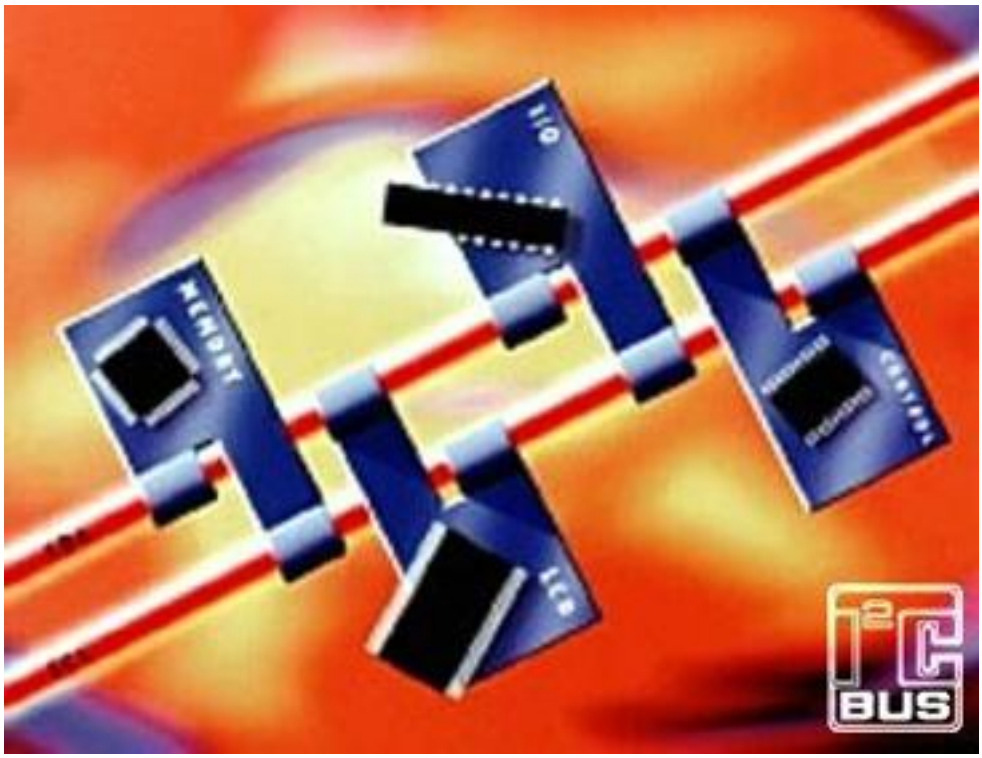
Liaison I2C



Exemple de programme en C

Pour µC de type PIC

Frédéric GIAMARCHI
IUT de Nîmes – Université Montpellier II

Sommaire

Liaison I2C

L'utilisation de la liaison I2C, pour dialoguer entre composants, réduit le nombre de lignes et permet l'échange de données entre plusieurs composants de type maître et esclave. Ce que ne permet pas la liaison série.

Le bus I2C a été créé par Philips. Il est nécessaire de placer deux résistances de 3,3kΩ sur chaque ligne en rappel au +5Volts.

Généralités

L'objet de ce document est de donner des exemples de programmation en C pour la série des PICs. Le langage est le C de CCS. Celui-ci possède des routines I2C pour les PICs ne possédant pas de noyau I2C en interne (PIC16F84) ou exploite les ressources internes de ceux qui en ont (PIC16F87x).

Seul le mode mono-maître sera vu ici. Cela concerne donc le dialogue entre un PIC en maître et plusieurs esclaves dont certains peuvent être des PICs.

Composants I2C

Les composants I2C disponibles possèdent leur propre adresse.

Remarques : Un PIC en esclave

Le mode esclave avec un PIC n'est possible que si celui-ci possède un noyau I2C en interne. Les routines générées par le compilateur CCS ne sont pas opérationnelles dans ce cas.

Lorsqu'un PIC est utilisé en esclave, il est plus simple d'utiliser les interruptions pour le dialogue I2C.

Commandes I2C

#use i2c(master, fast, SCL=PIN_C3, SDA=PIN_C4, Force_HW)

Déclaration de dialogue I2C pour un PIC en maître avec un noyau I2C en interne.

Le mode fast permet une vitesse de 400kHz ; par défaut, il est en mode slow à 100kHz, si le quartz le permet (20MHz)

#use i2c(slave,SCL=PIN_C3,SDA=PIN_C4,address=0xA0) // adresse choisie

Déclaration de dialogue I2C pour un PIC en esclave. Il n'est pas utile de déclarer Force_HW, ni fast.

i2c_start()

Lancement d'un dialogue I2C par un maître, ou changement de sens de dialogue.

i2c _stop()

Arrêt d'un dialogue I2C.

i2c _write(data)
Envoie d'une information vers le bus I2C.

i2c _read() ex : data = i2c _read()
réception d'une information vers le bus I2C.

i2c_poll()
permet de savoir si le maître envoie une donnée ou attend une donnée.
ne concerne que le mode esclave.

Dialogue avec un composant I2C

PCF8574(A) : extension pour 8 entrées-sorties parallèles.

