

Fiche guide 1	TS SI		P.P.E. Robot solaire	
Analyse - Conception	4h			
	Mise en œuvre du bus I2C			

Nom(s) :	Classe :	Groupe :
----------	----------	----------

Objectifs

A la fin de cette étude :

- Vous serez capable d'établir un programme écrit en langage C afin de commander des circuits électroniques reliés par un bus I2C.

Matériels : Carte SSI + carte interface I2C.

Logiciel : CodeVision AVR.

Documentation

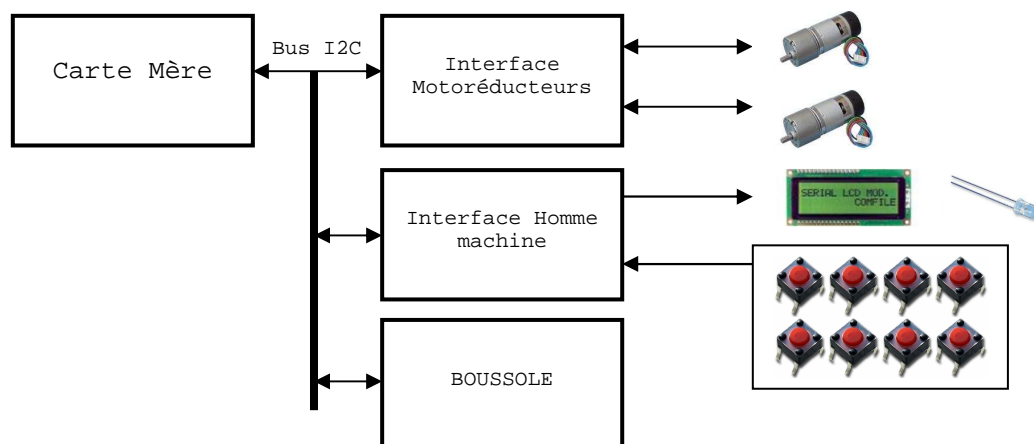
Fiche technique du circuit PCF8574, schémas de la carte SSI et de la carte interface I2C. Cours « Introduction au bus I2C ». Résumé sur le langage C.

Le présent document et la documentation sont téléchargeables sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)



A) Mise en situation

Le « robot solaire » à réaliser se déplace grâce à deux motoréducteurs équipés de codeurs. Ces actionneurs sont commandés par une « **carte interface** » capable de régler leur vitesse, de mesurer leur fréquence de rotation etc... Le nombre d'informations nécessaires au fonctionnement de la carte ajouté à celles qu'elle est capable de fournir justifie l'utilisation d'un **bus de communication série**. Le constructeur de cette carte a choisi le bus I2C.

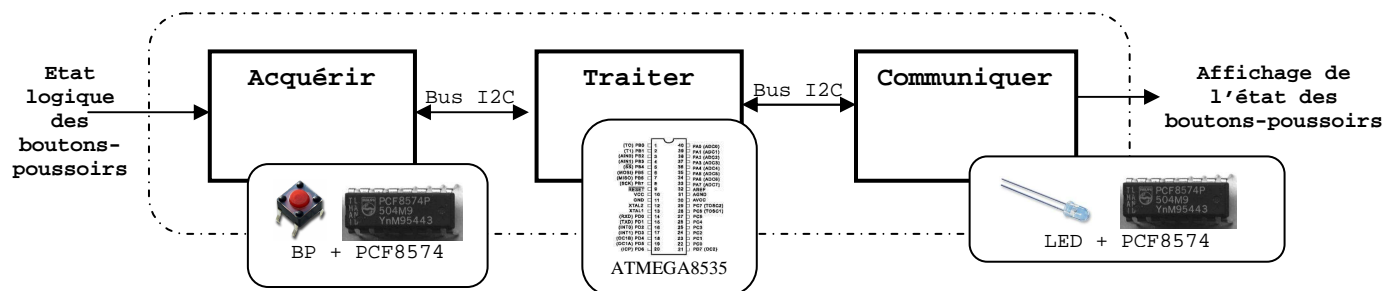
Schéma fonctionnel partiel du robot solaire



D'autres sous-ensembles électroniques du robot tels que la boussole ou l'interface « Homme-machine » fonctionnant avec ce bus, ce TP va vous permettre d'acquérir les connaissances nécessaires à sa mise en œuvre.

