```
/*****************************************************************************\
ABClock - QLOCKTWO remake
D'après BIEGER&FUNK http://www.qlocktwo.com/
QUARTZ 20MHZ
OSC _HS
RA3 as MCLR (prévoir une charge RC pour l'allumage)


changelog
100819 : Ajout de la gestion HARD du 74HC4017 et 4094
100817 : Changement affichage, maintenant par TIMER0 à la freq de 450Hz (50FPS)
100817 : mise en place de 2 boutons à "l'arrache" pour mise à l'heure
100815 : ajout gestion heures avec PCF8385

\*****************************************************************************/



#include "prepro.c"
#define debug   false

/* TODO LIST *\

- Arriver à faire fonctionner l'interruption du PCF toute les minutes et non secondes...
- Revoir l'init du flag de l'interruption du PCF!!!!!
- Prévoir l'algo pour les boutons +/-   mnt+=i²/4 (un truc du genre sans être trop hard)



\*************/









/* LIAISONS I2C */
sbit SCL at RB1_bit;            // Horloge @ 100Khz
sbit SDA at RB0_bit;            // Donnée séries
/* LIAISONS REGISTRE A DECALAGE */
sbit rDta at RD4_bit;           // Donnée séries
sbit rClk at RD5_bit;           // Horloge
sbit rStb at RD6_bit;           // Standby mode    USELESS pour le moment
sbit rOe  at RD7_bit;           // Output enable   USELESS pour le moment
/* LIAISONS COMPTEUR */
sbit cClk at RD2_bit;           // Horloge


unsigned char bp1OldFlag = 0;



char PB_IN at LATB;
```

```
56  unsigned unsigned char PB_OUT at PORTB;
57  unsigned int matrice[9];
58  unsigned char INTi = 0;
59  unsigned int TMR0nbInt;
60  unsigned char _print_line = 0;
61  unsigned char _print_frame = 0;
62  char counterStart = 0;
63
64  unsigned char timechange = false;
65  char j = 0;
66  struct times{
67          char sec;
68          char mns;
69          char hrs;
70          char mnt;               //0 = AM / 1 = PM
71  }now,nowm;
72  char newMnt = 0;
73  unsigned char buff = 0;
74
75
76
77
78  unsigned char flag =  FLAG_RTC_INIT;
79  /* FUNCTIONS  PCF */
80  void PCF_READ_TIME();
81  void PCF_WR_CONFIG(char adress, char config);
82  void PCF_UP_TIME(char dir);
83  void unsetRtcFlag();
84
85  /* FUNTIONS  MATRICE */
86  void iniMatrice();
87  void razMatrice();
88  void updateMatrice();
89  void updateLine(char ligne, int entry);
90
91  /* FUNCTIONS REGISTRE A DECALAGE */
92  void sendLine(char line);       /*Mouai... a modifier rapidement... */
93
94  void main()
95  {
96          #if(debug)
97          //time lecture et mémoire pour SoftDebug
98          now.sec = 30;
99          now.mns = 30;
100         now.hrs = 12;
101         now.mnt = 0;
102
103         nowm.sec = 20;
104         nowm.mns = 40;
105         nowm.hrs = 11;
106         nowm.mnt = 0;
107         #endif
108
109
110
111         PORTD = 0x00;
112         TRISD = PORT_OUT;
```

```
113         LATD = 0x00;
114         PORTD = 0x00;
115
116         ADCON0 = 0x00;                          //      A/D         desactivé
117         ADCON1 = 0x0F;                          //      NO A/D entry, all in TOR
118         CMCON = 0x07;                           //      Comparateur désactivé
119         PB_IN = 0x00;                           //      INIT du portB à 0
120         TRISB = PORT_IN;                    //      Port B en INPUT
121         PORTB = 0x00;
122
123         IPEN_bit = 0;                           //      Pas d'interruption prioritaire
124         RBPU_bit = 1;                           //      PORTB PULL'UP diseable
125         INTEDG2_bit = 0;                    //      Actif sur front descandant
126         INT2IP_bit = 0;
127         INT2E_bit = 1;
128
129         T0CON  = 0b11000100;                //          Set TMR0 in 8bit mode, assign
    prescaler to TMR0
130         TMR0L  = 0x80;                          //          Timer0 initial value
131          TMR0IE_bit = 1;
132
133
134         RBPU_bit = 1;                           //      PORTB PULL'UP diseable
135         RBIF_bit = 0;                       //      INIT du flag à 0
136         RBIE_bit = 1;                       //      PORTB<7:4> int on change enable
137
138
139
140         PEIE_bit = 1;
141         //GIE_bit = 1;                          //      Activation des interruptions
    non masqué
142
143
144
145           razMatrice();
146         iniMatrice();
147         Delay_ms(1000);
148         for(;;)
149         {
150                 switch(flag)
151                 {
152                         case FLAG_RTC_INIT :  //DEVRA SORTIE DU SWITCH
153                         {
154                                 I2C1_Init(100000);
