```
1   // ETML Ecole Technique
2   // Fichier GesFifoTh.c
3
4   // Exemple gestion d'un fifo de caractère, utilisation de pointeur et
5   // d'un descripteur de fifo
6
7   //  Auteur      :   C. Huber
8   //
9   //  Version     :   V1.6
10  //  Compilateur :   XC32 V1.42 + Harmony 1.08
11  //
12  //      Modifications :
13  //      CHR 19.12.2014  remplacement typedef32 par stdint
14  //      CHR 20.12.2016  fifosize en int32_t pour permettre des
15  //                      fifo de grande taille
16  //
17  /*-------------------------------------------------------*/
18  #include <stdint.h>
19  #include "app.h"
20  #include "GesFifoTh32.h"
21
22  /*---------------*/
23  /* InitFifo      */
24  /*===============*/
25
26  // Init avec possibilité de fournir une valeur de remplissage
27  // Initialisation du descripteur de FIFO
28
29  void InitFifo ( S_fifo *pDescrFifo, int32_t FifoSize, int8_t *pDebFifo, int8_t InitVal
     )
30  {
31     int32_t i;
32     int8_t *pFif;
33     pDescrFifo->fifoSize =   FifoSize;
34     pDescrFifo->pDebFifo =   pDebFifo; // début du fifo
35     // fin du fifo
36     pDescrFifo->pFinFifo =   pDebFifo + (FifoSize - 1);
37     pDescrFifo->pWrite   =   pDebFifo;  // début du fifo
38     pDescrFifo->pRead     =   pDebFifo;  // début du fifo
39     pFif = pDebFifo;
40     for (i=0; i < FifoSize; i++) {
41        *pFif  = InitVal;
42        pFif++;
43     }
44  } /* InitFifo */
45
46
47  /*---------------*/
48  /* GetWriteSpace */
49  /*===============*/
50
51  // Retourne la place disponible en écriture
52
53  int32_t GetWriteSpace ( S_fifo *pDescrFifo)
54  {
55     int32_t writeSize;
56
57     // Détermine le nb de car.que l'on peut déposer
58     writeSize = pDescrFifo->pRead - pDescrFifo->pWrite -1;
59     if (writeSize < 0) {
60        writeSize = writeSize + pDescrFifo->fifoSize;
61      }
62     return (writeSize);
63  } /* GetWriteSpace */
64
65
66  /*-------------*/
67  /* GetReadSize */
68  /*=============*/
69
70  // Retourne le nombre de caractères à lire
71
72  int32_t GetReadSize ( S_fifo *pDescrFifo)
```

```
73  {
74     int32_t readSize;
75
76     readSize = pDescrFifo->pWrite - pDescrFifo->pRead;
77     if (readSize < 0) {
78         readSize = readSize +  pDescrFifo->fifoSize;
79     }
80
81     return (readSize);
82  } /* GetReadSize */
83
84  /*---------------*/
85  /* PutCharInFifo */
86  /*===============*/
87
88  // Dépose un caractère dans le FIFO
89  // Retourne 0 si OK, 1 si FIFO full
90
91  uint8_t PutCharInFifo ( S_fifo *pDescrFifo, int8_t charToPut )
92  {
93     uint8_t writeStatus;
94
95     // test si fifo est FULL
96     if (GetWriteSpace(pDescrFifo) == 0) {
97        writeStatus = 1; // fifo FULL
98     }
99     else {
100       // écrit le caractère dans le FIFO
101       *(pDescrFifo->pWrite) = charToPut;
102
103       // incrément le pointeur d'écriture
104       pDescrFifo->pWrite++;
105       // gestion du rebouclement
106       if (pDescrFifo->pWrite > pDescrFifo->pFinFifo) {
107           pDescrFifo->pWrite = pDescrFifo->pDebFifo;
108       }
109
110       writeStatus = 0; // OK
111    }
112    return (writeStatus);
113 } // PutCharInFifo
114
115
116 /*-----------------*/
117 /* GetCharFromFifo */
118 /*=================*/
119
120 // Obtient (lecture) un caractère du fifo
121 // retourne 0 si OK, 1 si empty
122 // le caractère lu est retourné par réference
123
124 uint8_t GetCharFromFifo ( S_fifo *pDescrFifo, int8_t *carLu )
125 {
126    int8_t readSize;
127    uint8_t readStatus;
128
129    // détermine le nb de car. que l'on peut lire
130    readSize = GetReadSize(pDescrFifo);
131
132    // test si fifo est vide
133    if (readSize == 0) {
134       readStatus = 1; // fifo EMPTY
135       *carLu = 0;     // carLu = NULL
136    }
137    else {
138       // lis le caractère dans le FIFO
139       *carLu = *(pDescrFifo->pRead);
140
141       // incrément du pointeur de lecture
142       pDescrFifo->pRead++;
143       // gestion du rebouclement
144       if (pDescrFifo->pRead > pDescrFifo->pFinFifo) {
145           pDescrFifo->pRead = pDescrFifo->pDebFifo;
```

```
        }
        readStatus = 0; // OK
    }
    return (readStatus);
} // GetCharFromFifo
```