Robotech 2021 - CAN

Généré par Doxygen 1.9.3

Chapitre 1

README

pour crée un bus can virtuel (vcan0) :

- sudo modprobe vcan
- sudo ip link add dev vcan0 type vcan
- sudo ip link set up vcan0

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la classe Can

`#include <canClass.h>`

Fonctions membres publiques

- Can ()
- int init (uint myAddr)
  initialise le bus can
- bool is_valid_addr (uint addr)
  regarde si l'adresse est connue
- bool is_valid_code_fct (uint codeFct)
  regarde si le code fct est connue
- int send (uint addr, uint codeFct, uint8_t data[ ], uint data_len, bool isRep, uint repLenght)
  envoie un message sur le bus can
- void traitement (CanResponse_t msg)
  traite le message décodé dans un nouveau thread
- int start_listen ()
  démarre le thread d'écoute du bus can

4.1.1 Description détaillée

Définition à la ligne 40 du fichier canClass.h.

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 Can()

`Can::Can ( )`

Définition à la ligne 33 du fichier canClass.cpp.

4.1.3 Documentation des fonctions membres

4.1.3.1 init()

```
int Can::init (
            uint myAddr )
```

initialise le bus can

Paramètres

myAddr	adresse sur le bus can

Valeurs retournées

0	: succes
{CAN_E_SOCKET_ERROR}	: erreur dans l'ouverture du socket
{CAN_E_BIND_ERROR}	: erreur dans le bind du bus
{CAN_E_OOB_ADDR}	: adresse en dehors des bornes
{CAN_E_UNKNOW_ADDR}	: l'adresse n'est pas dans le #define

Définition à la ligne 48 du fichier canClass.cpp.

4.1.3.2 is_valid_addr()

```
bool Can::is_valid_addr (
            uint addr )
```

regarde si l'adresse est connue

Paramètres

myAddr	addresse à verifier

Valeurs retournées

true	: l'adresse est connue
false	: l'adresse n'est pas connue

Définition à la ligne 92 du fichier canClass.cpp.

4.1.3.3 is_valid_code_fct()

```
bool Can::is_valid_code_fct (
            uint codeFct )
```

regarde si le code fct est connue

Paramètres

codeFct	code fct à verifier

Valeurs retournées

true	: le code fct est connue
false	: le code fct n'est pas connue

Définition à la ligne 107 du fichier canClass.cpp.

4.1.3.4 send()

```
int Can::send (
            uint addr,
            uint codeFct,
            uint8_t data[],
            uint data_len,
            bool isRep,
            uint repLenght )
```

envoie un message sur le bus can

Paramètres

addr	addresse du destinataire
codeFct	code fct du message
data	tableau d'octet d'une taille 0 ... 8
data_len	nombre d'octet de data, compris entre 0 et 8
isRep	true si le message est une réponse à une requete, false sinon
repLenght	isRep = true : id du msg dans la réponse. isRep = false : nbr de reponse atendu

Valeurs retournées

0	: le message a bien etait envoyé
{CAN_E_DATA_SIZE_TOO_LONG}	: data_len n'est pas compris entre 0 et 8 inclu
{CAN_E_OOB_ADDR}	: l'adresse n'est pas dans les valeurs possible (0 - {CAN_MAX_VALUE_ADDR})
{CAN_E_OOB_CODE_FCT}	: le code fct n'est pas dans les valeurs possible (0 - {CAN_MAX_VALUE_CODE_FCT})

Valeurs retournées

{CAN_E_OOB_REP_NBR}	: le rep nbr n'est pas dans les valeurs possible (0 - {CAN_MAX_VALUE_REP_NBR})
{CAN_E_OOB_DATA}	: au moins une des donnés n'est pas dans les valeurs possible (0 - 255)
{CAN_E_UNKNOW_ADDR}	: l'adresse n'est pas dans le #define
{CAN_E_UNKNOW_CODE_FCT}	: le code fonction n'est pas dans le #define
{CAN_E_WRITE_ERROR}	: une erreur à eu lieu lors de l'envoie du message

Définition à la ligne 134 du fichier canClass.cpp.

