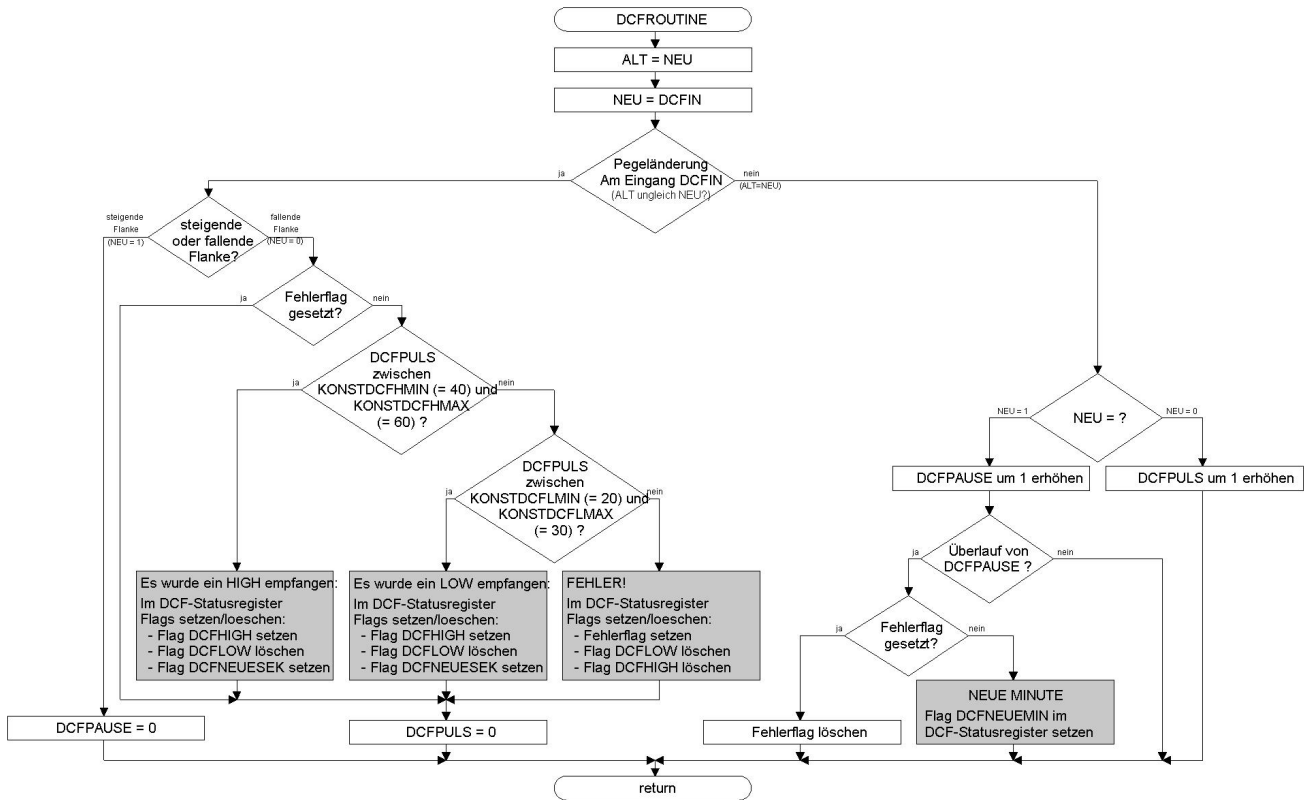
DCFPULS gelöscht. Beinhaltet das Zählregister DCFPULS einen Wert zwischen den Konstanten KONSTDCFHMIN und KONSTDCFHMAX, so wurde ein HIGH des DCF-Telegramms ermittelt. Im DCF-Statusregister (DCFSTATUS) werden daher die Flags DCFHIGH und DCFNEUESEK gesetzt. Das Flag DCFNEUESEK dient auch hier als Zeichen 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 KONSTDCFLMIN und KONSTDCFLMAX noch zwischen den Konstanten KONSTDCFHMIN und KONSTDCFHMAX 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 1 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.

DCF (Dekodierung mit PIC-Mikrocontroller)



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. Pause-Zähler 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
    
```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

                                ; (Plausibilitaetspruefung)
                                ;ja: Puls-Zaehler loeschen
goto DCFPULSLOESCHEN

DCFPLAUSIBEL ;Pruefen, ob ein High empfangen wurde
DCFCECHKHIGH
    movlw KONSTDCFHMIN          ;unteren High-Grenzwert laden
    subwf DCFPULS,w
    btfss STAT,C                ;Grenzwert < DCFPULS ?
    goto DCFCECHKLOW           ;nein: kein HIGH-Pegel, pruefen, ob
                                ; LOW
    movlw KONSTDCFHMAX          ;ja: DCFPULS mit oberen HIGH-
                                ; Grenzwert vergleichen
    subwf DCFPULS,w
    btfsc STAT,C                ;DCF PULS < Grenzwert ?
    goto DCFCECHKLOW           ;nein: es handelt sich um kein HIGH-
                                ; Signal, pruefen, ob ein LOW
                                ; empfangen wurde
    bcf DCFSTATUS,DCFLOW        ;ja: Es wurde ein High-Signal
                                ; empfangen, das Flag DCFLOW
                                ; loeschen,
    bsf DCFSTATUS,DCFHIGH       ; das Flag DCFHIGH setzen,
    bsf DCFSTATUS,DCFNEUESEK    ; das Flag DCFNEUESEK setzen
    goto DCFPULSLOESCHEN       ; und Puls-Zaehler loeschen

                                ;Pruefen, ob ein Low empfangen wurde
DCFCECHKLOW movlw KONSTDCFMIN    ;unteren Low-Grenzwert laden
    subwf DCFPULS,w
    btfss STAT,C                ;Grenzwert < DCFPULS ?
    goto DCFSETFEHLER          ;nein: kein LOW-Pegel,
                                ; Empfangsfehler
    movlw KONSTDCFMAX          ;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 loeschen,
    bsf DCFSTATUS,DCFNEUESEK    ; das Flag DCFNEUESEK setzen
    goto DCFPULSLOESCHEN       ; und den Puls-Zaehler loeschen

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

                                ;Puls- oder Pausen-Zaehler erhoehen
DCFINCPULSPAUSE
    btfsc DCFSTATUS,DCFPORTNEU
    goto DCFINCPAUSE

DCFINCPULS  incf DCFPULS,f      ;Puls-Zaehler um 1 erhoehen
    goto DCFROUTINEFERTIG

DCFINCPAUSE incfsz DCFPAUSE,f   ;Pause-Zaehler erhoehen und auf
                                ; Ueberlauf pruefen
    goto DCFROUTINEFERTIG      ;kein Ueberlauf: Unterprogramm
                                ; verlassen

