

```

1 //*****
2 // Devoir BSE - Session 1 - Janvier 2021
3 //*****
4 //Préprocesseur
5 #include "C8051F020.h"
6 #define YES 1
7 #define NO 0
8 //*****
9 // Prototypes de fonction
10 long int Get_EVENT();
11 void Send_string(char *msg);
12 //*****
13 // Déclaration SFR16
14 sfr16 TMR3RL = 0x92; // Timer3 reload value
15 sfr16 TMR3 = 0x94; // Timer3 counter
16 sfr16 RCAP2 = 0xca; // Timer2 capture/reload
17 sfr16 T2 = 0xcc; // Timer2
18 sfr16 RCAP4 = 0xe4; // Timer4 capture/reload
19 sfr16 T4 = 0xf4; // Timer4
20 //*****
21 // GPIO
22 sbit Flag_IntA = P3^0;
23 sbit Flag_IntB = P3^1;
24 sbit Flag_IntC = P3^2;
25 sbit BP1 = P2^0; sbit BP2 = P2^1; sbit BP3 = P2^2; sbit BP4 = P2^3;
26 sbit LED1 = P2^4; sbit LED2 = P2^5; sbit LED3 = P2^6; sbit LED4 = P2^7;
27 //*****
28 // Variables globales
29 unsigned char Info_Heures, Info_Minutes, Info_Secondes, Info_Centimies;
30 long int CP_EVENT = 0;
31 long int Mesure_TH;
32 const char msg_test[] = "Message de Test\n";
33 //*****
34 void Reset_Sources_Init()
35 {
36     WDTCN = 0xDE;
37     WDTCN = 0xAD;
38 }
39 //*****
40 void Oscillator_Init() // Configuration SYSCLK = Quartz externe = 22,1184 MHz
41 {
42     int i = 0;
43     OSCXCN = 0x67;
44     for (i = 0; i < 3000; i++);
45     while ((OSCXCN & 0x80) == 0);
46     OSCICN = 0x0C;
47 }
48 //*****
49 void Port_IO_Init()
50 {
51     XBR0 = 0x04;
52     XBR1 = 0x80;
53     XBR2 = 0x5C;
54     P0MDOUT |= 0x05;
55     P2MDOUT = 0xF0;
56     P2 |= 0x0F;
57 }
58 //*****

```

```

59 void Init_Device(void)
60 {
61     Reset_Sources_Init();
62     Oscillator_Init();
63     Port_IO_Init();
64 }
65 //*****
66 void Timer3_Init()
67 {
68     TMR3RLH = 0xB8;
69     TMR3RLL = 0x00;
70     TMR3CN = 0x04;
71     EIE2 |= 0x01;
72     EIP2 &= ~0x01;
73 }
74 //*****
75 void Timer2_Init() // Timer compteur d'évènements
76 {
77     T2CON = 0x21;
78     RCAP2 = 0;
79     ET2 = 1;
80     PT2 = 1;
81     TR2 = 1;
82 }
83 //*****
84 void Timer1_Init()
85 {
86     CKCON |= 0x10;
87     PCON &= ~0x80;
88     TMOD = 0x20;
89     TH1 = 0xB8;
90     TCON |= 0x40;
91 }
92 //*****
93 void Timer4_Init()
94 {
95     CKCON |= 0x40;
96     T4CON = 0x09;
97     RCAP4 = 0;
98
99     EIE2 |= 0x04;
100    PT2 &= ~0x04;
101    T4CON |= 0x04;
102 }
103 //*****
104 void CFG_uart()
105 {
106     PCON &= ~0x80; //SMOD0: UART0 Baud Rate Divide by two Enabled.
107     T2CON &= ~0x30;
108     PCON &= 0xBF; // SSTAT0=0
109     SCON0 = 0x72; // Mode 1 - Check Stop bit - Reception validée
110                 // Transmission: octet transmis (prêt à recevoir un char
111                 // pour transmettre
112 }
113
114
115
116 //*****

```

```

117 //*****
118 void ISR_Horodatage (void) interrupt 14
119 {
120     Flag_IntA = 1;    // Flag Interruption mis à 1
121     TMR3CN &= ~0x80;
122     Info_Centimies++;
123     if (Info_Centimies >= 100)
124     {
125         Info_Centimies = 0;
126         Info_Secondes++;
127         if (Info_Secondes >= 60)
128         {
129             Info_Secondes = 0;
130             Info_Minutes++;
131             if (Info_Minutes >= 60)
132             {
133                 Info_Minutes = 0;
134                 Info_Heures++;
135                 if (Info_Heures >= 24) Info_Heures = 0;
136             }
137         }
138     }
139     Flag_IntA = 0; // Flag Interruption mis à 0
140 }
141 //*****
142 //*****
143 void ISR_CP_EVENT (void) interrupt 5
144 {
145     Flag_IntB = 1;
146     if (TF2 == 1)
147     { TF2 = 0;
148       CP_EVENT += 65536;
149     }
150     if (EXF2 == 1)
151     { EXF2 = 0;
152     }
153     Flag_IntB = 0;
154 }
155 //*****
156 //*****
157 void ISR_Mesure_TH (void) interrupt 16
158     // déclenchement sur capture et overflow
159 {
160     static int CP_Overflow=0;
161     static int OLD_Timer_capture = 0;
162     int Capture_value = 0;
163
164     Flag_IntC = 1;    // Flag Interruption
165     if ((T4CON & 0x80) != 0)
166     { T4CON &= ~0x80;
167       CP_Overflow++;
168     }
169     if ((T4CON & 0x40) != 0)
170     { T4CON &= ~0x40;
171       Capture_value = RCAP4;
172       Mesure_TH = ((CP_Overflow * 65536L) + Capture_value - OLD_Timer_capture)* 0.045211;
173       OLD_Timer_capture = Capture_value;
174     }

```

```

175   Flag_IntC = 0;
176 }
177 //*****
178 //Main
179 //*****
180 main()
181 {
182     Init_Device();
183     Timer3_Init();
184     Timer2_Init();
185     Timer1_Init();
186     Timer4_Init();
187     CFG_uart();
188     EA = 1;
189
190     Send_string(msg_test);
191
192     while(1) { }
193 }
194 //*****
195 // FONCTIONS APPLICATIVES
196 //*****
197 long int Get_EVENT()
198 {
199     long int temp_CP_event;
200     TR2 = 0;
201     temp_CP_event = T2;
202     TR2 = 1;
203     return (temp_CP_event + CP_EVENT);
204 }
205 //*****
206 char putchar(char c)
207 {
208     while (!TI0);
209     TI0 = 0;
210     SBUF0 = c;
211     return c;
212 }
213 //*****
214 void Send_string(char *msg)
215 {
216     while (*msg != 0)
217     { putchar(*msg);
218       msg++;
219     }
220 }

```