	FG1	Mise en œuvre du bus I2C		PPE ROBOT Solaire	1
---	-----	--------------------------	---	-------------------	---

L'étude du bus I2C va permettre de répondre à la **problématique suivante** :
Indiquer par l'éclairement d'une LED qu'une touche du clavier a été correctement actionnée. Pour cela, on propose de mettre en œuvre une structure correspondant à la chaîne d'information donnée ci-dessous.



La fonction « Acquérir » est réalisée par un clavier numérique à huit touches. La fonction « Traiter » est assurée par un programme implanté dans un microcontrôleur (ATMEGA8535). La fonction « communiquer » est remplie par une LED.

L'ensemble des structures matérielles étant réunies sur la carte ATMELSSI et sur la carte interface I2C, votre travail va se limiter à la **réalisation du logiciel** à implanter dans le microcontrôleur.

Pour cela, vous allez **créer et configurer un projet** avec le magicien du cross-compileur **CodeVisionAVR** afin d'obtenir la structure de votre programme. Puis, vous complèterez cette structure avec les fonctions nécessaires à la mise en œuvre du **bus I2C**.

La suite de ce document décrit le travail à réaliser étape par étape. A la fin de cette activité, vous serez capable de commander des circuits électroniques reliés par un bus I2C.

B) Travail demandé

Etape 1 : Création et configuration d'un projet

Lancez le logiciel **CodeVisionAVR**

(1) **Création d'un nouveau projet**

Dans la barre d'outils : « **File** » puis « **New** » pour obtenir la boîte de dialogue ci-contre.

Cochez « **Project** » puis clic sur « **Ok** »

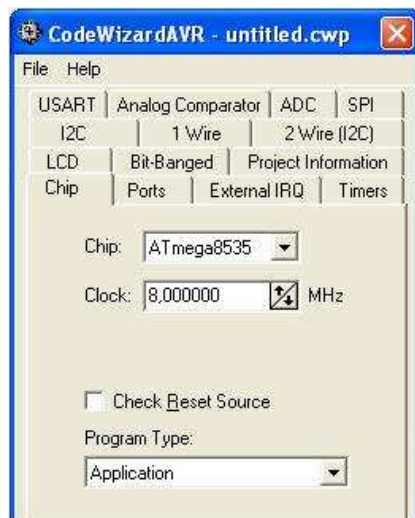


Ici « **Yes** »



(2) Sélection du composant cible

La boîte du « **Magicien** » ci-dessous s'ouvre. Choisissez le « **Chip** » ATmega8535 et réglez le signal d'horloge « **Clock** » à 8Mhz.

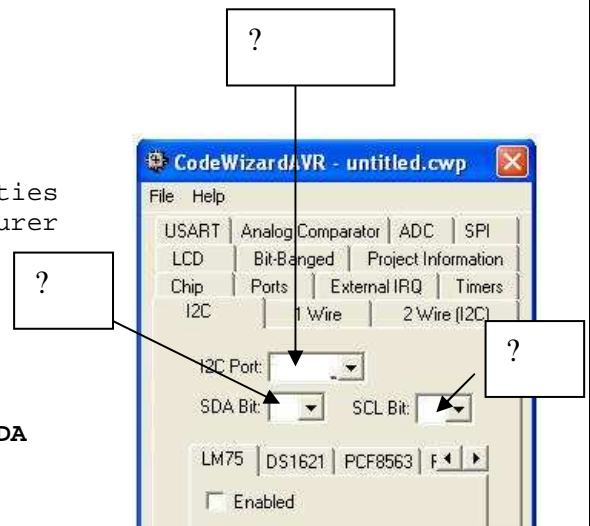


(3) Configuration du bus I2C

sélectionnez l'onglet « **I2C** ».