4.1.3.5 start_listen()

```
int Can::start_listen ( )
```

démarre le thread d'écoute du bus can

Valeurs retournées

0	: le thread a bien etait lancer

Définition à la ligne 291 du fichier canClass.cpp.

4.1.3.6 traitement()

```
void Can::traitement (
            CanResponse_t msg )
```

traite le message décodé dans un nouveau thread

Paramètres

msg	structure contenant le message decoder

Définition à la ligne 263 du fichier canClass.cpp.

La documentation de cette classe a été générée à partir du fichier suivant :
- canClass.h
- canClass.cpp

4.2 Référence de la classe CAN

classe qui gère l'envoie et la réception de msg via un bus can

```
#include <canClass.h>
```

4.2.1 Description détaillée

classe qui gère l'envoie et la réception de msg via un bus can

Paramètres

s	idantifiant du gestionnaire du bus
threadListen	objet du thread d'écoute du bus can

La documentation de cette classe a été générée à partir du fichier suivant :

— canClass.h

4.3 Référence de la structure CanResponse_t

classe de gestion d'un bus can

`#include <canClass.h>`

Attributs publics

— uint addr
— uint emetteur
— uint codeFct
— bool isRep
— uint RepId
— uint dataLen
— char data [8]

4.3.1 Description détaillée

classe de gestion d'un bus can

Cette classe permet d'envoyer et de recevoir des messages via un bus can

Auteur

Theo RUSINOWITCH theo.rusinowitch1@etu.univ-lorraine.fr

Version

4.1a

Date

2021-2022

contient un message décoder venant du bus can

Paramètres

addr	addresse du destinataire du message
emetteur	adresse de l'émeteur du message
codeFct	code fonction du message
isRep	vrai si c'est une reponse a une requete, faux sinon
RepId	indique le nombre de reponse atendu pour une requete et le num de la reponse pour une reponse
dataLen	frame payload length in byte (0 .. 8) aka data length code
data	CAN frame payload (up to 8 byte)

Définition à la ligne 23 du fichier canClass.h.

4.3.2 Documentation des données membres

4.3.2.1 addr

```
uint CanResponse_t::addr
```

Définition à la ligne 24 du fichier canClass.h.

4.3.2.2 codeFct

```
uint CanResponse_t::codeFct
```

Définition à la ligne 26 du fichier canClass.h.

4.3.2.3 data

```
char CanResponse_t::data[8]
```

Définition à la ligne 30 du fichier canClass.h.

4.3.2.4 dataLen

```
uint CanResponse_t::dataLen
```

Définition à la ligne 29 du fichier canClass.h.

4.3.2.5 emetteur

```
uint CanResponse_t::emetteur
```

Définition à la ligne 25 du fichier canClass.h.

4.3.2.6 isRep

```
bool CanResponse_t::isRep
```

Définition à la ligne 27 du fichier canClass.h.

4.3.2.7 RepId

```
uint CanResponse_t::RepId
```

Définition à la ligne 28 du fichier canClass.h.

La documentation de cette structure a été générée à partir du fichier suivant :

— canClass.h

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier canClass.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <iostream>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>
#include <thread>
#include "canClass.h"
#include "defineCan.h"
```

5.2 canClass.cpp

Aller à la documentation de ce fichier.