```

```

    btfscc DCFSTATUS,DCFFEHLER      ;Ueberlauf: NEUE MINUTE; wenn
                                    ; Fehlerflag
    goto   DCFROUTINEW1

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

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

                                ;Puls-Zaehler loeschen
DCFPUSSLOESCHEN
    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.

3.4.2. Unterprogramm 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
    btfscc 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 hinzufuegen
DCFSCHIEBELOW
        bcf STAT,C ;ja: Dem Telegramm ein High mit Hilfe des
        ; Carry hinzufuegen

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 loeschen
        return

```

3.4.3. Unterprogramm 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

        clrf DCFJAHRBCD ;Jahr zusammensetzen (BCD-Format)
        btfsc DCFTELEGRAMM1,1
        bsf DCFJAHRBCD,7
        btfsc DCFTELEGRAMM1,2
        bsf DCFJAHRBCD,6
        btfsc DCFTELEGRAMM1,3

```

DCF (Dekodierung mit PIC-Mikrocontroller)

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

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

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

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

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

clrf   DCFMINBCD        ;Minute zusammensetzen (BCD-Format)
btfs   DCFTELEGRAMM4, 7
bsf    DCFMINBCD, 6
```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

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

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

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

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

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

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

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

movf  DCFWTAGALT,w                 ;Wochentag pruefen
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

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```
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 binaer umwandeln
movwf UHRSTUNDE ;und die Stunde der mitlaufenden Uhr
;ueberschreiben

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

clrf UHRSEKUNDE ;Sekunde loeschen

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

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

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

movf DCFWOCHENTAG,w ;DCFWOCHENTAG
movwf DATUMWTAG

bsf DCFSTATUS,DCFSYNC

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

return
```

Anmerkung:

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

3.4.4. Unterprogramm 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 erhoehen
             movf  UHRSEKUNDE,w
             sublw .60
             btfss STAT,Z          ;Pruefen, ob Sekunde=60
             goto  INNERERUHRFERTIG
             clrf  UHRSEKUNDE      ;Wenn Sekunde=60, Sekunde loeschen
             incf  UHRMINUTE,f      ; und Minute um 1 erhoehen
             movf  UHRMINUTE,w
             sublw .60
             btfss STAT,Z          ;Pruefen, ob Minute=60
             goto  INNERERUHRFERTIG
             clrf  UHRMINUTE      ;Wenn Minute=60, Minute loeschen
             incf  UHRSTUNDE,f     ; und Stunde erhoehen
             movf  UHRSTUNDE,w
             sublw .24
             btfsc STAT,Z         ;Pruefen, ob Stunde=24
             clrf  UHRSTUNDE      ;Wenn Stunde=24, Stunden loeschen

INNERERUHRFERTIG
             Return
```

3.5. Änderungen wenn DCFROUTINE alle 10ms aufgerufen wird

Noch nicht ausgeführt!