Pour que le microcontrôleur ATmega8535 communique avec les ports d'entrées sorties à interface I2C PCF8574, il faut configurer le bus I2C. Pour cela, vous devez :

- **Identifier** la position des lignes SCL et SDA sur le microcontrôleur. (voir le schéma de la carte **ATMELSSI V1**)
- **Compléter** les champs « **I2C Port** », « **SDA Bit** » et « **SCL Bit** » dans la boîte de dialogue.



(4) Enregistrement du projet



Sélectionnez « **Program Preview** ».



Si le projet est correctement configuré, le début du fichier source du programme doit correspondre au texte ci-dessous.

```
#include <mega8535.h>

// I2C Bus functions
#asm
    .equ __i2c_port=0x12
    .equ __sda_bit=???    // Vos valeurs
    .equ __scl_bit=???
#endasm
#include <i2c.h>

// I2C Bus initialization
i2c_init();
```

Plus loin

Fermez la fenêtre.

Sélectionnez

→ File

→ Generate, save and Exit

Donnez le nom **TestI2C** à votre projet (3 fois) pour créer les trois fichiers à la base du projet. (.c, .prj, .cwp)

ATTENTION : Le Magicien ne peut plus être utilisé pour modifier votre projet. Voir le prof pour d'éventuelles corrections.

Etape 2 : Acquisition de l'état des « switches » et commande de la LED

Dans ce paragraphe, vous allez compléter la partie déclarative ...

```
void main(void)
{
    // Declare your local variables here
```

... et la partie exécutive du programme.

```
while (1)
{
    // Place your code here

};
```

On rappelle que la partie déclarative d'un programme est la zone dans laquelle sont créées les variables alors que la partie exécutive est la zone de traitement de ces variables.

Malgré la complexité des structures à mettre en œuvre, le programme à réaliser reste relativement simple. Ceci est dû à la « richesse » des bibliothèques de fonctions fournies avec le cross-compileur CodeVisionAVR.

Par exemple, l'initialisation du bus I2C s'effectue avec la fonction :

i2c_init();

Cette fonction est placée automatiquement dans le fichier source C par le « Magicien ». Elle est située dans la bibliothèque I2C.

Le programmeur accède à cette bibliothèque lorsqu'une référence au fichier <I2C.h> est présente dans le fichier source.

L'ensemble des fonctions nécessaires à la mise en œuvre du bus I2C est accessible dans la bibliothèque I2C.

Dans le paragraphe ci-dessous, vous allez identifier le rôle de ces différentes fonctions avant de les utiliser.

Ouvrez la rubrique d'aide comme ci-dessous.

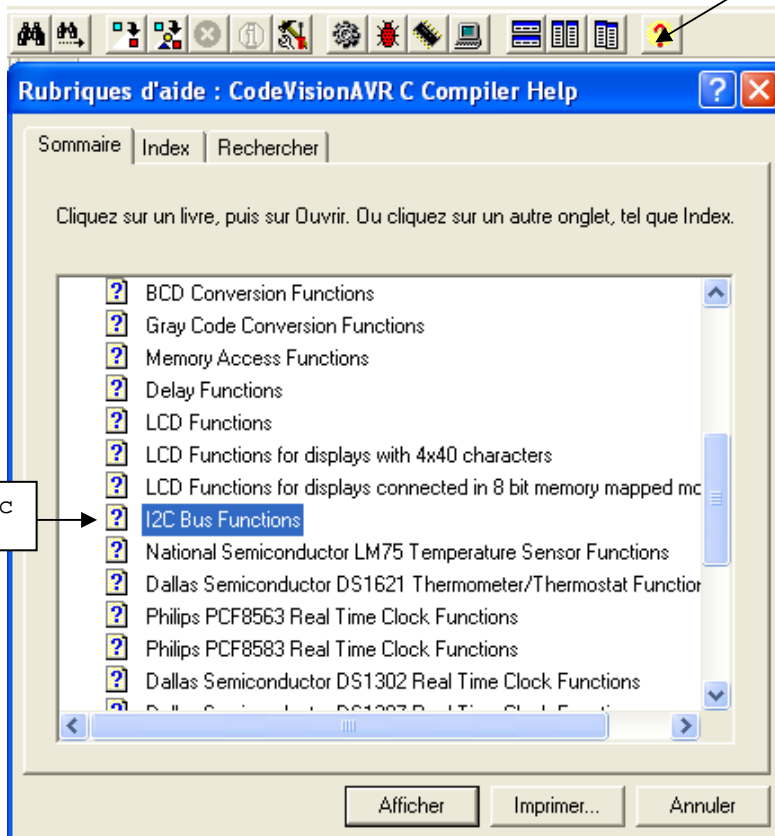
Clic

Sélectionnez
« CodeVisionAvr C compiler
Library Functions »