```
00001
00009 #include <stdio.h>
00010 #include <stdlib.h>
00011 #include <string.h>
00012 #include <unistd.h>
00013 #include <iostream>
00014
00015 #include <net/if.h>
00016 #include <sys/ioctl.h>
00017 #include <sys/socket.h>
00018
00019 #include <linux/can.h>
00020 #include <linux/can/raw.h>
00021
00022
00023 #include <unistd.h>
00024
00025 #include <thread>
00026
00027
00028
00029 #include "canClass.h"
00030 #include "defineCan.h"
00031 using namespace std;
00032
```

```
00033 Can::Can(){
00034
00035     return;
00036 }
00037
00038
00048 int Can::init(uint myAddr){
00049     int i;
00050     int nbytes;
00051     struct sockaddr_can addr;
00052     struct ifreq ifr;
00053
00054
00055     if ((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
00056         perror("Socket");
00057         return CAN_E_SOCKET_ERROR;
00058     }
00059     strcpy(ifr.ifr_name, CAN_BUS_NAME); //definie le bus sur le quel on travaille (can0 ou vcan0
      normalement)
00060     ioctl(s, SIOCGIFINDEX, &ifr);
00061
00062     memset(&addr, 0, sizeof(addr));
00063     addr.can_family = AF_CAN;
00064     addr.can_ifindex = ifr.ifr_ifindex;
00065
00066     if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
00067         perror("Bind");
00068         return CAN_E_BIND_ERROR;
00069     }
00070
00071     // si l'address est incorect on ne setup juste pas de filtre
00072     if(myAddr <0 || myAddr > CAN_MAX_VALUE_ADDR) return CAN_E_OOB_ADDR;
00073     if(!is_valid_addr(myAddr)) return CAN_E_UNKNOW_ADDR;
00074
00075     //  Initialisation de l'adresse
00076     struct can_filter rfilter[1];
00077     rfilter[0].can_id   = myAddr « CAN_DECALAGE_ADDR_RECEPTEUR;
00078     rfilter[0].can_mask = CAN_FILTER_ADDR_RECEPTEUR;
00079
00080
00081     setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));
00082
00083     return 0;
00084 }
00085
00092 bool Can::is_valid_addr(uint addr){
00093     int size = sizeof(CAN_LIST_ADDR)/sizeof(CAN_LIST_ADDR[0]);
00094     for (int i = 0; i < size; i++)
00095     {
00096         if(addr == CAN_LIST_ADDR[i]) return true;
00097     }
00098     return false;
00099 }
00100
00107 bool Can::is_valid_code_fct(uint codeFct){
00108     int size = sizeof(CAN_LIST_CODE_FCT)/sizeof(CAN_LIST_CODE_FCT[0]);
00109     for (int i = 0; i < size; i++)
00110     {
00111         if(codeFct == CAN_LIST_CODE_FCT[i]) return true;
00112     }
00113     return false;
00114 }
00115
00134 int Can::send(uint addr, uint codeFct , uint8_t data[], uint data_len, bool isRep, uint repLenght){
00135     if (data_len >8){
00136         cout « "vous ne pouvez envoyer que 8 octet de data" « endl;
00137         return CAN_E_DATA_SIZE_TOO_LONG;
00138     }
00139
00140     if(addr < 0 || addr > CAN_MAX_VALUE_ADDR) return CAN_E_OOB_ADDR;
00141     if(codeFct < 0 || codeFct > CAN_MAX_VALUE_CODE_FCT) return CAN_E_OOB_CODE_FCT;
00142     if(repLenght < 0 || repLenght > CAN_MAX_VALUE_REP_NBR) return CAN_E_OOB_REP_NBR;
00143
00144     if(!is_valid_addr(addr)) return CAN_E_UNKNOW_ADDR;
00145     if(!is_valid_code_fct(codeFct)) return CAN_E_UNKNOW_CODE_FCT;
00146
00147
00148
00149
00150     struct can_frame frame;
00151     frame.can_id = (
00152           (addr « CAN_DECALAGE_ADDR_RECEPTEUR)
00153         + (CAN_ADDR_RASPBERRY « CAN_DECALAGE_ADDR_EMETTEUR)
00154         + (codeFct « CAN_DECALAGE_CODE_FCT)
00155         + (isRep « CAN_DECALAGE_IS_REP)
00156         + (repLenght « CAN_DECALAGE_REP_NBR)
00157         )| CAN_EFF_FLAG;
```

