

|  |
| --- |
| **Systèmes Embarqués II** |
| Communication CAN |
| ISAT – EPHEC 2020-2021  Juan Alvarez et Olivier Grabenweger |

Table des matières

[1 Introduction 2](#_Toc60832756)

[2 Organigramme – Mindmapping 3](#_Toc60832757)

[3 Schéma de câblage 4](#_Toc60832758)

[4 Code source 4](#_Toc60832759)

[5 Conclusion 8](#_Toc60832760)

# Introduction

Dans ce TP, nous utilisons la communication CAN afin d’envoyer ou recevoir une trame.

Le matériel utilisé est le suivant (pour 1 utilisateur) :

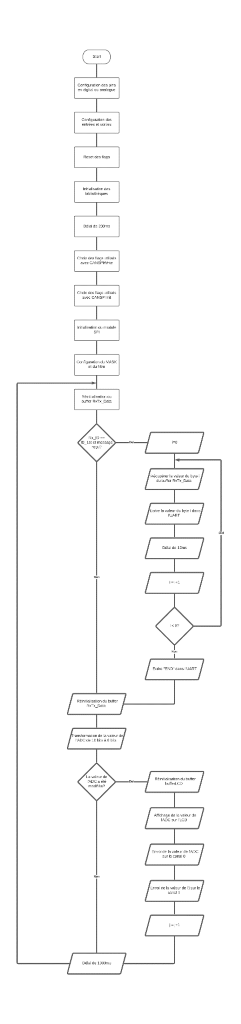
* 1 PIC18F45K22
* 1 Ecran LCD
* 2 Boutons Poussoir
* 1 ADC
* 1 Câble CAN
* 1 UART

(En sachant qu’on a besoin de minimum 2 utilisateurs donc le matériel doit être doublé).

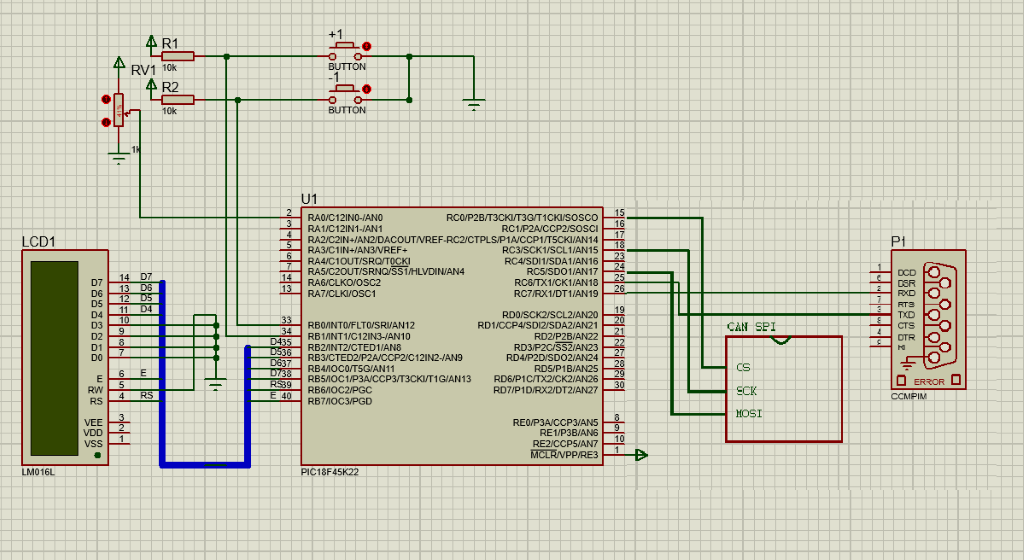
Nous avons 2 compteurs qui s’incrémentent/décrémentent.   
Le premier compteur voit sa valeur changer à l’aide d’un ADC et est affiché en temps réel sur un LCD. Le second compteur est contrôlé à l’aide de 2 boutons poussoirs. Nous utilisons le protocole de communication CAN afin d’envoyer nos valeurs sous formes de trames vers un autre utilisateur.

Sur notre moniteur série, nous affichons les données, préalablement traitées, que nous recevons de ce même utilisateur.

