//-----------------------------------------------------------------------------

// TP4\_BSE\_Main.c

// AUTH: FJ, LAJUGIE Rodolphe, CORNATON Maxime

// DATE: 5/12/21

//

// Target: C8051F02x

// Tool chain: KEIL Microvision5

//

//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------

// Cette application exécute les tâches suivantes:

// SYSCLK = Quartz externe = 22.1184 MHz (Visu sur P1.0)

// Clignotement de la LED P1.6

// Commande du clignotement de la LED avec le bouton poussoir P3.7 (INT7) et

// via le signal DECL\_EXT sur P3.6 (INT6)

// Mise en oeuvre de la base de temps Timer2 de 10ms et interruption Timer2

// Mise en oeuvre d'un comptage d'évènements SIG\_IN (sur T4)

// Le comptage d'évènement est assuré par le Timer4 et par une interruption Timer4

// Comptage Programmable selon les facteurs 10-100-1000-10000

// La programmation du comptage est assurée au travers des niveaux sur P2.5 et P2.6

// Le signal SIG\_OUT change d'état à chaque fois que l'on a compté 10...10000 évènements

// Calcul de la fréquence de SIG\_IN et stockage dans la variable Frequence

// La mesure de fréquence est assurée par une mesure du nombre d'évènements sur une seconde

// Mesure réalisée dans l'interruption Timer2

// Visu Flag INT7 sur P2.4

// Visu Flag INT6 sur P6.4

// Visu Flag INT Timer2 sur P3.5

// Visu Flag INT Timer4 sur P3.4

// Visu SIG\_OUT sur P3.3

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

// Fichiers d'entête

#include "intrins.h"

#include<c8051F020.h>

#include<c8051F020\_SFR16.h>

#include<TP4\_BSE\_Lib\_Config\_Globale.h>

#include<TP4\_BSE\_Lib\_Divers.h>

#include<TP4\_BSE\_Main.h>

void Cfg\_CLK\_UART0(void);

void Cfg\_UART0(void);

//-----------------------------------------------------------------------------

// Déclaration des MACROS

#define SYSCLK 22118400L

#define LED\_ON 1

#define LED\_OFF 0

#define LED\_BLINK 0

#define BP\_ON 0

#define BP\_OFF 1

#define TO\_BE\_PROCESSED 1

#define PROCESSED 0

#define SET\_VISU\_INT6 P6 |= (1<<4)

#define RESET\_VISU\_INT6 P6 &= ~(1<<4)

//-----------------------------------------------------------------------------

// Déclarations Registres et Bits de l'espace SFR

sbit LED = P1^6; // LED

sbit BP =P3^7;

sbit VISU\_INT7 = P2^4;

sbit VISU\_INT\_TIMER2 = P3^5;

sbit VISU\_INT\_TIMER4 = P3^4;

sbit SIG\_OUT = P3^3;

//-----------------------------------------------------------------------------

// Variables globales

bit Event = PROCESSED; // Détection des évènements pour changer le clignotement de la LED

unsigned int Event\_to\_Count = 100 ; //Valeur du comptage d'évènements

long frequence = 0; // Fréquence calculée de SIG\_IN

unsigned int CP\_Overflow\_Timer4; // Compteur d'overflows du Timer4

//-----------------------------------------------------------------------------

// MAIN Routine

//-----------------------------------------------------------------------------

void main (void) {

// Configurations globales

Init\_Device();

// Configurations spécifiques

Config\_INT7(); // Configuration de INT7

Config\_INT6(); // Configuration de INT6

Config\_Timer2\_TimeBase();

Config\_Timer4\_Event\_Counter();

Cfg\_CLK\_UART0();

Cfg\_UART0();

// Fin des configurations

EA = 1; // Validation globale des interruptions

// Boucle infinie

while(1){

SBUF0 = 0x40;

Software\_Delay\_10ms();

}

}

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

void Config\_INT7(void)

