DSPIC30F3011

iButton (ou bus "One Wire") : couche liaison

Principe retenu : utilisation intensive des interruptions

1. Résumé des spécifications de la couche physique

- Un seul signal permet des liaisons duplex et l'alimentation du "iButton"
- Type "collecteur ouvert" (ou "drain ouvert") :
  - résistance de pull-up de 4,7kΩ pour assurer l'alimentation du iButton
  - état "H" par mise en haute impédance du port
  - état "L" par mise en conduction du transistor "pull-down"
- Flancs actifs = ↓ (car plus rapides que les ↑)
- Les données échangées sont divisées en octets et transmises bit à bit.
- La transmission des données est divisée en "**time-slot**" d'une durée totale T_{SLOT} de 60µS à 120µS :
  - WRITE-0 : écriture d'un "0" par le master :
    - niveau "L" pdt 60µS à T_{SLOT}
    - niveau "H" le reste du T_{SLOT}
  - WRITE-1 : écriture d'un "1" par le master :
    - niveau "L" pdt 1µS à 15µS
    - niveau "H" le reste du T_{SLOT}
  - READ-DATA : lecture d'un état logique par le master :
    - le master commence par imposer un niveau "L" pdt 1µS à 15µS (le plus petit possible, pour maximaliser la fenêtre de lecture)
    - en réaction, le "iButton" place l'état du bit lu sur le bus pdt 15µS environ (typique)
  - La durée de repos entre 2 time-slot doit être supérieure à 1µS (pour charger le condensateur d'alim du iButton)
- "Reset" : chaque trame d'échange doit être précédée d'un reset :
  - La ligne est mise à "L" par le master pdt une durée de 480µS à 960µS
  - Le "iButton" répond 15µS à 60µS après l'impulsion du master par une impulsion à "L" de 60µS à 240µS : c'est l'**impulsion de présence**.

2. Principe

La transmission des bits est basée sur des impulsions de durées variables dont la tolérance est assez large. On veut absolument éviter les boucles d'attente logicielles pour obtenir ces durées :

- pour ne pas gâcher le temps CPU
- les autres interruptions peuvent perturber fortement les durées jusqu'à dépasser les tolérances

Ainsi, à l'exception du déclenchement de la transmission, tout le traitement est réalisé par une seule fonction d'interruption : "_T2Interrupt".
Cette fonction est déclenchée quand le compteur TMR2 du Timer 2 atteint la valeur du registre PR2. Ce registre est affecté dans cette fonction pour provoquer la prochaine interruption après un délai déterminé.

2.1.1 Codage des phases de la transmission

La fonction d'interruption "_T2Interrupt" est activée à la fin de chaque phase de chaque "time-slot".
Pour effectuer la suite du traitement, elle doit identifier la phase en cours dans la transmission d'une trame.
Les variables suivantes sont utilisées pour ce faire :

- `Cpt_octets_iButton` : compteur d'octets transmis ou reçus
- `Nb_Cmd_iButton` : nombre d'octets de la commande
- `Nb_Dta_iButton` : nombre d'octets de données (à écrire ou lire)
- `iButtonTimeSlot` : les valeurs de cette variable identifient les phases d'un "time-slot" :
  - `Reset_L` : état "L" d'un time-slot RESET en cours
  - `Reset_H` : état "H" d'un time-slot RESET en cours
  - `Write_0_L` : état "L" d'un time-slot WRITE-0 en cours
  - `Write_1_L` : état "L" d'un time-slot WRITE-1 en cours

- `Write_H_Cmd` : état "H" d'un time-slot WRITE-0 d'un octet de commande en cours
- `Write_H_Cmd` : état "H" d'un time-slot WRITE-0 d'un octet de "datas" en cours

2.1.2 Octets d'une trame "iButton"

Une trame "iButton" est constituée :

- d'une commande accompagnée de ses paramètres (4 octets maxi)
- d'un paquet de données à écrire ou à lire (16 octets maxi dans cette version)

Ces octets sont mémorisés dans les champs "Cmd_i[]" et "Datas_i[]" de la structure "Trame_iButton"

3. Algorithmes de la fonction d'interruption "_T2Interrupt"

Cette fonction d'interruption est exécutée au positionnement de l'indicateur T2IF du Timer 2. Ces demandes d'interruptions sont masquées, sauf pendant la transmission d'une trame "iButton".
Cette transmission est préparée dans la fonction "Read_iButton" qui réalise les opérations suivantes :

- Raz du compteur TMR2 du Timer 2
- Affectation de PR2 pour obtenir un délai de 720µS (quand TMR2 atteint PR2)
- Mise à "L" de la ligne "iButton"
- Affectation de la variable "iButtonTimeSlot" avec la valeur "Reset_L" : le premier time-slot est toujours un RESET
- Déblocage du compteur du Timer 2 et validation des interruptions T2IF

La fonction d'interruption "_T2Interrupt" est alors lancée pour la première fois 720µS plus tard.

