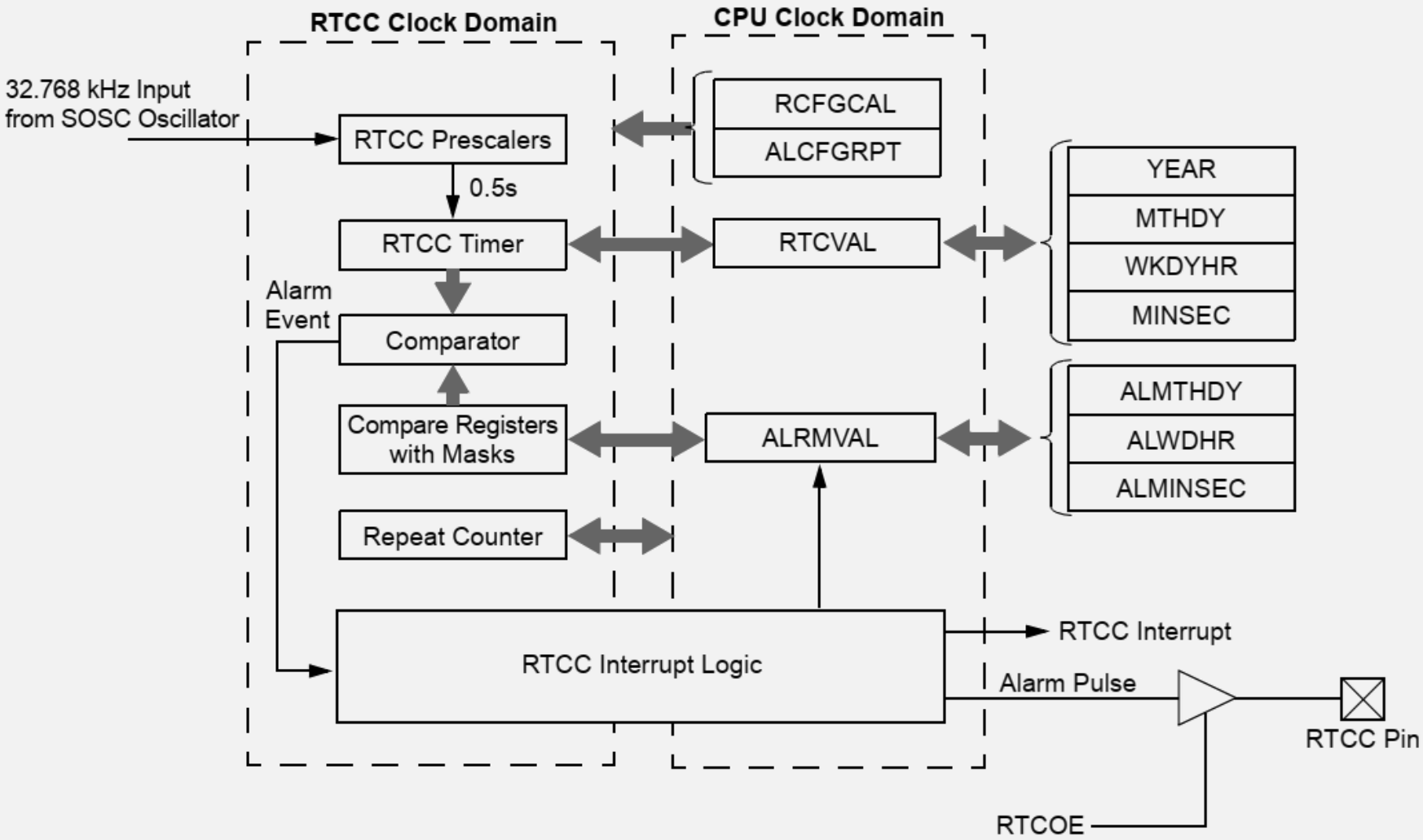
Real-Time Clock and Calendar (RTCC) modul alkalmazása