4.2. Software

```

;*****
; ** Demonstrationsbeispiel zur DCF-Dekodierung **
; ** **
; ** Entwickler: Buchgeher Stefan **
; ** Entwicklungsbeginn der Software: 19. Oktober 2003 **
; ** Funktionsfaehig seit: 8. Dezember 2003 **
; ** Letzte Bearbeitung: 26. Februar 2004 **
;*****

List p=PIC16F628

;***** Register (in Registerseite 0) *****
TMR0      equ    1      ;Timer0-Register
PC        equ    2      ;Programmzaehler
STAT      equ    3      ;Statusregister
PORTA     equ    5      ;PortA-Register
PORTB     equ    6      ;PortB-Register
INTCON    equ    0B     ;Interrupt-Register
CMCON     equ    1F     ;Komparator-Register

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

;***** Eigene Register (in Registerbank 0) *****
ISR_STAT_TEMP equ    20      ;Zwischenspeicher des Statusregister der ISR
ISR_w_TEMP    equ    21      ;Zwischenspeicher des Arbeitsregister der ISR
FLAGISRHP    equ    22      ;beinhaltet Botschaftsflags ISR -> HP
ZAEHLERISR1SEK equ    23      ;Zaehlerregister fuer 1-Sekunden-Zeitbasis
DCFSTATUS    equ    24      ;DCF-Statusregister
DCFPU LS     equ    25      ;Impulszeit des DCF-Impulses
DCFPAUSE     equ    26      ;Zeit zwischen 2 DCF-Impulsen
DCFTELEGRAMM1 equ    27      ;in diese Register wird das DCF-Telegramm
DCFTELEGRAMM2 equ    28      ; im Sekunden-Takt eingelesen. Beginnt eine
DCFTELEGRAMM3 equ    29      ; neue Minute, so werden diese Register
DCFTELEGRAMM4 equ    2A      ; ausgewertet, und in die entsprechenden
DCFTELEGRAMM5 equ    2B      ; DCF-Zeit-Datums-Register kopiert (z.B. in das
DCFTELEGRAMM6 equ    2C      ; Register DCFMINBCD)
DCFTELEGRAMM7 equ    2D      ;Reserve
DCFTELEGRAMM8 equ    2E      ;Reserve
DCFMINBCD    equ    2F      ;Beinhaltet die aktuelle dekodierte Minute (BCD-Format)
DCFSTDBCD    equ    30      ;Beinhaltet die aktuelle dekodierte Stunde (BCD-Format)
DCF TAGBCD   equ    31      ;Beinhaltet den aktuell dekodierten Tag (BCD-Format)
DCFMONBCD    equ    32      ;Beinhaltet den aktuell dekodierten Monat (BCD-Format)
DCFJAHRBCD   equ    33      ;Beinhaltet das aktuell dekodierte Jahr (BCD-Format)
DCFWOCHENTAG equ    34      ;Beinhaltet den aktuell dekodierten Wochentag
; (1=Montag, 2=Dienstag, usw., 7=Sonntag)
DCFZUSATZINFOS equ    35      ;Beinhaltet Zusatzinformationen (Sommer/Winterzeit,
; Reserveantenne usw.)
DCFMINALT    equ    36      ;In den Registern DCFxxxALT befinden sich die Uhrzeit-
DCFSTDALT    equ    37      ; und Datumsinfos der vorangegangenen Minute. Diese
DCF TAGALT   equ    38      ; werden zur Ueberpruefung des neuen DCF-Telegramms
DCFMONALT    equ    39      ; benoetigt
DCFJAHRALT   equ    3A
DCFWTAGALT   equ    3B
UHRSEKUNDE   equ    3C      ;Sekundenzaehler der inneren quartzgesteuerten Uhr
UHRMINUTE    equ    3D      ;Minutenzaehler der inneren quartzgesteuerten Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert
UHRSTUNDE    equ    3E      ;Stundenzaehler der inneren quartzgesteuerten Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert
DATUMWTAG    equ    3F      ;Wochentagszaehler der inneren quartzgest. Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert
DATUMTAG     equ    40      ;Tageszaehler der inneren quartzgesteuerten Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert
DATUMMONAT   equ    41      ;Monatszaehler der inneren quartzgesteuerten Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert
DATUMJAHR    equ    42      ;Jahreszaehler der inneren quartzgesteuerten Uhr, wird
; mit jedem gueltigen DCF-dekod. Wert synchronisiert

TEMP1       equ    5A      ;allgemeines Hilfsregister 1
TEMP2       equ    5B      ;allgemeines Hilfsregister 2
TEMP3       equ    5C      ;allgemeines Hilfsregister 3

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

TEMP4      equ    5D      ;allgemeines Hilfsregister 4
TEMP5      equ    5E      ;allgemeines Hilfsregister 5

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

;Register INTCON
T0IF      equ    2        ;TMR0-Interruptflag im INTCON-Register

;***** 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      ;Vorankuendigung der Sommer/Winterzeit-Umstellung
DCFSOMMER  equ    2      ;Sommerzeit
DCFWINTER  equ    3      ;Winterzeit
DCFSCHALTSEK equ    4    ;Vorankuendigung einer zusaetzliche Schaltsekunde

;***** Portbelegung *****
;Port A
DCFINPORT  equ    PORTA
DCFINTRIS  equ    TRISA
DCFIN      equ    5

;***** Konstanten *****
KONSTISR1SEK equ    .250
KONSTD CFLMIN equ    .20
KONSTD CFLMAX equ    .30
KONSTD CFHMIN equ    .40
KONSTD CFHMAX equ    .60

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

;***** Konfigurations-Bits *****
_BODEN_ON      EQU    H'3FFF'
_BODEN_OFF     EQU    H'3FBF'

_CP_ALL        EQU    H'03FF'
_CP_75         EQU    H'17FF'
_CP_50         EQU    H'2BFF'
_CP_OFF        EQU    H'3FFF'

_DATA_CP_ON    EQU    H'3EFF'
_DATA_CP_OFF   EQU    H'3FFF'

_PWRTE_OFF     EQU    H'3FFF'
_PWRTE_ON      EQU    H'3FF7'

_WDT_ON        EQU    H'3FFF'
_WDT_OFF       EQU    H'3FFB'

_LVP_ON        EQU    H'3FFF'
_LVP_OFF       EQU    H'3F7F'

_MCLRE_ON      EQU    H'3FFF'

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

_MCLRE_OFF          EQU      H'3FDF'

_ER_OSC_CLKOUT      EQU      H'3FFF'
_ER_OSC_NOCLKOUT    EQU      H'3FFE'

_INTRC_OSC_CLKOUT   EQU      H'3FFD'
_INTRC_OSC_NOCLKOUT EQU      H'3FFC'

_EXTCLK_OSC         EQU      H'3FEF'
_LP_OSC             EQU      H'3FEC'
_XT_OSC             EQU      H'3FED'
_HS_OSC             EQU      H'3FEE'

__config            _MCLRE_OFF & _PWRTE_OFF & _WDT_OFF & _HS_OSC & _BODEN_OFF & _LVP_OFF

                ORG      0x000
                goto     Beginn
                ORG      0x004
                goto     ISR

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

;*****
;** Tabelle TABSTUNDEN: **
;** Ausgabetablelle der Stunde (mit Umwandlung von 0-23 in 1-12 und BCD-Form) **
;*****
TABSTUNDEN      addwf    PC,f
                dt      0x12
                dt      .1,.2,.3,.4,.5,.6,.7,.8,.9,0x10,0x11,0x12
                dt      .1,.2,.3,.4,.5,.6,.7,.8,.9,0x10,0x11
                dt      .0,.0,.0,.0,.0,.0,.0,.0,.0

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

;*****
;** Interrupt Service Routine: **
;** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **     **