Note : se reporter au §5 : "Relevés expérimentaux" qui illustre les descriptions.

3.1 Structure générale de _T2Interrupt

Après acquitement de l'interruption, la fonction effectue un choix multiple en fonction de la valeur de la variable "iButtonTimeSlot" pour réaliser le traitement correspondant.

`_T2Interrupt`					
`Acquitement de l'interruption`					
`Reset_L`	`Reset_H`	`Write_0_L`	`Write_1_L`	`Write_H_Cmd`	`Write_H_Dta`
1. Reset_L	2. Reset_H	3. Write_0_L	4. Write_1_L	5. Write_H_Cmd	6. Write_H_Dta

`iButtonTimeSlot`

3.2 Reset_L

Ce traitement est effectué à la 1° interruption après le lancement de la transmission d'une trame par la fonction "Read_iButton".

`1. Reset_L`
`Ligne iButton en HiZ (état "H")`
`iButtonTimeSlot <- Reset_H`
`iButton_OK <- 0 (faux)`
`Valider les interruptions IC1 au flanc descendant`

Si un "iButton" est présent, il réagit en produisant l'impulsion de présence pendant l'état "HiZ" : mise à "L" pendant 60µS à 240µS, 15µS à 60µS après la mise à "H".
Cette impulsion est détectée par la fonction d'interruption "_IC1Interrupt" (au flanc descendant) qui positionne l'indicateur "iButton_OK". Les autres interruptions IC1 sont alors inhibées (jusqu'à réception des datas de la trame).
Le contenu du registre PR2 n'est pas modifié : la prochaine interruption T2IF est provoquée encore 720µS plus tard.

3.3 Reset_H

Ce traitement est effectué à la 2° interruption, la précédente ayant effectuée le traitement "Reset_L". Entre-temps, le programme d'interruption "_IC1Interrupt" a positionné l'indicateur "iButton_OK".

2. Reset_H : phase "H" du time-slot RESET		
iButton_OK présent ?		
J		N
Ligne iButton à "L" (début d'un time-slot WRITE)		Bloquer le Timer 2
Raz compteur Timer 2 (TMR2)		
Cpt_octets_iButton <- 0 (compteur d'octets de la trame)		Inhiber les interruptions Timer 2
Cpt_bits_iButton <- 8 (compteur de bits d'un octet)		
Data_WR_iButton <- 1° octet de la commande		Inhiber les interruptions et les captures IC1
Bit indice 0 de Data_WR_iButton = 0 ?		
J	N	
iButtonTimeSlot <- Write_0_L (début d'un time-slot WRITE-0)	iButtonTimeSlot <- Write_1_L (début d'un time-slot WRITE-1)	
Prochaine interruption T2 ds 90µS (durée de la phase "L")	Prochaine interruption T2 ds 8 µS (durée de la phase "L")	

Le traitement consiste à préparer les variables utilisées pour séquencer la transmission de la trame et commence le premier time-slot WRITE. La prochaine interruption T2IF est programmée à une durée dépendant du bit de poids faible du premier octet de la commande.
Si le "iButton" est absent : le Timer 2 est bloqué et toutes les interruptions associées inhibées.

3.4 Write_0_L

Ce traitement est effectué à la fin de chaque phase "L" des time-slot WRITE-0, soit 90µS après la mise à "L" de la ligne "iButton".

3. Write_0_L : phase "L" d'un time-slot WRITE-0	
Ligne iButton à "H" (début de la phase "H" time-slot WRITE)	
Prochaine interruption ds 10µS (durée de la phase "H")	
Cpt_octets_iButton < Nb_Cmd_iButton	
J	N
iButtonTimeSlot <- Write_H_Cmd (phase "H" d'un bit commande)	iButtonTimeSlot <- Write_H_Dta (phase "H" d'un bit data)
Cpt_octets_iButton == 1 (dernier bit d'un octet transmis) ?	
J	N
Prochaine interruption retardée de 200µS (pour identifier la fin d'un octet à l'oscilloscope)	

La ligne est mise à "H" et la prochaine interruption est programmée 10µS plus tard, ce qui correspond à l'intervalle entre 2 time-slot (il doit être supérieur à 1µS).
Le programme détermine ensuite la nature du prochain time-slot (écriture d'un bit d'un octet de commande ou de data) et affecte la variable iButtonTimeSlot en conséquence ("Write_H_Cmd" ou "Write_H_Dta").
S'il s'agit du dernier time-slot d'un octet, la prochaine est retardée de 200µS pour repérer cette phase à l'oscilloscope (peut être éliminée dans la version finale).