# Organigramme – Mindmapping



# Schéma de câblage



# Code source

|  |  |
| --- | --- |
| 1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160 | //  // Created by Juan & Oli on 5/14/2020.  //    **unsigned** **char** Can\_Init\_Flags, Can\_Send\_Flags, Can\_Rcv\_Flags; // can flags  **unsigned** **char** Rx\_Data\_Len;                                   // received data length in bytes  **char** RxTx\_Data[**8**] = {**0**};                                           // can rx/tx data buffer  **char** Msg\_Rcvd;                                               // reception flag  **const** **long** ID\_1st = **12111**, ID\_2nd = **3**;                       // node IDs  **long** Rx\_ID;  **int** recu;  **int** i,j,cpt,oldcpt;  **char** bufferLCD[**16**];  **int** nombre,oldnbr;  **char** bufferUART[**16**];    // LCD module connections  sbit LCD\_RS at RB6\_bit;  sbit LCD\_EN at RB7\_bit;  sbit LCD\_D4 at RB2\_bit;  sbit LCD\_D5 at RB3\_bit;  sbit LCD\_D6 at RB4\_bit;  sbit LCD\_D7 at RB5\_bit;  sbit LCD\_RS\_Direction at TRISB4\_bit;  sbit LCD\_EN\_Direction at TRISB5\_bit;  sbit LCD\_D4\_Direction at TRISB0\_bit;  sbit LCD\_D5\_Direction at TRISB1\_bit;  sbit LCD\_D6\_Direction at TRISB2\_bit;  sbit LCD\_D7\_Direction at TRISB3\_bit;  // End LCD module connections    // CANSPI module connections  sbit CanSpi\_CS            at  RC0\_bit;  sbit CanSpi\_CS\_Direction  at  TRISC0\_bit;  sbit CanSpi\_Rst           at  RC2\_bit;  sbit CanSpi\_Rst\_Direction at  TRISC2\_bit;  // End CANSPI module connections    **void** **Interrupt**()  {      //External Interrupt  **if**(INT0F\_bit)      {          INT0F\_bit = **0**;          cpt++;      }  **if**(INT1F\_bit)      {          INT1F\_bit = **0**;          cpt--;      }  }    **void** **main**()  {    ANSELA = **0x01**;                      // Configure AN pins as digital I/O    ANSELB = **0**;    ANSELC = **0**;    ANSELD = **0**;      C1ON\_bit = **0**;               // Disable comparators    C2ON\_bit = **0**;      TRISD = **0x00**;    TRISA = **0x01**;                          // OUTPUT = 0 Iput = 1    TRISB = **0x03**;    TRISC = **0x00**;    Can\_Init\_Flags = **0**;                                         //    Can\_Send\_Flags = **0**;                                         // clear flags    Can\_Rcv\_Flags  = **0**;                                         //    ADC\_Init();    Lcd\_Init();    UART1\_Init(**9600**);      Delay\_ms(**200**);      Can\_Send\_Flags = \_CANSPI\_TX\_PRIORITY\_0 &                    // form value to be used                     \_CANSPI\_TX\_XTD\_FRAME &                     //     with CANSPIWrite                     \_CANSPI\_TX\_NO\_RTR\_FRAME;      Can\_Init\_Flags = \_CANSPI\_CONFIG\_SAMPLE\_THRICE &             // Form value to be used                     \_CANSPI\_CONFIG\_PHSEG2\_PRG\_ON &             //  with CANSPIInit                     \_CANSPI\_CONFIG\_XTD\_MSG &                     \_CANSPI\_CONFIG\_DBL\_BUFFER\_ON &                     \_CANSPI\_CONFIG\_VALID\_XTD\_MSG;      SPI1\_Init();                                                     // initialize SPI1 module      CANSPIInitialize(**1**,**3**,**3**,**3**,**1**,Can\_Init\_Flags);                      // Initialize external CANSPI module    CANSPISetOperationMode(\_CANSPI\_MODE\_CONFIG,**0xFF**);                // set CONFIGURATION mode    