;** Aufruf: **
;** alle 4 ms **
;** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **     **
;** Aufgaben: **
;** + w-Register (=Arbeitsregister) und Status-Register zwischenspeichern (PUSH) **
;** + Zeitbasis fuer 4ms und 1 Sekunde erzeugen **
;** + Das Timer-Interrupt-Flag T0IF wieder loeschen **
;** + 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   ;Zaehregister fuer 1-Sekunden-Zeitbasis um 1
                                                ; vermindern

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

ISRWEITER1
                ;Ende der eigentlichen ISR-Routine
ISRFERTIG     bcf     INTCON,T0IF          ;T0-Interruptflag loeschen

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

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

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

;*****
;** Initialisierung des Prozessor:
;** + Comparatoreingänge auf digitale I/O-Pins umschalten
;** + TMR0-ISR soll alle 4ms aufgerufen werden, daher Vorteiler mit 15 laden (Bei einem
;** 4,096-MHz-Quarz)
;** + Ports: Port A: Ausgänge
;** Port B: Ausgänge
;** Der DCF-Eingang wird separat mit dem Befehl bsf DCFINTRIS,DCFIN als Eingang
;** definiert
;** + Zaehregister fuer den Sekundentakt vorbelegen
;** + DCF-Statusregister (DCFSTATUS) initialisieren (Fehlerflag setzen, alle anderen
;** Flags loeschen)
;** + Pegel von DCFIN einlesen und im Flag DCFPORTNEU des Register DCFSTATUS sichern
;** + Anzeige (Port A und Port B) loeschen
;** + diverse Register vorbelegen
;*****
INIT      clrf      TMR0                ;Timer0 auf 0 voreinstellen

          clrf      PORTA
          movlw    0x07                ;Alle Comparatoreingänge
          movwf   CMCON                ; auf digital I/O umschalten

          bsf      STAT,RP0            ;Registerseite 1
          movlw   b'00000011'         ;interner Takt, Vorteiler = 16 an TMR0
          movwf   OPTREG

          clrf     TRISA                ;Port A und
          clrf     TRISB                ;Port B als Ausgang definieren

          bsf      DCFINTRIS,DCFIN     ;DCFIN als Eingang definieren
          bcf      STAT,RP0            ;Registerseite 0

          movlw   KONSTISR1SEK         ;Zaehregister fuer den Sekundentakt mit
          movwf   ZAEHLERISR1SEK      ; der Konstanten KONSTISR1SEK laden

          clrf     DCFSTATUS           ;DCF-Statusregister initialisieren
          bsf      DCFSTATUS,DCFHEHLER ; Fehlerflag setzen, alle anderen Flags
          ; loeschen

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

          clrf     PORTA                ;Anzeige (Ports A und B) loeschen
          clrf     PORTB

          clrf     UHRSEKUNDE          ;Sekundenzaehler initialisieren (mit 0)
          clrf     UHRMINUTE           ;Minutenzaehler initialisieren (mit 0)
          clrf     UHRSTUNDE          ;Stundenzaehler initialisieren (mit 0)
          clrf     DATUMWTAG           ;Wochentagzaehler initialisieren (mit 0)
          clrf     DATUMTAG            ;Tageszaehler initialisieren (mit 0)
          clrf     DATUMMONAT          ;Monatszaehler initialisieren (mit 0)
          clrf     DATUMJAHR           ;Jahreszaehler initialisieren (mit 0)

          movlw   .99
          movwf   DCFMINALT            ;Vergleichsregister initialisieren (mit 99)
          movwf   DCFSTDALT            ;Vergleichsregister initialisieren (mit 99)
          movwf   DCFRTAGALT           ;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
;**
;** 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 Zaehregister DCFPULS oder
;** das Zaehregister DCFPAUSE um 1 erhöht (inkrementiert), wenn sich der Pegel vom DCF-
;** Eingang zum vorhergehenden Aufruf dieses Unterprogramms nicht veraendert hat. Ist
;** der DCF-Eingang Low (also logisch 0), so wird das Zaehregister DCFPULS um 1 erhöht,

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

**      ist der DCF-Eingang High (also logisch 1), so wird das Zaehregister DCFPAUSE um 1      **
**      erhoeht. Beide Zaehregister koennen 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 zunaechst eine Aus-    **
**      wertung des Zaehregisters DCFPULS. Beinhaltet dieses Zaehregister einen Wert        **
**      zwischen den Konstanten KONSTDCFLMIN und KONSTDCFLMAX, 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 gueltigen Sekunde. Anschliessend wird das Zaehregister DCFPULS geloescht.        **
**      Beinhaltet das Zaehregister DCFPULS einen Wert zwischen den Konstanten KONSTDCFHMIN **
**      und KONSTDCFHMAX, so wurde ein HIGH des DCF-Telegramms ermittelt. Im DCF-Status-    **
**      register (DCFSTATUS) werden daher die Flags DCFHIGH und DCFNEUESEK gesetzt. Das Flag **
**      DCFNEUESEK dient auch hier als Zeichen einer neu empfangenen und gueltigen Sekunde, **
**      und auch das Zaehregister DCFPULS wird anschliessend geloescht. Beinhaltet das      **
**      Zaehregister DCFPULS einen Wert der weder zwischen den Konstanten KONSTDCFLMIN und    **
**      KONSTDCFLMAX noch zwischen den Konstanten KONSTDCFHMIN und KONSTDCFHMAX liegt, so    **
**      handelt es sich um einen Uebertragungsfehler. In diesem Fall wird das Fehlerflag      **
**      (DCFFEHLER) im DCF-Statusregister (DCFSTATUS) gesetzt und auch das Zaehregister      **
**      DCFPULS wird anschliessend geloescht. Das Flag DCFNEUESEK wird hier aber nicht ge- **
**      setzt, da es sich in diesem Fall um keine gueltige Sekunde handelt. Das Fehlerflag    **
**      wird erst bei der Erkennung einer neuen Minute wieder geloescht. Achtung: Solange    **
**      das Fehlerflag gesetzt ist erfolgt keine Auswertung des Zaehregisters DCFPULS.      **
**      Dieses Zaehregister wird aber dennoch bei jeder fallenden Flanke zurueckgesetzt.     **
**      Bei einer steigenden Flanke (der aktuelle Pegel des DCF-Eingangs ist logisch 1) wird **
**      das Zaehregister DCFPAUSE geloescht.                                              **
**      In der 59. Sekunde erfolgt keine Absenkung des Traegers. Da es also in der 59.      **
**      Sekunde keine fallende Flanke gibt, gibt es auch keine steigende Flanke. Das Zaehl- **
**      register DCFPAUSE laeuft ueber, da es nur einen Zaehlbereich von 0 bis 255 besitzt.  **
**      Dieser Ueberlauf wird hier ausgenuetzt. Zur Bestimmung einer neuen Minute bzw. zur   **
**      Bestimmung des Telegrammbeginns. Ist das Fehlerflag gesetzt, so wird es nun ge-    **
**      loescht. War es nicht gesetzt, so wird das Flag DCFNEUEMIN im DCF-Statusregister     **
**      (DCFSTATUS) gesetzt.                                                            **
**
** Anmerkung:
** Die temporaeren Register TEMP1 und TEMP2 werden hier nur als Hilfsregister benoetigt.
** Sie koennen 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 geaendert?
                goto   DCFINCPULSPAUSE ;nein: Puls- bzw. Pause-Zaehler um 1 erhoehen