```
00158
00159     cout « hex « frame.can_id « endl;
00160     frame.can_dlc = data_len;
00161
00162     printf("send : 0x%03X [%d] ",frame.can_id, frame.can_dlc);
00163     for (int i = 0; i < data_len; i++)
00164     {
00165         if(data[i] <0 || data[i] > 255) return CAN_E_OOB_DATA;
00166         frame.data[i] = data[i];
00167         printf("%02X ",frame.data[i]);
00168     }
00169     printf("\r\n");
00170
00171     if (write(s, &frame, sizeof(struct can_frame)) != sizeof(struct can_frame)) { //on verifie que le
      nombre de byte envoyer est corecte
00172         perror("Write");
00173         return CAN_E_WRITE_ERROR;
00174     }
00175
00176     return 0;
00177 }
00178
00192 int Can::traitement_trame(CanResponse_t &rep, can_frame frame){
00193
00194     int nbytes;
00195         nbytes = read(s, &frame, sizeof(struct can_frame));
00196         if (nbytes < 0) {
00197             perror("Read");
00198             return CAN_E_READ_ERROR;
00199         }
00200
00201         rep.addr = (frame.can_id & CAN_FILTER_ADDR_RECEPTEUR ) » CAN_DECALAGE_ADDR_RECEPTEUR ;
00202         rep.emetteur = (frame.can_id & CAN_FILTER_ADDR_EMETTEUR) » CAN_DECALAGE_ADDR_EMETTEUR;
00203         rep.codeFct = (frame.can_id & CAN_FILTER_CODE_FCT) » CAN_DECALAGE_CODE_FCT;
00204         rep.isRep = (frame.can_id & CAN_FILTER_IS_REP) » CAN_DECALAGE_IS_REP;
00205         rep.RepId = (frame.can_id & CAN_FILTER_REP_NBR) » CAN_DECALAGE_REP_NBR;
00206
00207
00208         if(rep.addr <0 || rep.addr > CAN_MAX_VALUE_ADDR) return CAN_E_OOB_ADDR;
00209         if(rep.codeFct <0 || rep.codeFct > CAN_MAX_VALUE_CODE_FCT) return CAN_E_OOB_CODE_FCT;
00210         if(rep.RepId <0 || rep.RepId > CAN_MAX_VALUE_REP_NBR) return CAN_E_OOB_REP_NBR;
00211
00212         if(!is_valid_addr(rep.addr)) return CAN_E_UNKNOW_ADDR;
00213         if(!is_valid_addr(rep.emetteur)) return CAN_E_UNKNOW_ADDR;
00214         if(!is_valid_code_fct(rep.codeFct)) return CAN_E_UNKNOW_CODE_FCT;
00215         if (frame.can_dlc >8)  return CAN_E_DATA_SIZE_TOO_LONG;
00216         rep.dataLen = frame.can_dlc;
00217
00218
00219         for (int i = 0; i < frame.can_dlc; i++){
00220             if(frame.data[i] <0 || frame.data[i] > 255) return CAN_E_OOB_DATA;
00221             rep.data[i] = frame.data[i];
00222         }
00223
00224
00225     return 0;
00226 }
00227
00232 void Can::listen(){
00233     while(true){
00234         struct can_frame frame;
00235         struct CanResponse_t reponse;
00236
00237         int err =traitement_trame(reponse, frame);
00238         if(err < 0){
00239             cout « "erreur dans le décodage d'une trame. err n°" « dec « err « "\t\t c.f. #define" «
      endl;
00240             continue;
00241         }
00242
00243
00244         cout « "addr : " « hex « reponse.addr ;
00245         cout « "   emetteur : " « hex « reponse.emetteur;
00246         cout « "   codeFct : " « hex « reponse.codeFct;
00247         cout « "   isRep : " « reponse.isRep;
00248         cout « "   RepId : " « hex « reponse.RepId;
00249         cout « "       Data : ["« (int)reponse.dataLen «"] ";
00250         for (int i = 0; i < frame.can_dlc; i++) printf("%02X ",reponse.data[i]);
00251         printf("\r\n");
00252
00253         thread test(&Can::traitement,this, reponse);
00254         test.detach();
00255
00256     }
00257 }
00258
00263 void Can::traitement(CanResponse_t msg){
```