155                                 #if debug == 0
156                                 /* trouver comment gérer les ack en soft...*/
157                                 PCF_WR_CONFIG(PCFaddrReg,0b10000100);
    //Stop du count + alarm enable
158                                 PCF_WR_CONFIG(PCFaddrParam,0b11001001);
    //Alarm&timer inter enable + timer alarm enable + function 1/100 de sec
159
160
161                                 PCF_WR_CONFIG(2,defSec);                        //
    mise à l'heures des sec à 30
162                                 PCF_WR_CONFIG(3,defMns);                        //
    mise à l'heure des mns à 30
163                                 PCF_WR_CONFIG(4,(defHrs | 0b11000000));         //
```

```
    mise à l'heur des hrs à 8 + mode 12 + mnt AM
164
165                                 PCF_WR_CONFIG(PCFaddrReg,0b00000100);           //
    start du count
166
167                                 PCF_READ_TIME();
168                                 #endif
169                                 flag = FLAG_IDLE;
170                                 GIE_bit = 1;
171                                 break;
172                             }//ENDIF FLAG_RTC_INIT
173                             case FLAG_RTC_RD :
174                             {
175                             swInt;
176                                 unsetRtcFlag();
177                                 PCF_READ_TIME();
178                                 flag = FLAG_IDLE;
179                             swInt;
180                                 break;
181                             }//ENDIF FLAG_RTC_RD
182                             case FLAG_RTC_INTR :
183                             {
184                             swInt;
185                                 unsetRtcFlag();
186                                 flag = FLAG_IDLE;
187                             swInt;
188                                 break;
189                             }//ENDIF FLAG_RTC_INTR
190                             case FLAG_IDLE : // affichage
191                             {
192                                /* char m;
193                                 for(m=0;m<9;m++)
194                                 {
195                                        sendLine(m);
196                                         NOP;
197                                         rStb = 1; NOP; rStb = 0;
198                                 }
199                                 //Delay_ms(100);
200                                 RD1_bit = ~RD1_bit;
201                                 NOP;
202                                 cClk = 1; NOP; cClk = 0; // on balance la sauce des
    bascules en sortie!
203                                 */
204                                 break;
205                             }//ENDIF FLAG_IDLE
206                             case FLAG_BOUTON :
207                             {
208                             swInt;
209                                 //unsetRtcFlag();
210                                 if(Button(&PORTB,6,10,1)) PCF_UP_TIME(1);
211                                 if(Button(&PORTB,7,10,1)) PCF_UP_TIME(0);
212                                 flag = FLAG_IDLE;
213                             swInt;
214                                 break;
215                             }//ENDIF FLAG_BOUTON
216                             case FLAG_PRINT_LINE  :
217                             {
```

```
218                                 stopInt;
219                                 sendLine(_print_line);
220                                 NOP; rStb = 1; NOP; rStb = 0;
221                                 if(counterStart)
222                                 {
223                                         cClk = 1; NOP; cClk = 0;
224                                 }else counterStart = true;
225                                 _print_line++;
226                                 if(_print_line >= 9) _print_line = 0;
227                                 flag = FLAG_IDLE;
228                                 startInt;
229                                 break;
230                         }//ENDIF FLAG_PRINT_LINE
231                 }
232         }
233 }
234
235 void PCF_READ_TIME()
236 {
237         char bufferTime[3];
238
239         #if debug == 0
240         I2C1_Start();
241         I2C1_Wr(PCFaddrWr);             //Addresse du PCF8586
242         I2C1_Wr(2);                     //Adresse de départ de lecture (seconde)
243         I2C1_Repeated_Start();
244         I2C1_Wr(PCFaddrRd);