                btfss  DCFSTATUS,DCFPORTNEU ;ja: -> steigende oder fallende Flanke?
                goto   DCFHLFLANKE ;fallende Flanke
DCFHLFLANKE    clrf   DCFPAUSE ;steigende Flanke: Zaehregister DCFPAUSE
                goto   DCFROUTINEFERTIG ; loeschen 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
                ; (Plausibilitaetspruefung)
                goto   DCFPULSLOESCHEN ;ja: Puls-Zaehler loeschen

DCFPLAUSIBEL   ;Pruefen, ob ein High empfangen wurde
DCFCHECKHIGH   movlw  KONSTDCFHMIN ;unteren High-Grenzwert laden
                subwf  DCFPULS,w
                btfss  STAT,C ;Grenzwert < DCFPULS ?
                goto   DCFCHECKLOW ;nein: kein HIGH-Pegel, pruefen, ob LOW
                movlw  KONSTDCFHMAX ;ja: DCFPULS mit oberen HIGH-Grenzwert
                ; vergleichen

                subwf  DCFPULS,w
                btfsc  STAT,C ;DCF PULS < Grenzwert ?

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

goto    DCFCHECKLOW          ;nein: es handelt sich um kein HIGH-Signal
                                ; pruefen, ob ein LOW empfangen wurde
bcf     DCFSTATUS,DCFLOW     ;ja: Es wurde ein High-Signal empfangen,
                                ; das Flag DCFLOW loeschen,
bsf     DCFSTATUS,DCFHIGH    ; das Flag DCFHIGH setzen,
bsf     DCFSTATUS,DCFNEUESEK ; das Flag DCFNEUESEK setzen
goto    DCFPULSLOESCHEN     ; und Puls-Zaehler loeschen

;Pruefen, ob ein Low empfangen wurde
DCFCHECKLOW movlw KONSTDCFLMIN ;unteren Low-Grenzwert laden
subwf   DCFPULS,w
btfss   STAT,C               ;Grenzwert < DCFPULS ?
goto    DCFSETFEHLER        ;nein: kein LOW-Pegel, Empfangsfehler
movlw   KONSTDCFLMAX        ;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 loeschen,
bsf     DCFSTATUS,DCFNEUESEK ; das Flag DCFNEUESEK setzen
goto    DCFPULSLOESCHEN     ; und den Puls-Zaehler loeschen

;Empfangsfehler
DCFSETFEHLER bsf   DCFSTATUS,DCF FEHLER ;Bei einem Empfangsfehler
bcf     DCFSTATUS,DCFHIGH    ; Fehlerflag (DCF FEHLER) setzen,
bcf     DCFSTATUS,DCFLOW     ; die Flags DCFLOW und DCFHIGH loeschen,
goto    DCFPULSLOESCHEN     ; und den Puls-Zaehler loeschen

;Puls- oder Pausen-Zaehler erhoehen
DCFINCPULSPAUSE btfsc DCFSTATUS,DCFPORTNEU
goto     DCFINCPAUSE

DCFINCPULS incf   DCFPULS,f ;Puls-Zaehler um 1 erhoehen
goto     DCFROUTINEFERTIG

DCFINCPAUSE incfsz DCFPAUSE,f ;Pause-Zaehler erhoehen und auf Ueberlauf
                                ; pruefen
goto     DCFROUTINEFERTIG ;kein Ueberlauf: Unterprogramm verlassen

btfsc   DCFSTATUS,DCF FEHLER ;Ueberlauf: NEUE MINUTE; wenn Fehlerflag
goto    DCFROUTINEW1

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

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

;Puls-Zaehler loeschen
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) hinzufuegen **
;** + Anforderungsflag (Flag DCFNEUESEK) zuruecksetzen **
;** **
;** Vorgehensweise: **
;** Ist das Flag DCFLOW (im Register DCFSTATUS) gesetzt das Carryflag (im SFR STAT) **
;** loeschen, ist aber das Flag DCFHIGH (ebenfalls im Register DCFSTATUS) gesetzt das **
;** Carryflag setzten. Dieses Carryflag nun dem Telegramm hinzufuegen. Dieser Vorgang **
;** erfolgt mit einem so genannten Schiebepfehl. Dabei werden alle Bits des Registers **
;** DCFTELEGRAMM1 auf die naechst hoehere 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 **
;** 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: **

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

; ** Die Flags DCFLOW und DCFHIGH koennen nie gleichzeitig gesetzt sein. Es ist nur **
; ** moeglich, dass entweder nur DCFLOW oder nur DCFHIGH gesetzt ist, aber nie beide **
; ** gleichzeitig. **
; *****
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 ein
                                           goto  DCFSCHIEBE
DCFSCHIEBELOW bcf   STAT,C                ; ja: Dem Telegramm ein High mit Hilfe des Carry
                                           ; hinzufuegen

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 loeschen
             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 groesser als die vorher- **
; ** gehende Minute sein, und die soeben ermittelten Werte fuer die Stunde, Tag, Monat, **
; ** Jahr und Wochentag muss mit den Werten des vorhergehenden Telegramms ueberein- **
; ** stimmen. ACHTUNG: Bei dieser einfachen Ueberpruefung wird der Uebergang von z. B. **
; ** 13:59 auf 14:00 als falsch gewertet, da sich hier die Stunde aendert, und auch die **
; ** Minute um mehr als 1. Wuerden hier alle moeglichen Uebergaenge beruecksichtigt, **
; ** dann wuerde dieses Unterprogramm noch umfangreicher als es ohnehin schon ist. Durch **
; ** diese Vereinfachung dauert die Synchronisierung im schlimmsten Fall nur eine Minute **
; ** laenger. **
; ** + Das alte Datum bzw. die alte Uhrzeit (von der vorhergehenden Minute) durch das neue **
; ** Datum bzw. Uhrzeit ersetzen. Dies ist fuer die Ueberpruefung des naechsten Tele- **
; ** gramms notwendig. **
; ** + Wenn das neu empfangene Datum bzw. die neu empfangene Uhrzeit gueltig ist, die BCD- **
; ** kodierten Datums- bzw. Uhrzeitkomponenten in eine binaere 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) zuruecksetzen **
; ** **
; ** Anmerkung: **
; ** Das temporaere Register TEMP1 wird hier nur als Hilfsregister benoetigt, und kann **
; ** daher auch in anderen Unterprogrammen verwendet werden. **
; *****
DCFUPMINUTE  call   DCFUPSEKUNDE          ;Fehlende Sekunde nachholen

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

             clr   DCFMONBCD            ;Monat zusammensetzen (BCD-Format)
             btfsc DCFTELEGRAMM2,1
             bsf   DCFMONBCD,4