Généralités

Le circuit PCF8574 est un composant CMOS qui permet le pilotage de 8 entrées ou sorties par le bus I2C.

Adressage

0	1	0	0	A2	A1	A0	R/W

PCF8574

0	1	1	1	A2	A1	A0	R/W

PCF8574A

Adresse du composant PCF8574 : 0x40 en écriture et 0x41 en lecture, avec les trois lignes d'adresse A2, A1, et A0 à 0.

Adresse du composant PCF8574A : 0x70 en écriture et 0x71 en lecture, avec les trois lignes d'adresse A2, A1, et A0 à 0.

Programme pour PIC 16F87x

```
#use i2c(master, scl=PIN_C3, sda=PIN_C4)     // valide le dialogue I2C pour un maître

i2c_start();              // départ d'une procédure de dialogue unidirectionnel
i2c_write(0x40);          // adresse du PCF 8574 en écriture
i2c_write(0x88);          // donnée 0x88 à transférer dans le composant

i2c_start();              // nouveau départ pour changement du sens du dialogue
i2c_write(0x41);          // adresse du PCF 8574 en lecture
data = i2c_read(0);       // lit le contenu du composant sans acquittement

i2c_stop();               // fermeture du dialogue I2C
```

Remarques : particularités des lignes du composant PCF8574(A)

Le port d'Entrée/ Sortie est un port quasi-bidirectionnel ne possède qu'un registre d'écriture et de lecture. Lorsque l'on veut utiliser une broche en entrée, on commence par y écrire un « 1 » logique. Ensuite on lit le port complet et on réalise un masque pour isoler la ligne.

En sortie, il est conseillé d'utiliser le principe des sorties en collecteur ouvert.

En entrée, la ligne possède une résistance de rappel au niveau haut.

PCF8591 : convertisseur AN et NA

Généralités

Le circuit PCF8574 est un composant CMOS convertisseur AN et NA possédant 4 entrées analogiques et une sortie analogique configurables par le bus I2C.

Adressage

1	0	0	1	A2	A1	A0	R/W

Adresse du composant PCF8591 : 0x90 en écriture et 0x91 en lecture, avec les trois lignes d'adresse A2, A1, et A0 à 0.

Registre de contrôle

0	X	X	X	0	X	X	X

Le bit 6 valide le convertisseur numérique analogique.

Les bits 4 et 5 sélectionnent la configuration des entrées.

Le bit 2 valide l'auto incrémentation du numéro des entrées analogiques afin que la lecture des entrées se fasse en continu, sans avoir besoin de choisir une entrée.

Les bits 1 et 0 sélectionnent le numéro de l'entrée analogique.

Programme pour PIC 16F87x : utilisation du CNA

```
#use i2c(master, scl=PIN_C3, sda=PIN_C4, Force_HW)

int  i;

main()
{
while(1)
    {       i2c_start();
            i2c_write(0x90);                // adresse du PCF8591 en écriture
            i2c_write(0x40);                // valide le CNA
            for (i=0; i<=255; i++)
                    {       i2c_write(i);           // valeur vers CNA
                            delay_ms(1);
                    }
    i2c_stop();
    }
}
```

Programme pour PIC 16F87x : utilisation du CAN

```
#use i2c(master, scl=PIN_C3, sda=PIN_C4, Force_HW)

int  analog_0, analog_1, analog_2, analog_3, analog_4;

main()
{
while(1)
    {
    i2c_start();

    i2c_write(0x90);                    // adresse du PCF8591 en écriture
    i2c_write(0x44);                    // analogique : option de base et CNA validé
    i2c_write(analog_0);                // copie l'entrée analog_0 sur la sortie analogique

    i2c_start();
    i2c_write(0x91);                    // adresse du PCF8591 en lecture
    analog_0 = i2c_read();              // lecture non valide, renvoie l'adresse
    analog_0 = i2c_read();              // lecture des 4 entrées analogiques
    analog_1 = i2c_read();
    analog_2 = i2c_read();
    analog_3 = i2c_read(0);       // arrêt des conversions par suppression de l'acquittement

    i2c_stop();
    }
}
```

PCF8573: horloge calendrier en temps réel

Généralités

Le composant PCF8573 remplit les fonctions d'horloge en temps réel et calendrier avec un dialogue I2C. Il possède des sorties secondes, minutes et alarme.

Adressage

1	1	0	1	0	A1	A0	R/W

Adresse de la mémoire : 0xD0 en écriture et 0xD1 en lecture, avec les deux lignes d'adresse A1, et A0 à 0.