```
00264     switch (msg.emetteur){
00265     case CAN_ADDR_BASE_ROULANTE:
00266         switch (msg.codeFct){
00267         case AVANCE:
00268
00269
00270         break;
00271         default:
00272             cout « "code fonction inconu" « endl;
00273         break;
00274         }
00275
00276
00277     break;
00278     case CAN_ADDR_RASPBERRY:
00279         cout « "self emeteur" « endl;
00280     break;
00281     default:
00282         cout « "emetteur inconu" « endl;
00283     break;
00284     }
00285 }
00286
00291 int Can::start_listen(){
00292     thread tListen(&Can::listen, this);
00293     tListen.detach();
00294     threadListen = &tListen;
00295     return 0;
00296 }
00297
00298
00299
```

5.3 Référence du fichier canClass.h

```
#include <thread>
#include <linux/can.h>
```

Classes

— struct CanResponse_t
classe de gestion d'un bus can
— class Can

5.4 canClass.h

Aller à la documentation de ce fichier.

```
00001
00008 #if !defined CANCLASS_H
00009 #define CANCLASS_H
00010 #include <thread>
00011 #include <linux/can.h>
00023 struct CanResponse_t{
00024     uint addr;  /* addresse du destinataire du message */
00025     uint emetteur;  /* adresse de l'éméteur */
00026     uint codeFct;    /* code fonction du msg */
00027     bool isRep;    /* vrai si c'est une reponse a une requete, faux sinon */
00028     uint RepId;    /* nb de rep atendu si requete, num de la rep si reponse */
00029     uint dataLen;    /* frame payload length in byte (0 .. 8) */
00030     char data[8];
00031 };
00032
00033
00040 class Can{
00041   private:
00042     int s;
00043     std::thread* threadListen;
00044     void listen();
```

```
00045     int traitement_trame(CanResponse_t &rep, can_frame frame);
00046   public:
00047
00048     Can();
00049     int init(uint myAddr);
00050     bool is_valid_addr(uint addr);
00051     bool is_valid_code_fct(uint codeFct);
00052     int send(uint addr, uint codeFct , uint8_t data[], uint data_len, bool isRep, uint repLenght);
00053     void traitement(CanResponse_t msg);
00054     int start_listen();
00055 };
00056
00057
00058
00059
00060 #endif
```

5.5 Référence du fichier defineCan.h

Macros

- #define CAN_BUS_NAME "vcan0"
- #define CAN_FILTER_ADDR_EMETTEUR 0b00000000000111100000000000000000
- #define CAN_FILTER_ADDR_RECEPTEUR 0b00000000000000000111100000000000
- #define CAN_FILTER_CODE_FCT 0b00000000000000000000011111110000
- #define CAN_FILTER_IS_REP 0b00000000000000000000000000001000
- #define CAN_FILTER_REP_NBR 0b00000000000000000000000000000111
- #define CAN_DECALAGE_ADDR_EMETTEUR 15
- #define CAN_DECALAGE_ADDR_RECEPTEUR 11
- #define CAN_DECALAGE_CODE_FCT 4
- #define CAN_DECALAGE_IS_REP 3
- #define CAN_DECALAGE_REP_NBR 0
- #define CAN_MAX_VALUE_ADDR 16
- #define CAN_MAX_VALUE_CODE_FCT 128
- #define CAN_MAX_VALUE_REP_NBR 8
- #define CAN_ADDR_RASPBERRY 1
- #define CAN_ADDR_BASE_ROULANTE 2
- #define CAN_LIST_ADDR (int[ ]){CAN_ADDR_RASPBERRY, CAN_ADDR_BASE_ROULANTE}
- #define AVANCE 0x01
- #define REP_AVANCE 0x81
- #define CAN_LIST_CODE_FCT (int[ ]){AVANCE, REP_AVANCE}
- #define CAN_E_WRITE_ERROR -501
- #define CAN_E_READ_ERROR -502
- #define CAN_E_SOCKET_ERROR -503
- #define CAN_E_BIND_ERROR -504
- #define CAN_E_DATA_SIZE_TOO_LONG -510
- #define CAN_E_OOB_ADDR -511
- #define CAN_E_OOB_CODE_FCT -512
- #define CAN_E_OOB_REP_NBR -513
- #define CAN_E_OOB_DATA -514
- #define CAN_E_UNKNOW_ADDR -551
- #define CAN_E_UNKNOW_CODE_FCT -552

5.5.1 Documentation des macros

5.5.1.1 AVANCE