RTCC modul jellemzői

- Másodperc, perc, óra, nap, hónap és év tárolására és léptetésére alkalmas modul,
- külső órakvarccal szükséges hozzá
- Sleep módban is működik
- 24 órás időformátum
- Naptár 2000-2099-ig
- Szökőévet követi
- Az értékeket BCD formában tárolja
- Rendelkezik Alarm funkcióval
  - interruptot generálhat
  - kimente lehet egy RTCC lábnak
- Kalibrálható az órajele a pontos értékre

RTCC modul felépítése



RCFGCAL: RTCC Calibration and Configuration Register

R/W-0	U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0
RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

RTCEN: RTCC Enable bit
1 = RTCC module is enabled
0 = RTCC module is disabled

RTCWREN: RTCC Value Registers Write Enable bit
1 = RTCVALH and RTCVALL registers can be written to by the user
0 = RTCVALH and RTCVALL registers are locked out from being written to by the user

RTCSYNC: RTCC Value Registers Read Synchronization bit
1 = RTCVALH, RTCVALL and ALCFGRPT registers can change while reading due to a rollover ripple resulting in an invalid data read. If the register is read twice and results in the same data, the data can be assumed to be valid.
0 = RTCVALH, RTCVALL or ALCFGRPT registers can be read without concern over a rollover ripple

HALFSEC: Half-Second Status bit
1 = Second half period of a second
0 = First half period of a second

RTCOE: RTCC Output Enable bit
1 = RTCC output enabled
0 = RTCC output disabled

RTCPTR<1:0>: RTCC Value Register Window Pointer bits
Points to the corresponding RTCC Value registers when reading RTCVALH and RTCVALL registers;
the RTCPTR<1:0> value decrements on every read or write of RTCVALH until it reaches '00'.
RTCVAL<15:8>:
00 = MINUTES
01 = WEEKDAY
10 = MONTH
11 = Reserved
RTCVAL<7:0>:
00 = SECONDS
01 = HOURS
10 = DAY
11 = YEAR

RCFGCAL: RTCC Calibration and Configuration Register

R/W-0	U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0
RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

CAL<7:0>: RTC Drift Calibration bits
01111111 = Maximum positive adjustment; adds 508 RTC clock pulses every one minute
...
00000001 = Minimum positive adjustment; adds 4 RTC clock pulses every one minute
00000000 = No adjustment
11111111 = Minimum negative adjustment; subtracts 4 RTC clock pulses every one minute
...
10000000 = Maximum negative adjustment; subtracts 512 RTC clock pulses every one minute

PADCFG1: Pad Configuration Control Register

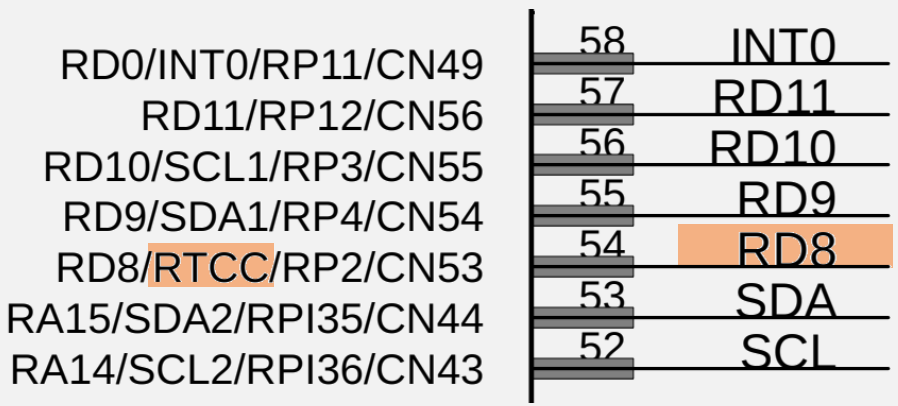
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