Sélectionnez « I2C Bus
Functions »

Clic

Clic sur « Afficher »



En langage C, les fonctions comprennent trois parties :

<Le type de la variable renvoyée> <Nom de la fonction> <La liste des paramètres>

Exemple :

void i2c_init(void)

L'aide en ligne commence par le texte ci-dessous :

« The I2C Functions are intended for easy interfacing between C programs and various peripherals using the Philips I2C bus. These functions treat the microcontroller as a bus master and the peripherals as slaves. »

Quel est le rôle (maître ou esclave) du microcontrôleur dans l'application ?

Quel est le rôle (maître ou esclave) des ports d'E/S PCF8574 dans l'application ?

Complétez le document « Introduction au bus I2C » avant de poursuivre.

Quelle fonction de la bibliothèque I2C permet au maître de prendre le contrôle du bus ? (Donnez son nom)

Quelle fonction de la bibliothèque I2C permet au maître de terminer la communication avec l'esclave ? (Donnez son nom)

Quelle fonction de la bibliothèque I2C permet au maître **d'adresser** un esclave (mettre l'adresse de l'esclave sur le bus)? (Donnez son nom)

Quelle fonction de la bibliothèque I2C doit-on utiliser pour envoyer une donnée ? (Donnez son nom)

Quelle fonction de la bibliothèque I2C doit-on utiliser pour lire une donnée ? (Donnez son nom)

(1) Déclaration des bibliothèques de fonctions utilisées dans le programme

La référence à la bibliothèque I2C a été automatiquement placée par le magicien avec la directive `#include <i2c.h>` (voir l'en-tête du programme).

(2) Partie exécutive du programme à réaliser

Le programme à réaliser se limite à la répétition de trois actions :

Lire (Etat logique des boutons-poussoirs)
Traiter l'état des boutons-poussoirs
Ecrire (Etat logique des boutons-poussoirs)

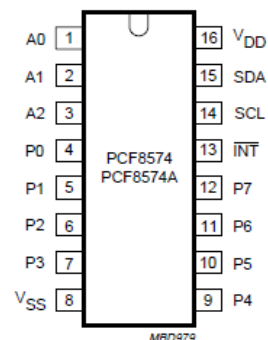
① Les opérations de lecture et d'écriture citées ci-dessus nécessitent la mise en œuvre du bus I2C. Ces opérations sont combinées avec un « **adressage** » du composant.

o **Identification des adresses des composants de la carte interface I2C**

① Sur un bus I2C, l'ensemble des composants (maître et esclave) est relié à la même **voie de communication (constituée des lignes SCL et SDA)**, le maître doit donc utiliser un mécanisme pour identifier l'esclave avec lequel il souhaite dialoguer. Ce mécanisme s'appelle **l'adressage**.

Chaque composant I2C possède une adresse sur 7 bit. Sur le **port d'E/S** PCF8574, quatre bit sont fixes et trois sont configurables par l'intermédiaire de l'état logique placé sur les broches A2, A1 et A0 du composant. On donne en annexe 1 le schéma de la carte interface I2C et en **annexe 2 fig7**, la constitution de l'adresse d'un PCF8574.

Après avoir identifié le type de composant (PCF8574 ou PCF8574A) et la position des cavaliers sur la carte interface, **donnez** l'adresse des circuits U1 (commande des LED) et U2 (relié aux switches) en hexadécimal.