3.5 Write_1_L

Ce traitement est effectué à la fin de chaque phase "L" des time-slot WRITE-1, soit 8µS après la mise à "L" de la ligne "iButton".

4. Write_1_L : phase "L" d'un time-slot WRITE-1

- Ligne iButton à "H" (début de la phase "H" time-slot WRITE)
- Prochaine interruption ds 92µS (durée de la phase "H")
- Cpt_octets_iButton < Nb_Cmd_iButton ?
  - J : iButtonTimeSlot <- Write_H_Cmd (phase "H" d'un bit commande)
  - N : iButtonTimeSlot <- Write_H_Dta (phase "H" d'un bit data)
- Cpt_octets_iButton == 1 (dernier bit d'un octet transmis) ?
  - J : Prochaine interruption retardée de 200µS (pour identifier la fin d'un octet à l'oscilloscope)
  - N :

La ligne est mise à "H" et la prochaine interruption est programmée 92µS plus tard (time-slot de 100µS). La suite du traitement est identique à "Write_0_L".

3.6 Write_H_Cmd

Ce traitement est effectué à la fin du time-slot précédent et commence le time-slot WRITE d'une commande (la ligne "iButton" est encore à "H").

5. Write_H_Cmd : phase "H" d'un time-slot WRITE-x d'une commande

- Décompter Cpt_bits_iButton (bit suivant)
- Cpt_bits_iButton==0 (dernier bit de Data_WR_iButton transmis) ?
  - J :
    - Cpt_bits_iButton <- 8 (on continue avec le bit LSB ...)
    - Incrémenter Cpt_octets_iButton (... de l'octet suivant)
    - Cpt_octets_iButton==Nb_Cmd_iButton (dernier octet de commande transmis) ?
      - J :
        - Data_WR_iButton <- 1° octet datas (pour le test du bit d'indice 0 ci-dessous)
        - Interruptions et captures IC1 aux flancs montants pour mesurer les durées à "L" des time-slot "READ"
      - N : Data_WR_iButton <- octet de commande suivant
  - N : Décaler Data_WR_iButton à droite (placer le bit suivant sur l'indice 0)
- Ligne iButton à "L" (début de la phase "L" d'un time-slot WRITE ou READ)
- Raz du compteur du Timer 2 (TMR2)
- Bit indice 0 de Data_WR_iButton = 0 ?
  - J :
    - iButtonTimeSlot <- Write_0_L (début d'un time-slot WRITE-0)
    - Prochaine interruption T2 ds 90µS (durée de la phase "L")
  - N :
    - iButtonTimeSlot <- Write_1_L (début d'un time-slot WRITE-1)
    - Prochaine interruption T2 ds 8µS (durée de la phase "L")

La variable "Data_WR_iButton" mémorise les 8 bits de la commande en cours de transmission. Les bits sont comptés par "Cpt_bits_iButton" pour tester le dernier bit et passer au prochain octet de commande. S'il s'agit du dernier bit du dernier octet de commande, on bascule sur le 1° octet "datas" et les captures IC1 aux flancs montants sont validées pour mesurer la durée des états "L".

La ligne "iButton" est alors mise à "L" et suivant l'état du bit à transmettre :