```
#define AVANCE 0x01
```

Définition à la ligne 40 du fichier defineCan.h.

5.5.1.2 CAN_ADDR_BASE_ROULANTE

```
#define CAN_ADDR_BASE_ROULANTE 2
```

Définition à la ligne 34 du fichier defineCan.h.

5.5.1.3 CAN_ADDR_RASPBERRY

```
#define CAN_ADDR_RASPBERRY 1
```

Définition à la ligne 33 du fichier defineCan.h.

5.5.1.4 CAN_BUS_NAME

```
#define CAN_BUS_NAME "vcan0"
```

Définition à la ligne 5 du fichier defineCan.h.

5.5.1.5 CAN_DECALAGE_ADDR_EMETTEUR

```
#define CAN_DECALAGE_ADDR_EMETTEUR 15
```

Définition à la ligne 17 du fichier defineCan.h.

5.5.1.6 CAN_DECALAGE_ADDR_RECEPTEUR

```
#define CAN_DECALAGE_ADDR_RECEPTEUR 11
```

Définition à la ligne 18 du fichier defineCan.h.

5.5.1.7 CAN_DECALAGE_CODE_FCT

```
#define CAN_DECALAGE_CODE_FCT 4
```

Définition à la ligne 19 du fichier defineCan.h.

5.5.1.8 CAN_DECALAGE_IS_REP

```
#define CAN_DECALAGE_IS_REP 3
```

Définition à la ligne 21 du fichier defineCan.h.

5.5.1.9 CAN_DECALAGE_REP_NBR

```
#define CAN_DECALAGE_REP_NBR 0
```

Définition à la ligne 22 du fichier defineCan.h.

5.5.1.10 CAN_E_BIND_ERROR

```
#define CAN_E_BIND_ERROR -504
```

Définition à la ligne 58 du fichier defineCan.h.

5.5.1.11 CAN_E_DATA_SIZE_TOO_LONG

```
#define CAN_E_DATA_SIZE_TOO_LONG -510
```

Définition à la ligne 61 du fichier defineCan.h.

5.5.1.12 CAN_E_OOB_ADDR

```
#define CAN_E_OOB_ADDR -511
```

Définition à la ligne 62 du fichier defineCan.h.

5.5.1.13 CAN_E_OOB_CODE_FCT

```
#define CAN_E_OOB_CODE_FCT -512
```

Définition à la ligne 63 du fichier defineCan.h.

5.5.1.14 CAN_E_OOB_DATA

```
#define CAN_E_OOB_DATA -514
```

Définition à la ligne 65 du fichier defineCan.h.

5.5.1.15 CAN_E_OOB_REP_NBR

```
#define CAN_E_OOB_REP_NBR -513
```

Définition à la ligne 64 du fichier defineCan.h.

5.5.1.16 CAN_E_READ_ERROR

```
#define CAN_E_READ_ERROR -502
```

Définition à la ligne 56 du fichier defineCan.h.

5.5.1.17 CAN_E_SOCKET_ERROR

```
#define CAN_E_SOCKET_ERROR -503
```

Définition à la ligne 57 du fichier defineCan.h.

5.5.1.18 CAN_E_UNKNOW_ADDR