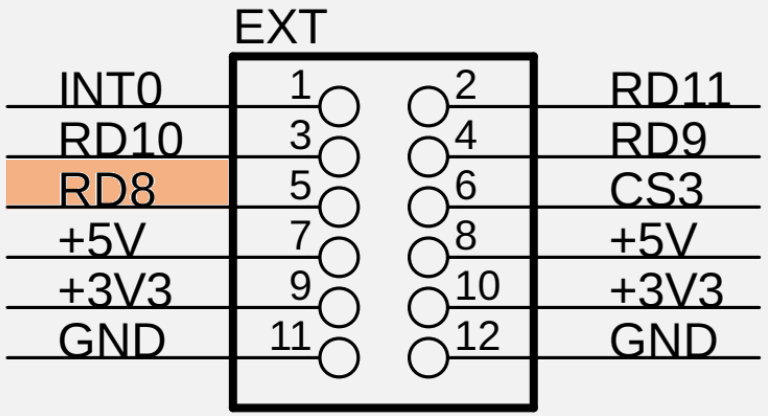
U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	RTSECSEL	PMPTTL
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

RTSECSEL: RTCC Seconds Clock Output Select bit
1 = RTCC seconds clock is selected for the RTCC pin - Másodpercenként átkapcsol
0 = RTCC alarm pulse is selected for the RTCC pin - Riasztáskor kapcsol át

PMPTTL: PMP Module TTL Input Buffer Select bit
1 = PMP module inputs (PMDx, PMCS1) use TTL input buffers
0 = PMP module inputs use Schmitt Trigger input buffers

Az RTCC modul kimeneti lába a uMOGI2 panelen:





ALCFGRPT: Alarm Configuration Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

ALRMEN: Alarm Enable bit
1 = Alarm is enabled (cleared automatically after an alarm event whenever ARPT<7:0> = 00h and CHIME = 0)
0 = Alarm is disabled

CHIME: Chime Enable bit
1 = Chime is enabled; ARPT<7:0> bits are allowed to roll over from 00h to FFh
0 = Chime is disabled; ARPT<7:0> bits stop once they reach 00h

AMASK<3:0>: Alarm Mask Configuration bits
0000 = Every half second
0001 = Every second
0010 = Every 10 seconds
0011 = Every minute
0100 = Every 10 minutes
0101 = Every hour
0110 = Once a day
0111 = Once a week
1000 = Once a month
1001 = Once a year (except when configured for February 29th, once every 4 years)
Other Reserved – do not use

ALRMPTR<1:0>: Alarm Value Register Window Pointer bits
Points to the corresponding Alarm Value registers when reading ALRMVALH and ALRMVALL registers; the ALRMPTR<1:0> value decrements on every read or write of ALRMVALH until it reaches '00'.
ALRMVAL<15:8>:
00 = ALRMMIN
01 = ALRMWD
10 = ALRMMNTH
11 = Unimplemented
ALRMVAL<7:0>:
00 = ALRMSEC
01 = ALRMHR
10 = ALRMDAY
11 = Unimplemented

ARPT<7:0>: Alarm Repeat Counter Value bits
11111111 = Alarm will repeat 255 more times
...
00000000 = Alarm will not repeat
The counter decrements on any alarm event. The counter is prevented from rolling over from 00h to FFh unless CHIME = 1.

Az RTCC modul használata

- A modult használatához engedélyezni kell a másodlagos órajelforrást:

```
__builtin_write_OSCCONL(OSCCON | 0x02); // Órakvarc engedélyezése
```

- A modul írhatósága egy szekvencia elvégzése után történhet:

```
    //A modul írhatóságának az engedélyezése
    NVMKEY = 0x55;
    NVMKEY = 0xAA;

    RCFGCALbits.RTCWREN = 1;        // feloldjuk az írásvédelmet
```

- Használhatjuk a következő rövidítést:

```
    //A modul írhatóságának az engedélyezése
    __builtin_write_RTCWEN();    // feloldjuk az írásvédelmet
```

Az RTCC modul használata

- Az értékeket az RTCVAL regiszteren keresztül tölthetjük fel, az RTCPTR értékének megfelelően:

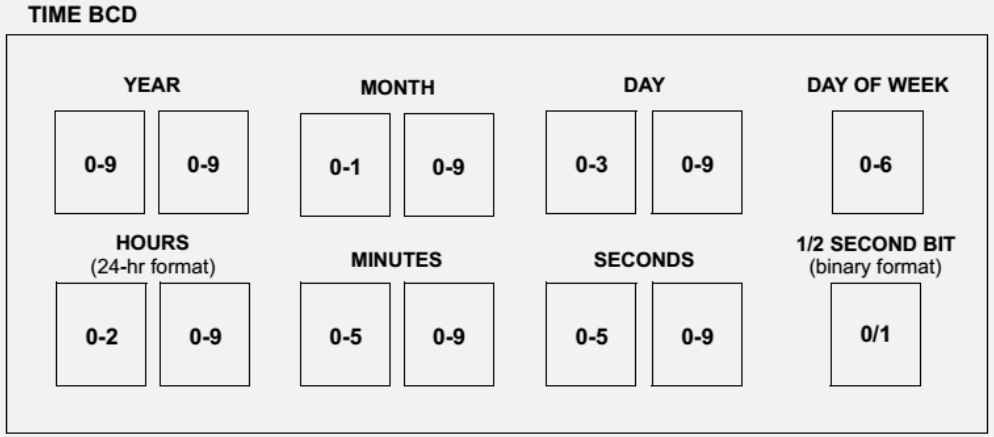
RTCPTR <1:0>	RTCC Value Register Window	
	RTCVAL<15:8>	RTCVAL<7:0>
00	Minutes	Seconds
01	Weekday	Hours
10	Month	Day
11	—	Year

- Az RTCPTR értéke minden egyes RTCVAL értékadás alkalmával egyel csökken (ezért érdemes 3-ról indulni)
- Az értékek feltöltése után bekapcsoljuk a modult és visszaállítjuk az írásvédelmet:

```
RCFGCALbits.RTCEN = 1;      // bekapcsoljuk a modult
RCFGCALbits.RTCWREN = 0;    // visszaállítjuk az írásvédelmet
```

Az RTCC modul használata

- Az értékek binárisan kódolt decimális (BCD) alakban vannak tárolva:




```
typedef struct _rtcTime   // Naptár struktúra
{
        uint8_t year;     // 0x00-0x99
        uint8_t month;    // 0x01-0x12
        uint8_t day;      // 0x01-0x31
        uint8_t weekday;  // 0x00-0x06 Hétfő 0, Vasárnap 6
        uint8_t hour;     // 0x00-0x23
        uint8_t minute;   // 0x00-0x59
        uint8_t second;   // 0x00-0x59
} rtcTime;
```

Az idő beállítása

```
// Naptár modul beállítása
void rtcSet(rtcTime ido)
{
    // A modul írhatóságának az engedélyezése
    __builtin_write_RTCWEN();    // feloldjuk az írásvédelmet
    RCFGCALbits.RTCEN = 0;       // kikapcsoljuk a modult

    // Beállítjuk az időt
    RCFGCALbits.RTCPTR = 3;                    // ráállunk az évre
    RTCVAL = ido.year;                         // év (#0x00YY)
    RTCVAL = (ido.month << 8) + ido.day;       // hónap és nap (#0xMMDD)
    RTCVAL = (ido.weekday << 8) + ido.hour;    // hétköznap és óra (#0x0WHH)
    RTCVAL = (ido.minute << 8) + ido.second; // perc és másodperc (#0xMMSS)

    RCFGCALbits.RTCEN = 1;      // bekapcsoljuk a modult
    RCFGCALbits.RTCWREN = 0;    // visszaállítjuk az írásvédelmet
}
```

Az idő lekérdezése

```
// Idő lekérdezése
rtcTime rtcGet()
{
        rtcTime ido;
        while(RCFGCALbits.RTCSYNC);     // várakozás az RTCSYNC 0-ra

        RCFGCALbits.RTCPTR = 3;         // ráállunk az évre
        ido.year = RTCVAL;              // év kiolvasása

        uint16_t month_date = RTCVAL;   // hónap és nap
        ido.month = month_date >> 8;
        ido.day = month_date;

        uint16_t wday_hour = RTCVAL;    // hétköznap és óra
        ido.weekday = wday_hour >> 8;
        ido.hour = wday_hour;

        uint16_t min_sec = RTCVAL;      // perc és másodperc
        ido.minute = min_sec >> 8;
        ido.second = min_sec;

        return ido;
}
```

Gyakorlás

Feladat

Állítsa be a pontos időt és jelenítse meg az LCD-n!