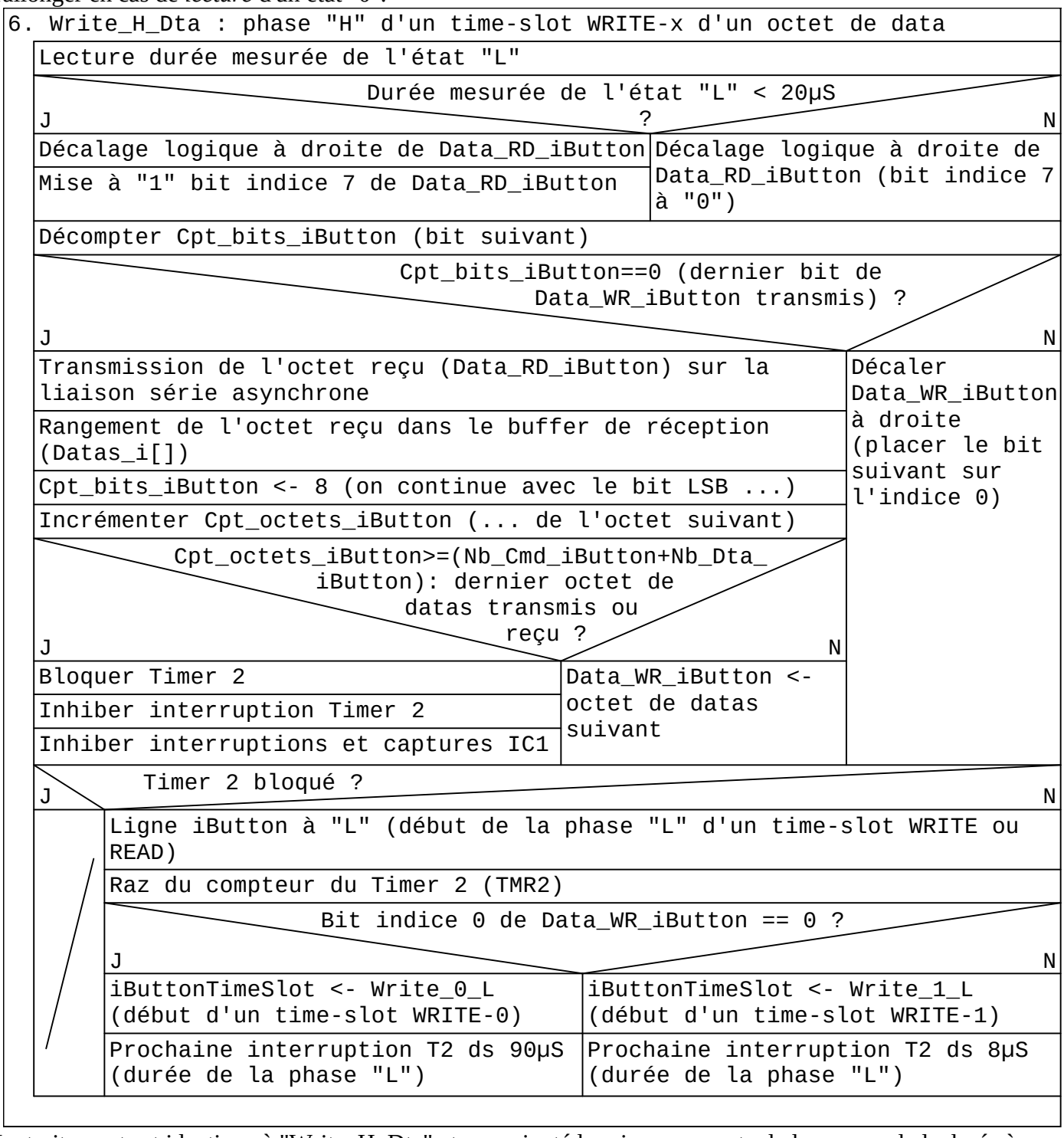
- la variable "iButtonTimeSlot" est affectée par "Write_1_L" ou "Write_0_L",
- le Timer 2 est programmé pour que la prochaine interruption soit déclenchée 8µS ou 90µS plus tard.

A la prochaine interruption, la fonction "_T2Interrupt" effectuera le traitement "Write_0_L" ou "Write_1_L" qui remet la ligne "iButton" à "H" (voir §3.4 & 3.5).

3.7 Write_H_Dta

Ce traitement est effectué à la fin du time-slot précédent et commence le time-slot WRITE d'une "data" (la ligne "iButton" est encore à "H").

Attention : la durée à l'état "L" du time-slot précédent dépend de la réaction du "iButton" qui peut la rallonger en cas de *lecture* d'un état "0".



Le traitement est identique à "Write_H_Dta" et on a ajouté la prise en compte de la mesure de la durée à l'état "L" du time-slot précédent.

De plus, le programme détecte la transmission du dernier bit de "datas" pour bloquer le Timer 2 et inhiber les 2 interruptions associées et terminer la transmission de la trame.

4. Sources

Ressources du dsPIC utilisées : "Timer 2" et "Input Capture 1"