```
#define CAN_E_UNKNOW_ADDR -551
```

Définition à la ligne 68 du fichier defineCan.h.

5.5.1.19 CAN_E_UNKNOW_CODE_FCT

```
#define CAN_E_UNKNOW_CODE_FCT -552
```

Définition à la ligne 69 du fichier defineCan.h.

5.5.1.20 CAN_E_WRITE_ERROR

```
#define CAN_E_WRITE_ERROR -501
```

Définition à la ligne 55 du fichier defineCan.h.

5.5.1.21 CAN_FILTER_ADDR_EMETTEUR

```
#define CAN_FILTER_ADDR_EMETTEUR 0b00000000000011110000000000000000
```

Définition à la ligne 11 du fichier defineCan.h.

5.5.1.22 CAN_FILTER_ADDR_RECEPTEUR

```
#define CAN_FILTER_ADDR_RECEPTEUR 0b00000000000000000111100000000000
```

Définition à la ligne 12 du fichier defineCan.h.

5.5.1.23 CAN_FILTER_CODE_FCT

```
#define CAN_FILTER_CODE_FCT 0b00000000000000000000011111110000
```

Définition à la ligne 13 du fichier defineCan.h.

5.5.1.24 CAN_FILTER_IS_REP

```
#define CAN_FILTER_IS_REP 0b00000000000000000000000000001000
```

Définition à la ligne 14 du fichier defineCan.h.

5.5.1.25 CAN_FILTER_REP_NBR

```
#define CAN_FILTER_REP_NBR 0b00000000000000000000000000000111
```

Définition à la ligne 15 du fichier defineCan.h.

5.5.1.26 CAN_LIST_ADDR

```
#define CAN_LIST_ADDR (int[]){CAN_ADDR_RASPBERRY, CAN_ADDR_BASE_ROULANTE}
```

Définition à la ligne 37 du fichier defineCan.h.

5.5.1.27 CAN_LIST_CODE_FCT

```
#define CAN_LIST_CODE_FCT (int[]){AVANCE, REP_AVANCE}
```

Définition à la ligne 48 du fichier defineCan.h.

5.5.1.28 CAN_MAX_VALUE_ADDR

```
#define CAN_MAX_VALUE_ADDR 16
```

Définition à la ligne 25 du fichier defineCan.h.

5.5.1.29 CAN_MAX_VALUE_CODE_FCT

```
#define CAN_MAX_VALUE_CODE_FCT 128
```

Définition à la ligne 26 du fichier defineCan.h.

5.5.1.30 CAN_MAX_VALUE_REP_NBR

```
#define CAN_MAX_VALUE_REP_NBR 8
```

Définition à la ligne 27 du fichier defineCan.h.

5.5.1.31 REP_AVANCE

```
#define REP_AVANCE 0x81
```

Définition à la ligne 45 du fichier defineCan.h.

5.6 defineCan.h

Aller à la documentation de ce fichier.