{

P3IF &= ~(1<<7); // IE7 mis à 0 pending flag de INT7 effacé

P3IF &= ~(1<<3); // IE7CF mis à 0 - sensibilité int7 front descendant

EIP2 &= ~(1<<5); // PX7 mis à 0 - INT7 priorité basse

EIE2 |= (1<<5); // EX7 mis à 1 - INT7 autorisée

}

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

void ISR\_INT7 (void) interrupt 19

{

VISU\_INT7 = 1;

P3IF &= ~(1<<7); // IE3 mis à 0 - remise à zéro du pending flag de INT7 effacé

Event = TO\_BE\_PROCESSED;

VISU\_INT7 = 0;

}

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

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

void Config\_INT6(void)

{

P3IF &= ~(1<<7); // IE6 mis à 0 pending flag de INT6 effacé

P3IF &= ~(1<<2); // IE6CF mis à 0 - sensibilité int6 front descendant

EIP2 &= ~(1<<4); // PX6 mis à 0 - INT7 priorité basse

EIE2 |= (1<<4); // EX6 mis à 1 - INT7 autorisée

}

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

void ISR\_INT6 (void) interrupt 18

{

SET\_VISU\_INT6;

P3IF &= ~(1<<6); // IE6 mis à 0 - remise à zéro du pending flag de INT6 effacé

P3IF ^= (1<<2); // Action sur IE6CF - Commutation Front montant / Front Descendant

Event = TO\_BE\_PROCESSED;

RESET\_VISU\_INT6;

}

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

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

void Config\_Timer2\_TimeBase(void)

{

CKCON &= ~(1<<5); // T2M: Timer 2 Clock Select

// CLK = sysclk/12TR2 = 0; //Stop Timer

TF2 = 0; // RAZ TF2

EXF2 = 0; // RAZ EXF2

RCLK0 = 0;

TCLK0 = 0;

CPRL2 = 0; // Mode AutoReload

EXEN2 = 0; // Timer2 external Enable Disabled

CT2 = 0; // Mode Timer

RCAP2 = -((SYSCLK/12)/100);

T2 = RCAP2;

TR2 = 1; // Timer2 démarré

PT2 = 1; // Priorité Timer2 Haute

ET2 = 1; // INT Timer2 autorisée

}

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

void ISR\_Timer2 (void) interrupt 5

{

static char CP\_Cligno;

static char CP\_Seconde = 0;

static bit STATE\_LED = LED\_BLINK;

char Read\_Cfg\_Event;

static unsigned int CP\_Timer4 = 0;

static unsigned int OLD\_CP\_Timer4 = 0;

VISU\_INT\_TIMER2 = 1;

if (TF2 == 1)

{

TF2 = 0;

// Gestion de la mesure de frequence - Synthèse

// Le calcul de la fréquence est fait toutes les secondes

CP\_Seconde++;

if (CP\_Seconde >= 100)

{

CP\_Seconde = 0;

T4CON &= ~(1<<2); // TR4 = 0 -- Timer4 stoppé

CP\_Timer4 = T4; // Lecture Compteur Timer4

T4CON |= (1<<2); // TR4 = 1 -- Timer4 redémarré

// Fréquence = nbre total d'évènements par seconde

frequence = ((CP\_Overflow\_Timer4) \* (long)Event\_to\_Count) +

(int)(CP\_Timer4 - OLD\_CP\_Timer4);

OLD\_CP\_Timer4 = CP\_Timer4;

CP\_Overflow\_Timer4 = 0;

}

// Gestion des évènements INT6 et INT7

// pour gérer les modes de clignotement de la LED

if (Event == TO\_BE\_PROCESSED)

{

Event = PROCESSED;

STATE\_LED = !STATE\_LED;

}

// Gestion du mode de clignotement de la LED

CP\_Cligno++;

if (CP\_Cligno > 11) CP\_Cligno = 0;

if (STATE\_LED == LED\_BLINK)

{

if (CP\_Cligno < 2) LED = LED\_ON;

else LED = LED\_OFF;

}

else LED = LED\_OFF;

// Etape5 - Lecture consigne du nbre d'évènements à compter pour générer SIG\_OUT