             btfsc DCFTELEGRAMM2,2

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

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           ;Zusaetzliche Informationen
;   btfsc   DCFTELEGRAMM5, 7         ; zusaetzliche Schaltsekunde
;   bsf     DCFZUSATZINFOS, DCFSCHALTSEK
;   btfsc   DCFTELEGRAMM6, 0         ; Sommerzeit
;   bsf     DCFZUSATZINFOS, DCF SOMMER
;   btfsc   DCFTELEGRAMM6, 1         ; Winterzeit
;   bsf     DCFZUSATZINFOS, DCF WINTER
;   btfsc   DCFTELEGRAMM6, 2         ; Vorankuendigung: Wechsel
;   bsf     DCFZUSATZINFOS, DCF SOMWIN ; Sommerzeit <-> Winterzeit
;   btfsc   DCFTELEGRAMM6, 3         ; Reserveantenne
;   bsf     DCFZUSATZINFOS, DCF RESANT

;Neues Telegramm mit dem alten Vergleichen;
bcf      DCFSTATUS, DCF FEHLER      ;zuerst Fehlerflag loeschen

movf     DCFMINALT, w                ;Minuten pruefen
subwf    DCFMINBCD, w
movwf    TEMP1                       ;TEMP1 = DCFMINBCD - DCFMINALT
decfsz   TEMP1, w                    ;TEMP1 - 1 = 0 ?

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

bsf      DCFSTATUS,DCFFEHLER    ;nein: Fehlerflag setzen

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

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

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

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

movf     DCFWTAGALT,w          ;Wochentag pruefen
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 binaer umwandeln
movwf   UHRSTUNDE               ;und die Stunde der mitlaufenden Uhr
;ueberschreiben

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

clrf    UHRSEKUNDE              ;Sekunde loeschen

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

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

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

movf     DCFWOCHENTAG,w         ;DCFWOCHENTAG
movwf   DATUMWTAG

bsf      DCFSTATUS,DCFSYNC

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

DCFUPMINUTEENDE bcf   DCFSTATUS,DCFNEUEMIN ;Anforderungsflag loeschen
                  bcf   DCFSTATUS,DCFNEUEMIN
                  return

;*****
;** INNERE UHR                                           **
;**                                                     **
;** Aufgabe:                                             **
;**   Fuer den Fall dass kein gueltiges DCF-Telegramm empfangen werden kann sorgt dieses **
;**   Unterprogramm dafuer, dass die Uhrzeit trotzdem jede Sekunde aktualisiert wird. Ist **
;**   fuer eine laengere Zeit kein korrekter DCF-Empfang moeglich, so wuerden die Minuten, **
;**   Stunden, Tage usw. stehen bleiben, da ja im Unterprogramm DCFUPMINUTE nur gueltige **
;**   DCF-Telegramme uebernommen werden. **
;**   Dieses Unterprogramm wird also parallel zu den Unterprogrammen fuer die DCF-Dekod- **
;**   ierung aufgerufen. **
;**                                                     **
;** Vorgehensweise:                                     **
;**   Die Sekunden (Register UHRSEKUNDE) um 1 erhoehen. Beinhaltete dieses Register nun den **
;**   Wert 60, so beginnt eine neue Minute. Daher die Sekunden loeschen und die Minuten **
;**   (Register UHRMINUTE) um 1 erhoehen. Beinhaltet dieses Register nun den Wert 60, so **
;**   beginnt eine neue Stunde. Daher die Minuten loeschen und die Stunden (Register **
;**   UHRSTUNDE) um 1 erhoehen. Beinhaltet dieses Register nun den Wert 24, so beginnt ein **
;**   neuer Tag. Daher die Stunden loeschen. Ist das Mitzaehlen des Datums notwendig, so **
;**   muss nun ein Register fuer den Tag (z.B. DATUMTAG) um 1 erhoehrt werden und dieses **
;**   Ueberprueft werden, wobei diese Pruefung 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 muessen! **
;**                                                     **
;** 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               ;Pruefen, ob Sekunde=60
                  goto   INNERERUHRFERTIG
                  clrf   UHRSEKUNDE          ;Wenn Sekunde=60, Sekunde loeschen
                  incf   UHRMINUTE,f         ; und Minute um 1 erhöhen
                  movf   UHRMINUTE,w
                  sublw  .60
                  btfss  STAT,Z               ;Pruefen, ob Minute=60
                  goto   INNERERUHRFERTIG
                  clrf   UHRMINUTE          ;Wenn Minute=60, Minute loeschen
                  incf   UHRSTUNDE,f        ; und Stunde erhoehen
                  movf   UHRSTUNDE,w
                  sublw  .24
                  btfsc  STAT,Z               ;Pruefen, ob Stunde=24
                  clrf   UHRSTUNDE         ;Wenn Stunde=24, Stunden loeschen
INNERERUHRFERTIG
                  return

;***** Demospezifisches Unterprogramm *****

;*****
;** Zeit anzeigen:                                       **
;**                                                     **
;** Dieses Unterprogramm wird vom Hauptprogramm jede Sekunde 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: **
;**   + Stunden Einer:      PORT RB0...RB3 **
;**   + Stunden Zehner:     PORT RA3      (im Format 1-12) **
;**   + Minuten Einer:      PORT RB4...RB7 **
;**   + Minuten Zehner:     PORT RA0...RA2 **
;**                                                     **
;** Anmerkungen:                                         **
;**   + Die Uhrzeit (Minuten und Stunden) ist in den Registern UHRSTUNDE und UHRMINUTE in **
;**   binaerer Form enthalten. Die Sekunde (enthalten im Register UHRSEKUNDE) wird hier **
;**   nicht benoetigt. **
;**   + Die temporaeren Register TEMP4 und TEMP5 werden hier nur als Hilfsregister **
;**   benoetigt, und kann daher auch in anderen Unterprogrammen verwendet werden. **
;**                                                     **
;** Achtung:                                             **

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

; ** Die temporaeren Register TEMP1 bis TEMP3 koennen hier nicht verwendet werden, da **
; ** diese im Unterprogramm BINBCD2 verwendet werden. **
; *****
ANZEIGE      btfs   DCFSTATUS,DCFSYNC      ;Ist die mitlaufende Uhr mit der DCF-Uhr
;                                                    ; synchronisiert?
              goto   ANZEIGEENDE        ;nein: Unterprogramm vorzeitig beenden

              movf   UHRMINUTE,w         ;ja: Die Minuten