4.1 Constantes et variables

```
                        /*****************************
                        *          Câblage           *
                        *****************************/
/* Ports "iButton"
******************
Sur port RD0=INT1 */
#define TRIS_iButton TRISDbits.TRISD0 // Bit de contrôle 3 états
#define LAT_iButton  LATDbits.LATD0   // Bit du "latch"
#define iButton_1    TRISDbits.TRISD0=1 // Mise à "1" par pull-up
#define iButton_0    TRISDbits.TRISD0=0 // Mise à "0"

/* Port série asynchrone sur UART2
*********************************
RXD sur port RF4=U2RX (broche 28)
TXD sur port RF5=U2TX (broche 27) */

/* Signaux de test
******************
TEST0 sur port RF0 (broche 30)
TEST1 sur port RF1 (broche 29) */
#define TEST0 LATFbits.LATF0
#define TEST1 LATFbits.LATF1
#define TEST0_Dir TRISFbits.TRISF0
#define TEST1_Dir TRISFbits.TRISF1

                        /*****************************
                        *    Définitions de type     *
                        *****************************/

typedef enum
{
  Write_0_L   = 0, // Time-slot WRITE-0, ligne iButton à "L"
  Write_1_L   = 1, // Time-slot WRITE-1, ligne iButton à "L"
  Write_H_Cmd = 2, // Time-slot WRITE-0 ou WRITE-1 pour Cmd, ligne iButton à "H"
  Write_H_Dta = 3, // Time-slot WRITE-0 ou WRITE-1 pour Data, ligne iButton à "H"
  Reset_L     = 4, // Time-slot RESET, ligne iButton à "L"
  Reset_H     = 5, // Time-slot RESET, ligne iButton à "L"
  Read_i      = 6
}tiButtonTimeSlot; // Type de time slot iButton

typedef enum
{
  Write   = 0,
  Read    = 1
}tR_W; // Type "R/W" pour accès iButton

                        /*****************************
                        * Constantes non mémorisées  *
                        *****************************/
#define FCY      4000000*16/4    // xtal = 4Mhz; PLLx16 -> 16 MIPS

                        /*******************************
                        *  Constantes en mémoire flash *
                        *******************************/

                        /*****************************
                        *       Variables            *
                        *****************************/
/* iButton
**********/
tiButtonTimeSlot iButtonTimeSlot;// Type de séquence iButton en cours
```

```
unsigned char Data_WR_iButton; // Octet en cours d'écriture
unsigned char Data_RD_iButton; // Octet en cours de réception
unsigned char Cpt_bits_iButton;// Compteur de bits en cours
unsigned int Cpt_octets_iButton; // Compteur d'octets transmis sur "iButton"
char iButton_OK;      // Indique la présence d'un iButton
char CLEF_OK;         // Indique la présence de la clef codée
struct tTrame_iButton
 {
  char Cmd_i[4];      // Jusqu'à 4 octets de commande
  char CRC;           // CRC de la commande
  tR_W R_W_i;         // Nature de la trame (lecture ou écriture)
  char Datas_i[16];   // Jusqu'à 16 octets de données (lecture ou écriture)
  char Nb_Dta_iButton;// Nombre d'octets de Data à transmettre ou recevoir
  char Nb_Cmd_iButton;// Nombre d'octets de la commande
 }Trame_iButton;
```

4.2 Initialisations

```
/*****************************************************************
  Function:        void InitTimer2(void)
  Description:     Initialisation du Timer 2 (Timer B)
                   Il est utilisé pour produire les impulsions "iButton"
*****************************************************************/
void InitTimer2(void)
{
 T2CON=0x0000; // Fonctionnement continu, horloge=TCY=16MHz,
               // compteur bloqué
 TMR2=0;       // Raz du compteur pour décaler la 1° interruption
 PR2=0xFFFF;   // Pour retarder l'int. au max
}

/*****************************************************************
  Function:        void InitIC1(void)
  Description:     Initialisation "Input Capture 1"
                    - interruption aux flancs montant de IC1 pour la
                      détection de l'impulsion de présence "iButton"
*****************************************************************/
void InitIC1(void)
{
 IC1CONbits.ICM=0;   // Aucune capture IC1
 IC1CONbits.ICTMR=1; // Échantillonnage de TMR2
}

/*****************************************************************
  Function:        void Init_iButton(void)
  Description:     Initialisation du port "iButton" :
                    - Latch affecté avec "0"
                    - Chgt d'état par le contrôle trois états
                    - État initial : "1" par pull-up
*****************************************************************/
void Init_iButton(void)
{
 TRIS_iButton=1; // Port en entrée => état "1" par pull-up
 LAT_iButton=0;  // État "0"
}
```