① L'adresse précédente constitue les 7 bits (b1 à b7) du **premier octet** envoyé à l'esclave lors d'un échange avec le maître. Le huitième bit (b0) est utilisé pour préciser si le maître souhaite effectuer une opération de **lecture ou d'écriture**.



o **Lecture de l'état des boutons poussoirs**

On donne en **annexe 2 fig11**, la trame à produire pour effectuer une opération de lecture. Donnez la valeur hexadécimale de l'octet à envoyer pour accéder au circuit U2 en lecture.

Complétez le fichier source C comme ci-dessous et remplacez le ? par la valeur déterminée précédemment.

```
while (1)
{
// ----- Lecture des switchs -----
i2c_start();
i2c_write(?);          // à compléter
BP = i2c_read(0);      // Les switchs sont remplacés par des BP sur la carte
i2c_stop();           // du robot
```

o **Traitement de l'état des boutons-poussoirs**

Le traitement de l'état des boutons-poussoirs peut être réalisé avec une structure alternative (si... alors... sinon). On appelle LED la variable de type octet contenant la valeur à envoyer pour éclairer ou éteindre la LED connectée au bit0 du Ports d'E/S U1.

Sachant qu'une LED s'éclaire lorsqu'on applique 0V à la sortie du PCF8574 auquel elle est connectée, donnez les valeurs de la variable LED lorsqu'on souhaite éteindre ou éclairer la LED connectée au bit0 du Ports d'E/S U1. Les autres LED doivent restées éteintes.

Sachant que lorsqu'un switch est fermé, le PCF8574 voit un « 1 » logique sur sa broche, **écrivez** l'algorithme ou **dessinez** l'algorithme de la partie traitement du programme.

Complétez le fichier source C.

```
while (1)
{
// ----- Lecture des switchs -----

i2c_start();
i2c_write(???); // complété précédemment
BP = i2c_read(0);
i2c_stop();

// ----- Traitement de l'état des switchs -----

// à compléter

} ;
```

o **Ecriture de l'état des boutons poussoirs (éclairer une LED)**

On donne en **annexe 2 fig10**, la trame à produire pour effectuer une opération d'écriture. **Donnez** la valeur hexadécimale de l'octet à envoyer pour accéder au circuit U1 en écriture.

Complétez le fichier source C comme ci-dessous et remplacez ??? par la valeur hexadécimale de l'octet à envoyer pour accéder au circuit U1 en écriture.

```
while (1)
{
// ----- Traitement de l'état des switchs -----

// Complété précédemment

// ----- Affichage de l'état des switchs -----
i2c_start();
i2c_write(???); // à compléter
i2c_write(LED);
i2c_stop();

};
```

Complétez le fichier source C comme ci-dessous.

```
// Initialisation des Ports d'E/S
i2c_start();
i2c_write(0x44);
i2c_write(0x00);
i2c_stop();

i2c_start();
i2c_write(0x42);
i2c_write(0xFF);
i2c_stop();

while (1)
{
```


(3) Partie déclarative du programme à réaliser

Le programme ci-dessus utilise un type de variable :

- BP, LED : entier de type octet et

Pour être reconnues, ces variables doivent être déclarées avant leur utilisation.

Complétez le fichier source C comme ci-dessous :

```
void main(void)
{
// Declare your local variables here
// -----
//type          nom          Commentaires
// -----
unsigned char BP, LED;          // Etat des BP et commande de la LED
```

Appel prof

Pour faire vérifier votre programme.

C) Programmation du composant

Configurez le projet

- > Project
- > Configure
- > Sélectionnez l'onglet "After Make"
- > Cochez "Program the Chip"
- > ok

Programmez le composant



- > Icône "Make the Project"
- > "Program"

Appel prof

Pour faire vérifier le fonctionnement

D) Réalisation de fonctions de lecture et d'écriture pour le bus I2C

La lecture et l'écriture sur le bus I2C nécessitant plusieurs actions, nous allons les rassembler dans des fonctions. Ainsi, notre programme deviendra plus lisible et plus facile à maintenir. Lorsque le nombre de fonctions est important, on les rassemble dans une librairie.

D1) Ecriture des fonctions Lire_Port_ES_I2C et Ecrire_Port_ES_I2C

Une fonction possède un nom, un corps et éventuellement une liste de paramètres.