Az RTCC modul használata

```
int main() {
    // Órajelforrás beállítása

    // Órakvarc engedélyezése
    __builtin_write_OSCCONL(OSCCON | 0x02);

    // Naptár modul beállítása 2023.10.17 (Kedd) 20:53:56
    rtcTime ido = {0x23,0x10,0x17,0x01,0x20,0x53,0x56};
    rtcSet(ido);              // idő beállítása

    while(1)
    {   // ciklikus feladatok elvégzése
        LEDR = !LEDR;
        DELAY_MS(500);
        ido = rtcGet();           // idő lekérdezése
        lcdClear();
        sprintf(LCD,"%02x:%02x:%02x",ido.hour, ido.minute, ido.second);
        lcdPutStr(LCD);
    }
 return (0);
}
```

Az Alarm funkció megvalósítása

- Egy előre beállított időpontot hasonlít össze az RTCC idejével
- Az időpontot az ALRMVAL lapozható regiszteren keresztül lehet feltölteni:

ALRMPTR <1:0>	Alarm Value Register Window	
	ALRMVAL<15:8>	ALRMVAL<7:0>
00	Minutes	Seconds
01	Weekday	Hours
10	Month	Day
11	—	Year

- Az ALPMPTR értéke mindenegyes ALRMVAL értékadás alkalmával egyel csökken.

Riasztási időpont maszkolása

- Az AMSK segítségével lehet beállítani, hogy mit vegyen figyelembe az összehasonlításkor:

Alarm Mask Setting (AMASK<3:0>)	Day of the Week	Month	Day	Hours	Minutes	Seconds
0000 – Every half second 0001 – Every second						
0010 – Every 10 seconds						s
0011 – Every minute						s s
0100 – Every 10 minutes					m	s s
0101 – Every hour					m m	s s
0110 – Every day				h h	m m	s s
0111 – Every week	d			h h	m m	s s
1000 – Every month			d d	h h	m m	s s
1001 – Every year(1)		m m	d d	h h	m m	s s

Riasztási időpont maszkolása

- Az AMSK segítségével lehet beállítani, hogy mit vegyen figyelembe az összehasonlításkor:

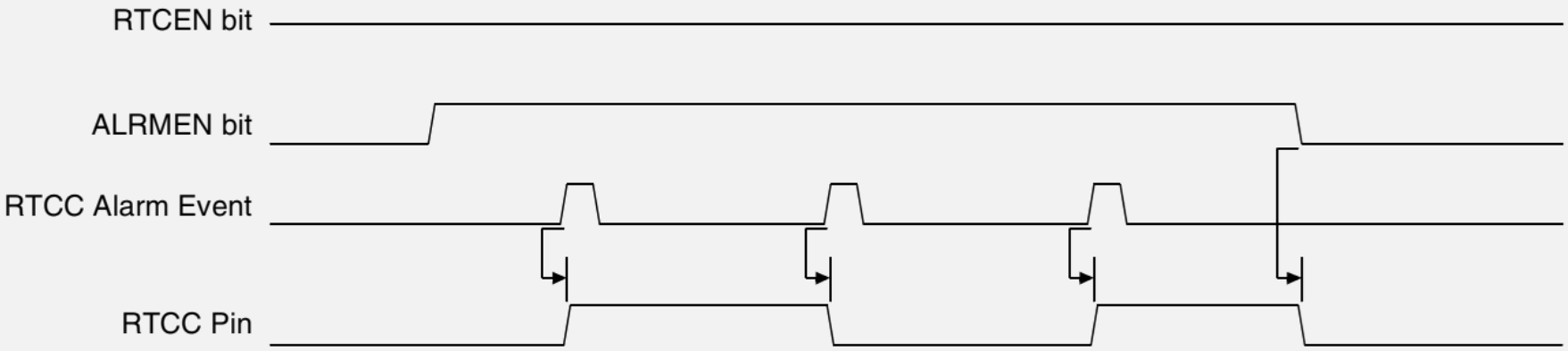
```
// Alarm maszkolása
#define RTCC_MASK_EVERY_HALF_SECOND  0  // fél másodpercenként
#define RTCC_MASK_EVERY_SECOND       1  // másodpercenként
#define RTCC_MASK_EVERY_10_SECOND    2  // 10 másodpercenként
#define RTCC_MASK_EVERY_MINUTE       3  // percenként
#define RTCC_MASK_EVERY_10_MINUTE    4  // 10 percenként
#define RTCC_MASK_EVERY_HOUR         5  // óránként
#define RTCC_MASK_EVERY_DAY          6  // naponta
#define RTCC_MASK_EVERY_WEEK         7  // hetente
#define RTCC_MASK_EVERY_MONTH        8  // havonta
#define RTCC_MASK_EVERY_YEAR         9  // évente
```