4.3 Fonctions classiques

```
/*****************************************************************
  Function:    void Read_iButton(void)
  Description: lecture de "Nb_octets_iButton" depuis l'iButton et
               affectation de Trame_iButton.Datas_i[]
  Pré-requis : Cmd_i[], Nb_Cmd_iButton et Nb_octets_iButton de la structure
               Trame_iButton doivent être affectés
  Traitement : - Mise à "L" de la ligne "iButton" (début d'un time-slot
                 "RESET")
               - Préparation du Timer 2 pour produire une interruption
                 720µS plus tard
               - Le programme d'interruption _T2Interrupt s'occupe de la fin
                 time-slot RESET, de la détection de l'impulsion de présence,
                 et du reste du traitement
*****************************************************************/
void Read_iButton(void)
{
 TMR2=0;           // Raz du compteur
 PR2=11519;        // L'état "0" dure 720µS
 iButton_0;        // Ligne "iButton" à "0"
 iButtonTimeSlot=Reset_L; // On commence par une impulsion "Reset" sur iButton
 T2CONbits.TON=1;  // Validation timer 1
 IEC0bits.T2IE=1;  // Validation interruption Timer 2
}
```

4.4 Fonctions d'interruptions

```
/*****************************************************************
  Function:     void _ISR _T2Interrupt (void)
  Description:  Transmission d'une trame "iButton" suivant le contenu
                de la structure "Trame_iButton" et lecture des données
                   Traitement :
                    - Fin du time-slot RESET déclenché par la fonction
                      "Read_iButton"
                    - Transmission des octets de commande du champ Cmd_i[]
                    - Transmission/lecture des "datas" depuis/dans le
                      champ Datas_i[]
*****************************************************************/
void _ISR _T2Interrupt(void)
{
 unsigned int i;
 IFS0bits.T2IF=0; //Acquitement interruption "Timer 2"
 switch (iButtonTimeSlot)
 {
  case Reset_L : // Fin 1° phase time slot "Reset" ?
  {
   iButton_1; // Affecter la ligne "iButton" à "1" côté Master
   iButtonTimeSlot=Reset_H; // Début 2° phase du time-slot "Reset"
   iButton_OK=0;     // Pas d'impulsion présence iButton détectée
   IC1CONbits.ICM=2; // Interruptions aux flancs descendants
   IFS0bits.IC1IF=0; // Raz indicateur interruption IC1 = ligne iButton
   IEC0bits.IC1IE=1; // Validation interruption IC1
   break;
  }
  case Reset_H : // Fin 2° phase time slot "Reset" ?
  {
   if (iButton_OK) // L'impulsion de présence a-t-elle eu lieu ?
   {
    iButton_0;     // Un time-slot "write" commence tjrs par "0"
    TMR2=0;        // Raz du compteur du timer 2
    Cpt_octets_iButton=0; // Compteur d'octets
    Cpt_bits_iButton=8;   // Décompteur de bits
    // Préparation 1° octet à transmettre
    Data_WR_iButton=Trame_iButton.Cmd_i[0];
    // Test du bit à transmettre
    if ((Data_WR_iButton & 0x01)==0)
     {// Bit à "0" :
      iButtonTimeSlot=Write_0_L; // Début phase "L" du time-slot WRITE-0
      PR2=1439;                  // Impulsion à l'état "0" de 90µS
     }
    else
     {// Bit à "1" :
      iButtonTimeSlot=Write_1_L; // Début phase "L" du time-slot WRITE-1
      PR2=127;  // Impulsion à l'état "0" de 8µS
     }
   }
   else
   {
    T2CONbits.TON=0;  // Blocage timer 2
    IEC0bits.T2IE=0;  // Inhibition interruption Timer 2
    IFS0bits.IC1IF=0; // Raz indicateur interruption IC1
    IEC0bits.IC1IE=0; // Inhiber les prochaines interruptions IC1
    IC1CONbits.ICM=0; // Aucune capture IC1
   }
   break;
  }
  case Write_0_L : // Fin de l'impulsion à l'état "0" (WRITE-0) ?
  {
   iButton_1; // Oui : début de la phase "H" du time-slot "WRITE-0"
   // Pas de RAZ de TMR2 car mise à "1" lente
   PR2=159;   // Prochain bit ds 10µS
   if (Cpt_octets_iButton < Trame_iButton.Nb_Cmd_iButton)
```

```
      iButtonTimeSlot=Write_H_Cmd; // Phase "H" du time-slot WRITE-0 "Commande"
  else
      iButtonTimeSlot=Write_H_Dta; // Phase "H" du time-slot WRITE-0 "Datas"
  if (Cpt_bits_iButton==1) PR2=PR2+3200; // Pour repérer les octets à l'oscillo
  break;
 }
 case Write_1_L : // Fin de l'impulsion à l'état "0" (WRITE-1) ?
 {
  iButton_1; // Oui : début de la phase "H" du time-slot "WRITE-1"
  // Pas de RAZ de TMR2 car mise à "1" lente
  PR2=1471;  // Prochain bit ds 92µS
  if (Cpt_octets_iButton < Trame_iButton.Nb_Cmd_iButton)
      iButtonTimeSlot=Write_H_Cmd; // Phase "H" du time-slot WRITE-1 "Commande"
  else
      iButtonTimeSlot=Write_H_Dta; //  Phase "H" du time-slot WRITE-1 "Datas"
  if (Cpt_bits_iButton==1) PR2=PR2+3200; // Pour repérer les octets à l'oscillo