245         bufferTime[0] = I2C1_Rd(1); //secondes
246         bufferTime[1] = I2C1_Rd(1); //minutes
247         bufferTime[2] = I2C1_Rd(1); //hours
248         I2C1_Stop();
249
250         now.sec = Bcd2Dec(bufferTime[0]);
251         now.mns = Bcd2Dec(bufferTime[1]);
252         now.hrs = Bcd2Dec(bufferTime[2] & 0x3F);
253
254         /* LE JOUR OU L'INTERRUPTION TOUTE LES MINUTES DU PCF FONCTIONNERA
255         JE POURRAI PEUT ETRE DECOMMENTER... */
256         //if(((bufferTime[2] & 0b01000000)) >= 1) now.mnt = 0;
257         //else now.mnt = 1;
258         //if((bufferTime[2] & 0b10000000) == 0)PCF_WR_CONFIG(4,0b10000000 |
    bufferTime[2]);
259
260
261         #endif
262
263         if(now.sec != nowm.sec) {nowm.sec = now.sec;}
264         if(now.mns != nowm.mns) {nowm.mns = now.mns; timeChange = true;}
265         if(now.hrs != nowm.hrs) {nowm.hrs = now.hrs; timeChange = true;}
266         //if(now.mnt != nowm.mnt) {nowm.mnt = now.mnt; timeChange = true;}
267         if(timeChange)
268         {
269                 updateMatrice();
270                 timeChange = 0;
271         }
272         flag = FLAG_IDLE;
273
```

```
274  }
275  void PCF_WR_CONFIG(char adress, char config)
276  {
277          I2C1_Start();
278          I2C1_Wr(PCFaddrWr);
279          I2C1_Wr(adress);
280          I2C1_Repeated_Start();
281          I2C1_Wr(PCFaddrWr);
282          I2C1_Wr(config);
283          I2C1_Stop();
284  }
285
286  void unsetRtcFlag()
287  {
288          #if debug == 0
289          PCF_WR_CONFIG(PCFaddrReg,0b00000100);
290          #endif
291  }
292
293  void iniMatrice()
294  {
295     #if ledStateOn == 1
296     matrice[ligne1] = il | est;
297     #elif ledStateOn == 0
298     matrice[ligne1] = ~(il | est);
299     #endif
300  }
301  void razMatrice()
302  {
303        unsigned char i;
304        for(i=0;i<9;i++)
305        {
306           #if ledStateOn == 0
307           matrice[i] = 0xFFFF;
308           #elif ledStateOn == 1
309           matrice[i] = 0x0000;
310           #endif
311        }
312  }
313
314  void updateLine(char ligne, int entry)
315  {
316       #if ledStateOn == 1
317       matrice[ligne] |= entry;
318       #elif ledStateOn == 0
319       matrice[ligne] &= ~(entry);
320       #endif
321  }
322  /*************************************************************************/
323  void updateMatrice()
324  {
325    if(nowm.mns<=32)
326    {
327        //Verif pour affichage de "heure" "heures" ou rien
328        if(nowm.hrs == 1) updateLine(ligne6,heure);
329        else if(nowm.hrs >1 && nowm.hrs != 12) updateLine(ligne6,heures);
330        // Switch de chaque heure possible (en mode AM/PM)
```

```
331         switch(nowm.hrs)
332         {
333              case 1 : updateLine(ligne4,une); break;
334              case 2 : updateLine(ligne3,deux); break;
335              case 3 : updateLine(ligne2,trois); break;
336              case 4 : updateLine(ligne4,quatre); break;
337              case 5 : updateLine(ligne3,cinq); break;
338              case 6 : updateLine(ligne3,six); break;
339              case 7 : updateLine(ligne2,sept); break;
340              case 8 : updateLine(ligne2,huit); break;
341              case 9 : updateLine(ligne4,neuf); break;
342              case 10: updateLine(ligne5,dix); break;
343              case 11: updateLine(ligne1,onze); break;
344              case 12:
345              {
346                   if(nowm.mnt == 0) updateLine(ligne5,midi);
347                   else updateLine(ligne5,minuit);
348              }
349         }
350         if(nowm.mns > 2 && nowm.mns <= 7)        updateLine(ligne8,mcinq);
351         else if(nowm.mns > 7 && nowm.mns <= 12)    updateLine(ligne7,mdix);
352         else if(nowm.mns >12 && nowm.mns <= 17)    {updateLine(ligne9,quart);updateLine
    (ligne6,et);}
353         else if(nowm.mns >17 && nowm.mns <= 22)    updateLine(ligne8,vingt);
354         else if(nowm.mns >22 && nowm.mns <= 27)    {updateLine(ligne8,vingt);updateLine
    (ligne8,tirret);updateLine(ligne8,mcinq);}
355         else {updateLine(ligne6,et); updateLine(ligne9,demie);}
356       }else
357       {
358         //Verif pour affichage de "heure" "heures" ou rien
359         if(nowm.hrs == 12) updateline(ligne6,heure);
360         else if(nowm.hrs <12 && nowm.hrs != 11) updateLine(ligne6,heures);
361         // Switch de chaque heure possible (en mode AM/PM)
362         switch(nowm.hrs)
363         {
364              case 0 : updateLine(ligne4,une); break;
365              case 1 : updateLine(ligne3,deux); break;