- Exemple de fonction de lecture des boutons-poussoirs

```
unsigned char Lire_Port_ES_I2C(unsigned char add)    // Nom = Lire_Port_ES_I2C
{
    unsigned char BP ;
    i2c_start();          // add est l'adresse du composant + le bit de lecture
    i2c_write(add);       // add est passé en paramètre à la fonction
    BP = i2c_read(0);     // no acknowledgment
    i2c_stop();
    return BP;            // la fonction retourne l'état des BP
}
```



- Fonction écriture sur le port

Complétez la fonction Ecrire_Port_ES_I2C ci-dessous :

```
Ecrire_Port_ES_I2C (unsigned char add, unsigned char data)
{
    i2c_start();
    _____
    _____
    _____
}
```

D2) Utilisation des fonctions dans le programme

Modifiez votre programme pour qu'il utilise les fonctions ci-dessus. Elles doivent être placées (**déclarées**) avant

```
void main(void)
```

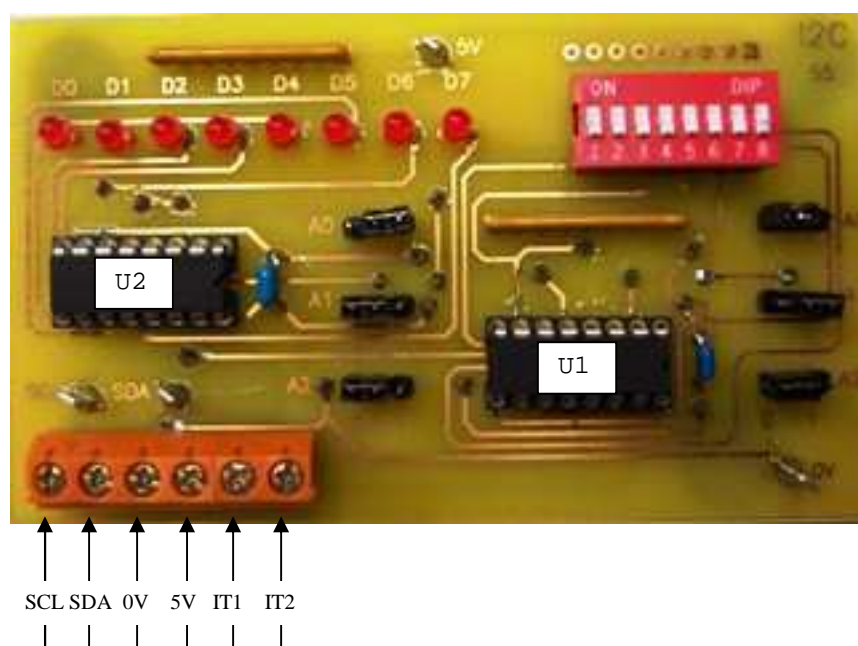
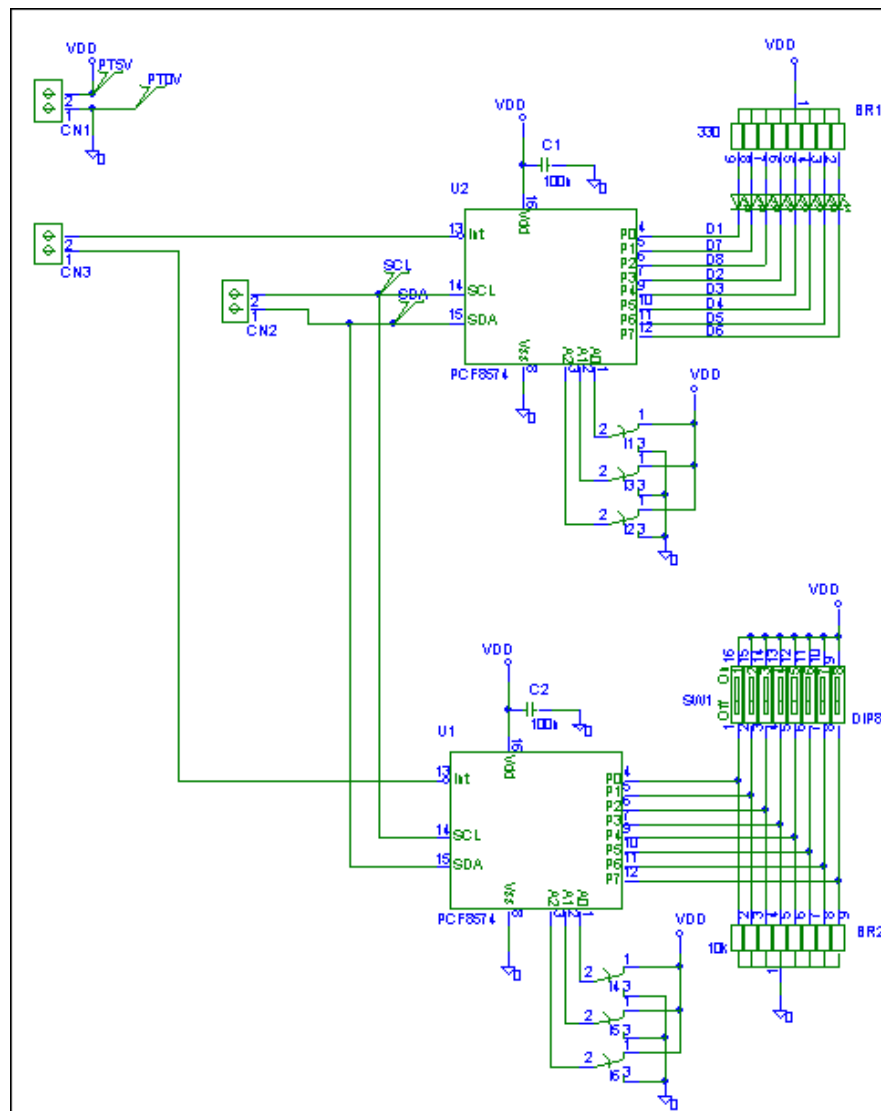
Elles sont utilisées dans le

```
while (1)
```