Read\_Cfg\_Event = (P2 & ((1<<5)|(1<<6)))>>5; // Lecture consigne sur P2.5 et P2.6

switch (Read\_Cfg\_Event)

{

case 0:

Event\_to\_Count = 10;

break;

case 1:

Event\_to\_Count = 100;

break;

case 2:

Event\_to\_Count = 1000;

break;

case 3:

Event\_to\_Count = 10000;

break;

default:

Event\_to\_Count = 100;

break;

}

}

// Sécurité: si EXF2 est à 1 - RAZ de EXF2

if (EXF2 == 1)

{

EXF2 = 0;

}

VISU\_INT\_TIMER2 = 0;

}

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

void Config\_Timer4\_Event\_Counter(void) // Pour étape 4

{

// Timer 4 configuré en compteur d'évènements SIG\_IN

// SIG\_IN câblé sur l'entrée T4 du Timer4

// Timer4 en mode autorechargement

// Mode Counter

// CLKTimer = SIG\_IN

// Valeur rechargement= (65536 - Nbre\_Event\_to\_count)

// L'overflow, et donc l'interruption Timer4 signale le comptage de 100 évènements

T4CON = 0x02; // Flags TF4 et EXF4 effacés

// RCLK1 et TCLK1 à zéro

// Transitions sur T4EX ignorées

// Timer stoppé

// Timer en mode counter

// Mode auto-reload

RCAP4 = 65536-Event\_to\_Count;

T4 = RCAP4;

T4CON |= (1<<2); // TR4 = 1 -- Timer2 démarré

EIP2 |= (1<<2); //PT4 = 0 -- Priorité Timer4 haute

EIE2 |= (1<<2); //ET4 = 1 -- INT Timer4 autorisée

}

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

void ISR\_Timer4 (void) interrupt 16 // Etape4

{

VISU\_INT\_TIMER4 = 1;

if ((T4CON & (1<<7)) != 0) // Test TF4 = 1

{

T4CON &= ~(1<<7); // RAZ TF4

SIG\_OUT = !SIG\_OUT; // Génération SIG\_OUT

// Prise en compte d'un comptage d'évènements configurable

// La lecture de la config 10-100-1000-10000 est faite dans ISR\_Timer2

RCAP4 = 65536 - Event\_to\_Count; //Reload compteur d'évènements

// Comptage des overflows (pour compter au delà de 65536)

CP\_Overflow\_Timer4++;

}

// Sécurité si T4EX est à 1

if ((T4CON & (1<<6)) != 0) // Au cas où: Test EXF4 = 1

{ T4CON &= ~(1<<6); } // RAZ EXF4

VISU\_INT\_TIMER4 = 0;

}

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

void Cfg\_CLK\_UART0(void)

{

TCON &= ~(1<<6); //Timer1 Stoppé (précaution)

TCON &= ~(1<<7);//Flag Timer1 effacé

CKCON |= (1<<4); //Config CKCON - | T1M: Timer1 ClockSelect – CLK Timer = Sysclk

TMOD |=(1<<5);//Config TMOD - Timer1 configuré en timer 8 bit avec auto-reload

TMOD &= ~(1<<4);//Config TMOD - Timer1 configuré en timer 8 bit avec auto-reload

TH1 = 0xBB;//Programmation du registre de rechargement TH1

//Initialisation du registre Timer TL1 – Nécessaire? /Non car ar defaut à 0.

IE &= ~(1<<3); //Dévalidation de l’interruption Timer1 – Action sur registre IE

TCON |= (1<<6); //Timer1 démarré - – Action sur TR1 de TCON

}

void Cfg\_UART0(void)

{

SCON0 |= (1<<4);

SCON0 &= ~(1<<0);

SCON0 &= ~(1<<1);

SCON0 |= (1<<5);

SCON0 &= ~(1<<7);

SCON0 |= (1<<6);

T2CON &= ~(1<<4);

T2CON &= ~(1<<5);

PCON |= (1<<7);

PCON &= ~(1<<6);

}