  break;
 }
 case Write_H_Cmd : // Fin phase "H" du time-slot WRITE-0 ou WRITE-1 d'une commande ?
 {
  Cpt_bits_iButton--;    // Décomptage des bits
  if (!Cpt_bits_iButton) // Dernier bit de Data_WR_iButton transmis ?
   {
    Cpt_bits_iButton=8;  // Oui : on recommence avec le bit LSB
    Cpt_octets_iButton++;//       de l'octet suivant
    if (Cpt_octets_iButton==Trame_iButton.Nb_Cmd_iButton)
    {// Si dernier bit du dernier octet "Cmd" transmis :
     Data_WR_iButton=Trame_iButton.Datas_i[0]; // 1° octet "datas" à transmettre
     IC1CONbits.ICM=3; // Interruptions et captures aux flancs montants
                       // pour la mesure des durées à l'état bas avec IC1
     IEC0bits.IC1IE=1; // Valider les prochaines interruptions IC1
    }
    // Sinon, on cherche l'octet de commande suivant
    else Data_WR_iButton=Trame_iButton.Cmd_i[Cpt_octets_iButton];
   }
  else Data_WR_iButton >>=1; // Pas dernier bit de Data_WR_iButton : bit suivant
  // Test du bit à transmettre
  iButton_0;  // Un time-slot "WRITE" ou "READ" commence tjrs par "0"
  TMR2=0;     // Raz du compteur du timer 2
  if ((Data_WR_iButton & 0x01)==0)
   {// Bit à "0" :
    iButtonTimeSlot=Write_0_L; // Début phase "L" du time-slot WRITE-0
    PR2=1439;  // Impulsion à l'état "0" de 90µS
   }
  else
   {// Bit à "1" :
    iButtonTimeSlot=Write_1_L; // Début phase "L" du time-slot WRITE-1
    PR2=127;  // Impulsion à l'état "0" de 8µS
   }
  break;
 }
 case Write_H_Dta : // Fin phase "H" du time-slot WRITE-0 ou WRITE-1 d'une data ?
 {
  // Lecture du dernier bit sur la ligne iButton
  i=IC1BUF;    // Lecture de la mesure de la durée à "L"
  // Accumulation bit lu
  Data_RD_iButton >>=1;               // État "0" à priori
  if (i < 320) Data_RD_iButton+=0x80; // < 20µS : état "1"
  //WriteIntTXD(i); // **** TEST **** : transmission de la durée mesurée
  Cpt_bits_iButton--;    // Décomptage des bits
  if (!Cpt_bits_iButton) // Dernier bit de Data_RD_iButton transmis ?
   {
    WriteTXD(Data_RD_iButton);// Oui : Transmission de l'octet lu
    // Affectation du buffer de "datas"
    Trame_iButton.Datas_i[Cpt_octets_iButton-
         Trame_iButton.Nb_Cmd_iButton]=Data_RD_iButton;
    Cpt_bits_iButton=8;  // On recommence avec le bit LSB
```

```
     Cpt_octets_iButton++;// de l'octet suivant
     if (Cpt_octets_iButton>=(Trame_iButton.Nb_Cmd_iButton+Trame_iButton.Nb_Dta_iButton))
     {// Si dernier bit du dernier octet "Datas" transmis :
      T2CONbits.TON=0;  // Blocage Timer 2
      IEC0bits.T2IE=0;  // Inhibition interruption Timer 2
      IFS0bits.IC1IF=0; // Raz indicateur interruption IC1
      IEC0bits.IC1IE=0; // Inhiber les prochaines interruptions IC1
      IC1CONbits.ICM=0; // Aucune capture IC1
      break;
     }
     // Sinon, on cherche l'octet de dta suivant
     else Data_WR_iButton=Trame_iButton.Datas_i[Cpt_octets_iButton-Trame_iButton.Nb_Cmd_iButton];
    }
   else Data_WR_iButton >>=1; // Pas dernier bit de Data_WR_iButton : bit suivant
   // Test du bit à transmettre
   iButton_0;  // Un time-slot "WRITE" ou "READ" commence tjrs par "0"
   TMR2=0;     // Raz du compteur du timer 2
   if ((Data_WR_iButton & 0x01)==0)
    {// Bit à "0" :
     iButtonTimeSlot=Write_0_L; // Début phase "L" du time-slot WRITE-0
     PR2=1439;  // Impulsion à l'état "0" de 90µS
    }
   else
    {// Bit à "1" :
     iButtonTimeSlot=Write_1_L; // Début phase "L" du time-slot WRITE-1
     PR2=127;  // Impulsion à l'état "0" de 8µS
    }
   break;
  }
 }
}
```

```
/*****************************************************************
  Function:     void _ISR _IC1Interrupt (void)
  Description : La ligne "iButton" est connectée sur IC1
                L'interruption IC1 est validée au flanc descendant pendant
                la 2° phase du time slot "Reset" pour détecter l'impulsion
                de présence "iButton" : affectation de iButton_OK.
                Puis à chaque flanc montant pdt la réception des datas pour
                échantilloner TMR2 et mesurer la durée des impulsions
*****************************************************************/
void _ISR _IC1Interrupt (void)
{
 int i;
 TEST1=1;
 IFS0bits.IC1IF=0; // Raz indicateur interruption
 if (iButtonTimeSlot==Reset_H) // 2° phase time-slot RESET en cours ?
 {
  do i=IC1BUF;      // Pour vider la mémoire FIFO
  while (IC1CONbits.ICBNE);
  IEC0bits.IC1IE=0; // Inhiber les prochaines interruptions IC1 jusqu'aux
                    // bits "datas"
  iButton_OK=1;     // Impulsion présence iButton détectée
  IC1CONbits.ICM=0; // Aucune capture jusqu'aux bits "datas"
 }
 TEST1=0;
}
```