              call   BINBCD2             ; in die BCD-Form umwandeln,
              andlw  b'00001111'        ; das niederwertigere Nibble ausmaskieren
              movwf  TEMP4               ; und im Hilfsregister TEMP4
              swapf  TEMP4,f             ; im hoeherwertigeren Nibble sichern

              movf   UHRSTUNDE,w         ;Die Stunden mit Hilfe der Tabelle TABSTUNDEN in
              andlw  b'00011111'        ; die Form 1-12 und in die BCD-Form umwandeln,
              call   TABSTUNDEN         ; das niederwertigere Nibble ausmaskieren,
              andlw  b'00001111'        ; im niederwertigeren Nibble von TEMP4 sichern
              iorwf  TEMP4,w             ; (im hoeherwertigeren Nibble befindet sich ja
;                                                    ; das niederwertigere Nibble der Minuten)
              movwf  PORTB               ; und dies am Port B ausgeben

              movf   UHRMINUTE,w         ;Die Minuten
              call   BINBCD2             ; noch einmal in die BCD-Form umwandeln
              movwf  TEMP4               ; und im Hilfsregister TEMP5 im
              swapf  TEMP4,f             ; niederwertigeren Nibble sichern

              movf   UHRSTUNDE,w         ;Die Stunden
              andlw  b'00011111'        ; noch einmal in die Form 1-12 und in die
              call   TABSTUNDEN         ; BCD-Form umwandeln und in TEMP5 sichern
              movwf  TEMP5

              bcf    TEMP4,3             ;Stunden-Zehner (1 Bit)
              btfs   TEMP5,4             ; in TEMP4,3 kopieren
              bsf    TEMP4,3

              movf   TEMP4,w             ;nur die 4 niederwertigsten Bits von TEMP4
              andlw  b'00001111'        ;
              movwf  PORTA               ; am Port A ausgeben

ANZEIGEENDE  return

; ***** Weitere, allgemeine Routinen *****
; *****
; ** Umwandlung von Binaer nach BCD **
; ** Aufgabe: **
; ** + Die im w-Register stehende Binaer-Zahl (0-255) nach BCD umwandeln. Die einzelnen **
; ** Ziffern werden zunaechst in temporaere Register (TEMP1 bis TEMP3) gespeichert und **
; ** zu Programmende werden die Einer- und Zehnerstelle wieder in das w-Register **
; ** kopiert (Zehnerstelle im H-Nibble, Einerstelle im L-Nibble). Wird die Hunderter- **
; ** stelle benoetigt, so muss das temporaere Register TEMP3 in ein entsprechendes **
; ** Register kopiert werden. **
; **
; ** Vorgehensweise: **
; ** + Von der Zahl wird 10 so oft abgezogen, bis der Rest kleiner 10 ist, bei jeder **
; ** Subtraktion wird TEMP2 (Zehnerstelle) um 1 erhoehrt. Ist Temp2 (Zehnerstelle) = 10 **
; ** wird TEMP3 (Hunderterstelle) um 1 erhoehrt und TEMP2 (Zehnerstelle) zu 0 gemacht. **
; ** + Ergebnis in w **
; **
; ** Anmerkung: Die temporaeren Register TEMP1 bis TEMP3 werden hier nur zum Zwischenspeichern**
; ** benoetigt. Sie koennen daher auch in anderen Unterprogrammen verwendet werden. **
; *****
BINBCD2      clrfs   TEMP3               ;TEMP3 loeschen (Hunderter)
              clrfs   TEMP2               ;TEMP2 loeschen (Zehner)
              movwf  TEMP1               ;w in TEMP1 (Einer)
BinBCDWdh   movlw   .10
              subwf  TEMP1,w             ;TEMP1 - 10 in w
              btfs   STAT,C               ;TEMP1 < 10 ?
              goto   BinBCDfertg         ; ja: Ruecksprung
              movwf  TEMP1               ; nein: (TEMP1 - 10) in TEMP1
              incf   TEMP2,f             ;TEMP2 + 1
              movlw  .10
              subwf  TEMP2,w             ;TEMP2 - 10 in w
              btfs   STAT,C               ;TEMP2 = 10 ?
              goto   BinBCDWdh           ; nein: wiederholen
              clrfs  TEMP2               ; ja: TEMP2 = 0
              incf   TEMP3,f             ; und TEMP3 + 1

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```

        goto    BinBCDWdh

BinBCDfertig  swapf   TEMP2,w           ;TEMP2 in Hi-Nibble
               iorwf   TEMP1,w        ;TEMP1 in Lo-Nibble
               return

;*****
;** Umwandlung von BCD nach Binaer:                               **
;** Aufgabe:                                                       **
;**   + Die im w-Register stehende 2-stellige-BCD-Zahl nach binaer umwandeln **
;**   + Vorgehensweise:                                           **
;**     + Die umzuwandelnde BCD-Zahl im temporaeren Register TEMP1 zwischenspeichern **
;**     + w-Register (Arbeitsregister) loeschen.                   **
;**     + Die BCD-Zahl bitweise ueberpruefen.                     **
;**       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                                             **
;**   + Anmerkung: Das temporaere Register TEMP1 wird hier nur zum Zwischenspeichern benoetigt **
;**     und kann daher auch wo in anderen Unterprogrammen verwendet werden. **
;*****
BCDBIN2      movwf   TEMP1           ;umzuwandelnde BCD-Zahl zwischenspeichern
             clrw    TEMP1           ;Arbeitsregister (w-Register) loeschen
             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

;*****
;*****   Hauptprogramm   *****
;*****
;** Aufgaben des Hauptprogramms:                                   **
;**   + Controller initialisieren (Unterprogramm INIT)             **
;**   + Interrupt freigeben                                       **
;**   + Taetigkeiten/Unterprogramme, die alle 4ms durchgefuehrt werden muessen: **
;**     + Unterprogramm DCFROUTINE aufrufen                         **
;**   + Taetigkeiten/Unterprogramme, die jede Sek. durchgefuehrt werden muessen: **
;**     + Unterprogramm INNEREUHR aufrufen                         **
;**     + Unterprogramm ANZEIGE aufrufen                           **
;**   + Bei einem Minutenwechsel (Flag DCFNEUEMIN ist gesetzt) das Unterprogramm **
;**     DCFUPMINUTE abarbeiten.                                    **
;**   + Bei einem Sekundenwechsel (Flag DCFNEUESEK ist gesetzt) as Unterprogramm **
;**     DCFUPSEKUNDE abarbeiten.                                   **
;*****
Beginn      call    INIT              ;Controller initialisieren

             movlw   b'10100000'     ;Timer0 freigeben durch Setzen von
             movwf   INTCON           ; GIE und T0IE im Register INTCON

HPSCHLEIFE  btfsc   FLAGISRHP,FLAG4MSEK
             goto    HPWEITER1