```
00001 #ifndef DEFINE_CAN_H
00002 #define DEFINE_CAN_H
00003
00004
00005 #define CAN_BUS_NAME "vcan0"
00006
00007
00008 //                                0b00000000000000000000000000000000
00009 //                                Ob|1  |5   |10  |15  |20  |25  |29
00010 //                                Ob|29  |25  |20  |15  |10  |5  |1
00011 #define CAN_FILTER_ADDR_EMETTEUR    0b00000000000111100000000000000000
00012 #define CAN_FILTER_ADDR_RECEPTEUR   0b00000000000000000111100000000000
00013 #define CAN_FILTER_CODE_FCT         0b00000000000000000000011111110000
00014 #define CAN_FILTER_IS_REP           0b00000000000000000000000000001000
00015 #define CAN_FILTER_REP_NBR          0b00000000000000000000000000000111
00016
00017 #define CAN_DECALAGE_ADDR_EMETTEUR    15
00018 #define CAN_DECALAGE_ADDR_RECEPTEUR  11
00019 #define CAN_DECALAGE_CODE_FCT       4
00020 //#define CAN_DECALAGE_ID_MSG         4
00021 #define CAN_DECALAGE_IS_REP         3
00022 #define CAN_DECALAGE_REP_NBR    0
00023
00024
00025 #define CAN_MAX_VALUE_ADDR  16
00026 #define CAN_MAX_VALUE_CODE_FCT  128
00027 #define CAN_MAX_VALUE_REP_NBR   8
00028
00029
00030
00031
00032 //definition des addresses sur le bus can (de type 0xX00)
00033 #define CAN_ADDR_RASPBERRY 1
00034 #define CAN_ADDR_BASE_ROULANTE 2
00035
00036
00037 #define CAN_LIST_ADDR (int[]){CAN_ADDR_RASPBERRY, CAN_ADDR_BASE_ROULANTE}
00038
00039 //definition des codes fonctions
00040 #define AVANCE 0x01
00041
00042
00043 //definition des codes fonctions de reponse
00044
00045 #define REP_AVANCE 0x81
00046
00047
00048 #define CAN_LIST_CODE_FCT (int[]){AVANCE, REP_AVANCE}
00049
00050 //definition des code erreur
00051
00052 // OOB => Out Of Bound
00053
00054
00055 #define CAN_E_WRITE_ERROR -501
00056 #define CAN_E_READ_ERROR -502
00057 #define CAN_E_SOCKET_ERROR -503
00058 #define CAN_E_BIND_ERROR -504
00059
00060 //  11X -> taille valeur
00061 #define CAN_E_DATA_SIZE_TOO_LONG -510
00062 #define CAN_E_OOB_ADDR -511
00063 #define CAN_E_OOB_CODE_FCT -512
00064 #define CAN_E_OOB_REP_NBR -513
00065 #define CAN_E_OOB_DATA -514
00066
00067 // 12X -> unknow
00068 #define CAN_E_UNKNOW_ADDR -551
00069 #define CAN_E_UNKNOW_CODE_FCT -552
00070
00071
00072
00073 #endif
```

5.7 Référence du fichier main.cpp

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include <iostream>
#include "canClass.h"
#include <thread>
#include "defineCan.h"
```

Fonctions

— void wait (int miliseconde)
— int main (int argc, char **argv)

5.7.1 Documentation des fonctions

5.7.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Définition à la ligne 27 du fichier main.cpp.

5.7.1.2 wait()

```
void wait (
            int miliseconde )
```

Définition à la ligne 20 du fichier main.cpp.

5.8 main.cpp

Aller à la documentation de ce fichier.

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003
00004 #include <unistd.h>
00005 #include <iostream>
00006
00007 #include <unistd.h>
00008
00009
00010 #include "canClass.h"
00011
00012 #include <thread>
00013
00014
00015
00016 #include "defineCan.h"
00017 using namespace std;
00018
00019
00020 void wait(int miliseconde){
```

```
00021     usleep(miliseconde*1000);
00022     return;
00023 }
00024
00025
00026
00027 int main(int argc, char **argv)
00028 {
00029
00030     Can can;
00031     int err = can.init(CAN_ADDR_RASPBERRY);
00032     if(err <0){
00033         cout « "erreur dans l'init du bus can. err n°" « dec « err « "\t\t c.f. #define" « endl;
00034         return;
00035     }
00036
00037     can.start_listen();
00038
00039     uint8_t data[8] = {0x01,0x02,0xFF,0x34,0x45};
00040
00041     int err = can.send(CAN_ADDR_RASPBERRY, AVANCE, data, 5, true, 5) ;
00042     if(err < 0){
00043         cout « "erreur dans l'envoie d'une trame. err n°" « dec « err « "\t\t c.f. #define" « endl;
00044     }
00045     while(true){
00046         //cout « endl;
00047
00048         //can.listen();
00049         wait(1000);
00050
00051     }
00052 }
00053
00054
```

5.9 Référence du fichier README.md

Index