4.5 Fonctions de test

4.5.1 Timer 1

Il est utilisé pour déclencher les trames de lecture au rythme de 10 par seconde

```
/*****************************************************************
  Function:        void InitTimer1(void)
  Description:     Initialisation du Timer 1 (Timer A) pour provoquer
                   une interruption à rythme régulier.
  Note : le rythme est choisi pour obtenir une fréquence des transmissions
         des paramètres sur I2C de 10Hz
*****************************************************************/
void InitTimer1(void)
{
 T1CON=0x0030;         // Fonctionnement continu, horloge=Fcy/256=2MHz
 TMR1=0;             // Raz du compteur pour décaler la 1° interruption
 PR1=(FCY/256)/10-1; // Fréquence = 10Hz
 T1CONbits.TON=1;    // Validation timer 1
}
```

```
/*****************************************************************
  Function:        void _ISR _T1Interrupt (void)
  Description:     Fréquence = 10Hz
                   Traitement :
                    - Préparation de la structure Trame_iButton
                    - Lecture du "iButton"
*****************************************************************/
void _ISR _T1Interrupt (void)
{
 char i;
 IFS0bits.T1IF=0; // Acquitement interruption
 TEST0=!TEST0;    // Inversion TEST0 pour tests
 Trame_iButton.Cmd_i[0]=0x33;
 Trame_iButton.Cmd_i[1]=0;
 Trame_iButton.Cmd_i[2]=0;
 Trame_iButton.Cmd_i[3]=0;
 for (i=5; i<5+8; i++) Trame_iButton.Datas_i[i-5]=0xFF;
 Trame_iButton.Nb_Dta_iButton=8;
 Trame_iButton.Nb_Cmd_iButton=1;
 Read_iButton();
}
```

4.5.2 Main

```
int main (void)
{
 U2MODE=0;    //Inhiber l'UART à cause du bootloader
 TEST0_Dir=0; // Signal TEST0 en sortie
 TEST1_Dir=0; // Signal TEST1 en sortie
 Init_iButton();// Port en entrée => état "1" par pull-up
 InitUART();  // 38400 bauds, 8 bits, pas de parité, 1 bit stop
 InitTimer1();// Initialisation Timer 1 pour RTI de 10Hz
 InitTimer2();// Initialisation Timer 2 pour "iButton"
 InitIC1();   // Initialisation IC1 pour détection impulsion présence iButton
 // Validations des interruptions
 IFS0bits.T1IF=0;   // Raz indicateur Timer 1
 IEC0bits.T1IE=1;   // Validation interruption Timer 1
 IFS1bits.U2TXIF=0; // Raz indicateur interruption TX sur UART2
 IEC1bits.U2TXIE=1; // Validation interruption TX sur UART2
 //IFS1bits.U2RXIF=0; // Raz indicateur interruption RX sur UART2
 //IEC1bits.U2RXIE=1; // Validation interruption RX sur UART2
 while (1) {};
}
```