```

DCF (Dekodierung mit PIC-Mikrocontroller)

```
    btfscl FLAGISRHP, FLAG1SEK
    goto   HPWEITER2

    btfscl DCFSTATUS, DCFNEUEMIN ;Wenn Anforderungsflag DCFNEUEMIN gesetzt
    call   DCFUPMINUTE           ;Unterprogramm DCFUPMINUTE ausfuehren
    btfscl DCFSTATUS, DCFNEUESEK ;Wenn Anforderungsflag DCFNEUESEK gesetzt
    call   DCFUPSEKUNDE         ;Unterprogramm DCFUPSEKUNDE ausfuehren

    goto   HPSCHLEIFE

HPWEITER1    call   DCFROUTINE           ;Taetigkeiten, die alle 4 ms durchgefuehrt
                                ; werden muessen
    bcf    FLAGISRHP, FLAG4MSEK ;Anforderungsflag fuer die 4ms-Aktivitaeten
                                ; zuruecksetzen

    goto   HPSCHLEIFE

HPWEITER2    call   INNEREUHR           ;Taetigkeiten, die jede Sekunde durchgefuehrt
                                ; werden muessen
    call   ANZEIGE

    bcf    FLAGISRHP, FLAG1SEK ;Anforderungsflag fuer 1-Sekunden-Aktivitaeten
                                ; zuruecksetzen

    goto   HPSCHLEIFE

end
```

4.3. Anmerkungen zur Software

Die Software besteht im wesentlichen aus einem kurzen Hauptprogramm, einer kurzen Interrupt-Service-Routine (kurz ISR), die im Abschnitt 3.4 besprochenen Unterprogramme zur DCF-Dekodierung, ein Unterprogramm zur Ausgabe der Uhrzeit (ANZEIGE) und zwei Unterprogramme, welche eine binäre Zahl in BCD-Form umwandeln bzw. eine BCD-Zahl in die binäre Form.

Die ISR wird alle 4ms aufgerufen und besitzt „nur“ die Aufgabe zwei Zeitbasen zu erzeugen, indem sie zwei Botschaftsflags für das Hauptprogramm erzeugt. Eine Zeitbasis alle 4ms und die zweite Zeitbasis jede Sekunde. Da die ISR alle 4ms aufgerufen wird ist die 4-ms-Zeitbasis besonders einfach. Es muss nur bei jedem Aufruf ein Flag gesetzt werden, und zwar das Flag FLAG4MSEK 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:

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

Also ein ISR-Aufruf alle 4000 μ s, was gleichbedeutend mit 4ms ist.

Das Hauptprogramm befindet sich nach der Initialisierung (Unterprogramm INIT) und der Interruptfreigabe 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.

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 3.4.1)
- Tätigkeiten und/oder Unterprogramme, die jede Sekunde durchgeführt werden müssen
- Unterprogramm INNEREUHR aufrufen (siehe Abschnitt 3.4.4)
- Unterprogramm ANZEIGE aufrufen
- Bei einem Minutenwechsel (Flag DCFNEUEMIN ist gesetzt) das Unterprogramm DCFUPMINUTE abarbeiten (siehe Abschnitt 3.4.3)
- Bei einem Sekundenwechsel (Flag DCFNEUESEK ist gesetzt) das Unterprogramm DCFUPSEKUNDE abarbeiten (siehe Abschnitt 3.4.2)

Das Unterprogramm INIT dient zur Initialisierung des Controller. Hier werden unter anderem die Ports konfiguriert (Port dient als Eingang oder als Ausgang), der oder die Timer eingestellt usw. Dieses Unterprogramm ist vom Controllertyp abhängig und je nach Anwendung mehr oder weniger umfangreich. Hier muss zusätzlich der Startzustand des DCF-Eingangs eingelesen werden. Siehe auch Abschnitt 3.3 (Initialisierung)

Das Unterprogramme ANZEIGE dient nur zur Ausgabe der Uhrzeit (hier der Stunden und Minuten) an den Ports. Die Aufgabe dieses Unterprogramms besteht darin die in den Registern UHRSTUNDE und UHRMINUTE enthaltene Uhrzeit an die Hardware anzupassen.