```
//****************************************************************************
// Devoir BSE - Session 1 - Janvier 2021
//****************************************************************************
//Préprocesseur
#include "C8051F020.h"
#define YES 1
#define NO 0
//****************************************************************************
// Prototypes de fonction
long int Get_EVENT();
void Send_string(char *msg);
//****************************************************************************
// Déclaration SFR16
sfr16 TMR3RL   = 0x92;                 // Timer3 reload value
sfr16 TMR3     = 0x94;                 // Timer3 counter
sfr16 RCAP2    = 0xca;                 // Timer2 capture/reload
sfr16 T2       = 0xcc;                 // Timer2
sfr16 RCAP4    = 0xe4;                 // Timer4 capture/reload
sfr16 T4       = 0xf4;                 // Timer4
//****************************************************************************
// GPIO
sbit Flag_IntA = P3^0;
sbit Flag_IntB = P3^1;
sbit Flag_IntC = P3^2;
sbit BP1 = P2^0; sbit BP2 = P2^1; sbit BP3 = P2^2; sbit BP4 = P2^3;
sbit LED1 = P2^4; sbit LED2 = P2^5; sbit LED3 = P2^6; sbit LED4 = P2^7;
//****************************************************************************
// Variables globales
unsigned char Info_Heures, Info_Minutes, Info_Secondes, Info_Centiemes;
long int CP_EVENT = 0;
long int Mesure_TH;
const char msg_test[] = "Message de Test\n";
//****************************************************************************
void Reset_Sources_Init()
{
    WDTCN     = 0xDE;
    WDTCN     = 0xAD;
}
//****************************************************************************
void Oscillator_Init()   // Configuration SYSCLK = Quartz externe = 22,1184 MHz
{
     int i = 0;
    OSCXCN    = 0x67;
    for (i = 0; i < 3000; i++);
    while ((OSCXCN & 0x80) == 0);
    OSCICN    = 0x0C;
}
//****************************************************************************
void Port_IO_Init()
{
    XBR0      = 0x04;
    XBR1      = 0x80;
    XBR2      = 0x5C;
    P0MDOUT  |= 0x05;
    P2MDOUT   = 0xF0;
    P2       |= 0x0F;
}
//****************************************************************************
```

```c
void Init_Device(void)
{
   Reset_Sources_Init();
   Oscillator_Init();
   Port_IO_Init();
}
//****************************************************************************
void Timer3_Init()
{
   TMR3RLH   = 0xB8;
   TMR3RLL   = 0x00;
   TMR3CN    = 0x04;
   EIE2     |= 0x01;
   EIP2     &= ~0x01;
}
//*****************************************************************************
void Timer2_Init() // Timer compteur d'évènements
{
   T2CON = 0X21;
   RCAP2 = 0;
   ET2   = 1;
   PT2   = 1;
   TR2   = 1;
}
//****************************************************************************
void Timer1_Init()
{
   CKCON    |= 0x10;
   PCON     &= ~0x80;
   TMOD      = 0x20;
   TH1       = 0xB8;
   TCON     |= 0x40;
}
//****************************************************************************
void Timer4_Init()
{
   CKCON |= 0x40;
   T4CON = 0x09;
   RCAP4 = 0;

   EIE2  |=0x04;
   PT2   &= ~0x04;
   T4CON |= 0x04;
}
//****************************************************************************
void CFG_uart()
{
      PCON  &= ~0x80;   //SMOD0: UART0 Baud Rate Divide by two Enabled.
      T2CON &=~0x30;
      PCON &= 0xBF;    // SSTAT0=0
      SCON0 = 0x72;   // Mode 1 - Check Stop bit - Reception validée
                  // Transmission: octet transmis (prêt à recevoir un char
                      // pour transmettre
}



//****************************************************************************
```

```
//****************************************************************************
void ISR_Horodatage (void) interrupt 14
{
   Flag_IntA = 1;     // Flag Interruption mis à 1
   TMR3CN &= ~0x80;
   Info_Centiemes++;
   if (Info_Centiemes >= 100)
   {
      Info_Centiemes = 0;
      Info_Secondes++;
      if (Info_Secondes >= 60)
      {
         Info_Secondes =0;
         Info_Minutes++;
         if (Info_Minutes >= 60)
         {
            Info_Minutes = 0;
            Info_Heures++;
            if (Info_Heures >= 24) Info_Heures = 0;
         }
      }
   }
   Flag_IntA = 0; // Flag Interruption mis à 0
}
//****************************************************************************
//****************************************************************************
void ISR_CP_EVENT (void) interrupt 5
{
   Flag_IntB = 1;
  if (TF2 ==1)
   {   TF2 = 0;
      CP_EVENT += 65536;
   }
    if (EXF2 ==1)
   {   EXF2 = 0;
   }
   Flag_IntB = 0;
}
//****************************************************************************
//****************************************************************************
void ISR_Mesure_TH (void) interrupt 16
                              // déclenchement sur capture et overflow
{
   static int CP_Overflow=0;
   static int OLD_Timer_capture = 0;
   int Capture_value = 0;

   Flag_IntC = 1;     // Flag Interruption
   if ((T4CON & 0x80) !=0)
   {   T4CON &= ~0x80;
      CP_Overflow++;
   }
   if ((T4CON & 0x40) !=0)
   {   T4CON &= ~0x40;
      Capture_value = RCAP4;
      Mesure_TH = ((CP_Overflow * 65536L) + Capture_value - OLD_Timer_capture)* 0.045211;
      OLD_Timer_capture = Capture_value;
   }
```

```
    Flag_IntC = 0;
}
//****************************************************************************
//Main
//****************************************************************************
main()
{
    Init_Device();
    Timer3_Init();
    Timer2_Init();
    Timer1_Init();
    Timer4_Init();
    CFG_uart();
    EA = 1;

    Send_string(msg_test);

    while(1) {   }
}
//********************************************************************************
//  FONCTIONS APPLICATIVES
//********************************************************************************
long int Get_EVENT()
{
long int temp_CP_event;
    TR2 = 0;
    temp_CP_event = T2;
    TR2 = 1;
    return (temp_CP_event + CP_EVENT);
}
//********************************************************************************
char  putchar(char c)
{
  while (!TI0);
  TI0 = 0;
  SBUF0 = c;
  return c;
}
//********************************************************************************
void Send_string(char *msg)
{
    while (*msg != 0)
    {   putchar(*msg);
        msg++;
    }
}
```