366              case 2 : updateLine(ligne2,trois); break;
367              case 3 : updateLine(ligne4,quatre); break;
368              case 4 : updateLine(ligne3,cinq); break;
369              case 5 : updateLine(ligne3,six); break;
370              case 6 : updateLine(ligne2,sept); break;
371              case 7 : updateLine(ligne2,huit); break;
372              case 8 : updateLine(ligne4,neuf); break;
373              case 9 : updateLine(ligne5,dix); break;
374              case 10: updateLine(ligne1,onze); break;
375              case 11:
376              {
377                   if(nowm.mnt == 0) updateLine(ligne5,midi);
378                   else updateLine(ligne5,minuit);
379              }
380         }
381         updateLine(ligne7,moins);
382         if(nowm.mns > 32 && nowm.mns <= 37) {updateLine(ligne8,vingt);updateLine(ligne8
    ,tirret);updateLine(ligne8,mcinq);}
383         else if(nowm.mns > 37 && nowm.mns <= 42) {updateLine(ligne8,vingt);}
384         else if(nowm.mns > 42 && nowm.mns <= 47) {updateLine(ligne7,le); updateLine(
```

```
    ligne9,quart);}
385       else if(nowm.mns > 47 && nowm.mns <= 52) {updateLine(ligne7,mdix);}
386       else if(nowm.mns > 52 && nowm.mns <= 57) {updateLine(ligne8,mcinq);}
387     }
388   }
389   /*************************************************************************/
390   void PCF_UP_TIME(char dir)
391   {
392           GIE_bit = 0;
393           if(dir)// OK on touche pas pour le moment
394           {
395                   if(nowm.mns < (60-minutePlus)) nowm.mns += minutePlus;
396                   else nowm.mns = 0;
397                   buff = Dec2Bcd(nowm.mns);
398                   PCF_WR_CONFIG(3,buff);
399           }else
400           {
401           /*
402           Remplacer cette suite pas belle de IF/ELSE par un switch
403           > quid de la vitesse d'exécution... OSEF c'est 1 fois par minute ..?
404           */
405                   if(nowm.hrs == 11 && nowm.mnt == 1)
406                   {
407                           nowm.hrs = 12;
408                           nowm.mnt = 0;
409                   }else if(nowm.hrs == 11 && nowm.mnt == 0)
410                   {
411                           nowm.hrs = 12;
412                           nowm.mnt = 1;
413                   }
414                   else if(nowm.hrs == 12 && nowm.mnt == 1)
415                   {
416                           nowm.hrs = 1;
417                           nowm.mnt = 1;
418                   }
419                   else if(nowm.hrs == 12 && nowm.mnt == 0)
420                   {
421                           nowm.hrs = 1;
422                           nowm.mnt = 0;
423                   }
424                   else if(nowm.hrs < 11 && nowm.mnt == 1)
425                   {
426                           nowm.hrs++;
427                           nowm.mnt = 1;
428                   }
429                   else if(nowm.hrs <11 && nowm.mnt == 0)
430                   {
431                           nowm.hrs++;
432                           nowm.mnt = 0;
433                   }
434                   buff = Dec2Bcd(nowm.hrs);
435                   PCF_WR_CONFIG(4,(buff | 0b11000000));
436           }
437           GIE_bit = 1;
438   }
439   void sendLine(char line)
440   {
```

```
441         char p;
442         rDta = 0;
443         for(p=0;p<11;p++)
444         {
445                 if((matrice[line]>>p)&0x01){ rDta = 1;}
446                 else { rDta = 0;}
447                 rClk = 1; /*Delay_us(1);*/ rClk = 0; // impulsion sur CLK pour
    charger le registre avec rDta
448                 rDta = 0;
449         }
450 }
451
452 /*INTERRUPTIONS*/
453 void interrupt()
454 {
455 swInt;
456         if(INT2F_bit)
457         {
458         INT2F_bit = 0;
459                 INTi++;
460                 if(INTi >= majHourTempo)
461                 {
462                         flag = FLAG_RTC_RD;
463                         INTi=0;
464                 }else flag = FLAG_RTC_INTR;
465         }
466         if(RBIF_bit)
467         {
468                flag = FLAG_BOUTON;
469                RBIF_bit = 0;
470                j = PORTB;  // pas beau!
471
472         }
473         if(TMR0IF_bit)
474         {
475                   RD3_bit = ~RD3_bit;
476                   TMR0nbInt++;
477                   flag = FLAG_PRINT_LINE;
478
479                   TMR0IF_bit = 0;                    // clear TIMER0 int flag
480                   TMR0L  = 0x53;              // init timer0 count at 0x53
481         }
482 swInt;
483 }
```