Ismétlés beállítása

- Az Alarm funkció végrehajtása:
  - Egyszer történik meg
    - CHIME = 0 és
    - ARPT<7:0> = 0 (Alarm Repeat Counter)
  - Megadott számú lefutás
    - CHIME = 0 és
    - Alarm Repeat Counter értéke a lefutások száma, amely minden egyes alkalommal csökken, 0-nál kikapcsolja az Alarm funkciót
  - Végtelenszer végrehajtódik
    - CHIME = 1
    - Ilyenkor is csökken az ARPT értéke, de 0x00 után 0xFF-el folytatódik

Riasztás generálása

- Minden riasztás hatására az RTCC láb átvált
  - ha RTSECSEL = 0 (a láb az RTCC riasztásához van hozzárendelve)
  - ha RTCOE = 1 (a kimenet engedélyezve van)



Az Alarm funkció beállítása

```
// Alarm beállítása
void rtcAlarm(rtcTime ido, uint8_t mask)
{
    ALCFGRPTbits.ALRMEN=0;        // kikapcsoljuk a modult

    // Beállítjuk az időt
    ALCFGRPTbits.ALRMPTR = 3;     // ráállunk az évre
    ALRMVAL = ido.year;                          // év (#0x00YY)
    ALRMVAL = (ido.month << 8) + ido.day;        // hónap és nap (#0xMMDD)
    ALRMVAL = (ido.weekday << 8) + ido.hour;     // hétköznap és óra (#0x0WHH)
    ALRMVAL = (ido.minute << 8) + ido.second;    // perc és másodperc (#0xMMSS)

    ALCFGRPTbits.AMASK = mask;     // összehasonlítás maszkolása

    ALCFGRPTbits.CHIME = 1;        // végtelen ismétlés
    ALCFGRPTbits.ALRMEN = 1;       // bekapcsoljuk a modult
}
```

Gyakorlás

Feladat

Állítsa be az óra végét ébresztési időpontnak! Ha bejelez, akkor világítson a kék led, és menjünk haza!

Az Alarm funkció megvalósítása

```
int main() {

    // Órajelforrás beállítása

    // Órakvarc engedélyezése
    __builtin_write_OSCCONL(OSCCON | 0x02);

    //Naptár modul beállítása 2024.10.16 (Szerda) 9:53:56
    rtcTime ido = {0x24,0x10,0x16,0x02,0x09,0x53,0x56};
    rtcSet(ido);         // idő beállítása

    ido.minute = 0x55;
    ido.second = 0x00;
    rtcAlarm(ido,0b0110);         // alarm indítása 9:55:00-kor naponta
    RCFGCALbits.RTCOE = 1;        // a kimenet engedélyezése

    while(1);
    return (0);
}
```

Az Alarm funkció interruptja

- Interrupt engedélyezése és a flag törlése:

```
    _RTCIE = 1;                 // RTCC interrupt engedélyezése
    _RTCIF = 0;                 // RTCC interrupt flag törlése
```

- ISR:

```
void _ISR _RTCCInterrupt(void)
{
    // utasítások
    _RTCIF = 0;                 // RTCC interrupt flag törlése
}
```

Ha menet közben változtatunk időpontot, vagy riasztási beállítást, akkor előtte kapcsoljuk ki az interruptot a téves riasztások elkerülése végett!