Appel prof

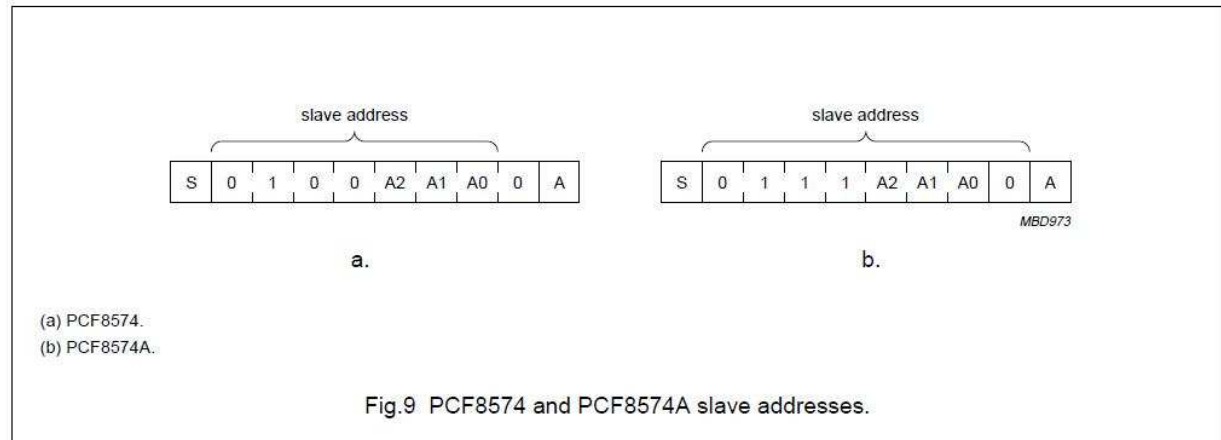
Pour faire vérifier le fonctionnement

Annexe 1 : Schéma de la carte interface I2C



7.1 Addressing

For addressing see Figs 9, 10 and 11.



Each of the PCF8574's eight I/Os can be independently used as an input or output. Input data is transferred from the port to the microcontroller by the READ mode (see Fig.11). Output data is transmitted to the port by the WRITE mode (see Fig.10).