CANSPISetMask(\_CANSPI\_MASK\_B1,-**1**,\_CANSPI\_CONFIG\_XTD\_MSG);        // set all mask1 bits to ones    CANSPISetMask(\_CANSPI\_MASK\_B2,-**1**,\_CANSPI\_CONFIG\_XTD\_MSG);        // set all mask2 bits to ones    CANSPISetFilter(\_CANSPI\_FILTER\_B2\_F4,ID\_1st,\_CANSPI\_CONFIG\_XTD\_MSG);// set id of filter B2\_F4 to 2nd node ID      CANSPISetOperationMode(\_CANSPI\_MODE\_NORMAL,**0xFF**);                // set NORMAL mode      //RxTx\_Data[0] = {};                                                // set initial data to be sent     Lcd\_Cmd(\_LCD\_CLEAR);     Lcd\_Out(**1**,**1**,"yol0");      //External      INTEDG0\_bit = **1**;      INT0E\_bit = **0**;      INT0E\_bit = **1**;      PEIE\_bit  = **1**;                    // Enable peripheral interrupts      GIE\_bit   = **1**;                    // Enable GLOBAL interrupts    **for**(;;)      {      memset(Rxtx\_Data,**0**,**sizeof**(RxTx\_Data));        Msg\_Rcvd = CANSPIRead(&Rx\_ID , RxTx\_Data , &Rx\_Data\_Len, &Can\_Rcv\_Flags);// receive message  **if** ((Rx\_ID == ID\_1st) && Msg\_Rcvd) // if message received check id          {    **for**(i=**0**; i < **8** ; i++)          {             recu = RxTx\_Data[i];            //memset(bufferUART,0,sizeof(bufferUART));              sprintf(bufferUART,"nbr rcvd : %04d",recu);            UART1\_Write\_Text(bufferUART);            UART1\_Write\_Text("**\n\r**");            delay\_ms(**10**);            }          UART1\_Write\_Text("end");          UART1\_Write\_Text("**\n\r**");        }        //}        memset(Rxtx\_Data,**0**,**sizeof**(RxTx\_Data));        nombre = ADC\_Read(**0**) >> **2**;  **if**((oldnbr != nombre)||(oldcpt != cpt))        {        // vérifie nouvelle valeur pot          oldnbr = nombre;          oldcpt = cpt;          memset(bufferLCD,**0**,**sizeof**(bufferLCD));          sprintf(bufferLCD,"nbr:%04d & cpt : %04d", nombre,cpt);          Lcd\_Out(**2**,**1**,bufferLCD);          Lcd\_Cmd(\_LCD\_CURSOR\_OFF);          RxTx\_Data [**0**] = nombre;          RxTx\_Data[**1**] = cpt;          CANSPIWrite(ID\_2nd, RxTx\_Data, **8**, Can\_Send\_Flags);     // send data          j++;        }        delay\_ms(**1000**);      }  } |

Analyse du code :   
Le fonctionnement du code est le suivant :

Première étape 🡪 déclaration et initialisation des différentes variables

Ensuite on règle les différents flags du CANSPI

* + Une fonction très importante est la suivante :   
    CANSPISetFilter (\_CANSPI\_FILTER\_B2\_F4,ID\_1st,\_CANSPI\_CONFIG\_XTD\_MSG);

Les arguments sont les suivants : (Type de filtre – id à filtrer – format du msg)

Ensuite on rentre dans la boucle où :

1) Deux interruptions externes sont utilisées pour incrémenter deux compteurs    
2) Lecture ADC

3) On vérifie si une valeur a changé

4)si oui on formate et on affiche la nouvelle valeur sur le LCD

Et finalement on envoie via CANSPI les deux compteurs s'ils ont changé sur le premier et le deuxième byte. 

# Conclusion

La réalisation de ce premier tp aura permis une approche rapide mais efficace du protocole CAN.

Et même si l’application de celui-ci dans ce premier tp reste assez simple, nous pensons que le travail réalisé peut être étendu à une application plus complexe.

Nous regrettons de ne pas avoir assez bien compris toutes les différentes constantes disponibles dans la configuration du protocole, mais espérons pouvoir revenir dessus ultérieurement.