Programme pour PIC 16F87x : écriture et lecture

```
#use i2c(master,SCL=PIN_C3,SDA=PIN_C4,Force_HW)

int  compt_heu, compt_minut, compt_jour, compt_mois;

main()
{
    i2c_start();
    i2c_write(0xD0);            // adresse du pcf8573 en écriture
    i2c_write(0x00);            // pointeur de mode sur compteur heures
    i2c_write(0x01);            // enregistrement de l'heure
    i2c_write(0x01);            // enregistrement des minutes
    i2c_write(0x01);            // enregistrement du jour
    i2c_write(0x01);            // enregistrement du mois
    i2c_stop();

    while(1)
    {
        i2c_start();
        i2c_write(0xD0);            // adresse du pcf8573 en écriture
        i2c_write(0x00);            // pointeur de mode sur compteur heures
        i2c_start();
        i2c_write(0xD1);            // adresse du pcf8573 en lecture
        compt_heu = i2c_read();     // lecture de l'heure
        compt_minut = i2c_read();   // lecture des minutes
        compt_jour = i2c_read();    // lecture du jour
        compt_mois = i2c_read(0);   // lecture du mois
        i2c_stop();
}   }
```

PCF8583: horloge calendrier avec RAM 2ko

Généralités

Le composant PCF8583 remplit les fonctions d'horloge temps réel, de calendrier et de compteur d'évènement avec un dialogue I2C. Il possède aussi une fonction alarme et 2ko de RAM.

Adressage

1	0	1	0	0	0	A0	R/W

Adresse de la mémoire : 0xA0 en écriture et 0xA1 en lecture, si A0 est à 0.

Programme pour PIC 18F452 : écriture et lecture

```
#use i2c(master,SCL=PIN_C3,SDA=PIN_C4)

int  compt_sec, compt_min, compt_heu;

main()
{
    i2c_start();                // Initialisation du composant
    i2c_write(0xA0);            // adresse du pcf8583 en écriture
    i2c_write(0x00);            // adresse du registre d'état
    i2c_write(0x00);            // configuration du registre d'état
    i2c_stop();

    while(1)
    {
        i2c_start();
        i2c_write(0xA0);            // adresse du pcf8583 en écriture
        i2c_write(0x02);            // pointeur sur le registre des secondes
        i2c_start();
        i2c_write(0xA1);            // adresse du pcf8583 en lecture
        compt_sec = i2c_read();     // lecture des secondes ( format bcd)
        compt_min = i2c_read();     // lecture des minutes ( format bcd)
        compt_heu = i2c_read(0);    // lecture des heures ( format bcd)
        i2c_stop();
}   }
```

DS1621: thermomètre numérique et thermostat

Généralités

Le composant DS1621 remplit les fonctions de thermomètre et thermostat avec un dialogue I2C. Il possède une sortie pour thermostat.

Adressage

1	0	0	1	A2	A1	A0	R/W

Adresse de la mémoire : 0x90 en écriture et 0x91 en lecture, avec les trois lignes d'adresse A2, A1 et A0 à 0.

Programme pour PIC 16F87x : écriture et lecture

```
#use i2c(master,SCL=PIN_C3,SDA=PIN_C4,Force_HW)

i2c_start();              // initialisations
i2c_write(0x90);          // adresse du DS1621 en écriture
i2c_write(0xAC);
i2c_write(0x09);          // conversion en continu
i2c_stop();
i2c_start();
i2c_write(0x90);          // adresse du DS1621 en écriture
i2c_write(0xEE);          // lance les conversions
i2c_stop();

while(1)
    {
    i2c_start();
    i2c_write(0x90);              // adresse du DS1621 en écriture
    i2c_write(0xAA);              // demande de lecture de la température
    i2c_start();
    i2c_write(0x91);              // adresse du DS1621 en lecture
    data_h = i2c_read();          // lecture octet haut
    data_l = i2c_read(0);         // lecture octet bas
    i2c_stop();
    delay_ms(1000);               // temps de conversion
    }
}
```

Remarques

Il est nécessaire de respecter le temps de conversion de 1 seconde.

X2404 : mémoire EEPROM 4k

Généralités (4k = 512 x 8)

Le circuit X2404 est une mémoire EEPROM de 512 octets pour le bus I2C.

Adressage

1	0	1	0	A1	A0	PA	R/W

Adresse de la mémoire : 0xA0 en écriture et 0xA1 en lecture, avec les deux lignes d'adresse A1, et A0 à 0. Les 2 blocs (ou pages) sont adressables par le bit PA.

Programme pour PIC 16F87x : écriture et lecture

```
#use i2c(master, scl=PIN_C3, sda=PIN_C4, Force_HW)

int  buffer[0x10];
main()
{
while(1)
    {
    i2c_start();
    i2c_write(0xA0);                // adresse du X2404 en écriture page 0
    i2c_write(0x00);                // début des adresses mémoire à écrire
    for(i=0;i<=254;i++)
            i2c_write(0xFF);        // donnée à écrire pour effacer la mémoire
    i2c_stop();
    delay_ms(11);                   // dépend du type de mémoire (voir data sheet)

    i2c_start();
    i2c_write(0xA0);                // adresse du X2404 en écriture page 0
    i2c_write(0x00);                // début des adresses mémoire à lire

    i2c_start();
    i2c_write(0xA1);                // adresse du X2404 en lecture page 0
    for(i=0;i<=10;i++)
            buffer[i] = i2c_read(); // les données sont stockées dans une table
    i2c_stop();
    }
}
```

PIC 16F87x en esclave

Programme de lecture d'une donnée unique reçue

```
#use i2c(slave,SCL=PIN_C3,SDA=PIN_C4,address=0xA0)    // adresse choisie

int  data;
//=================================================================
//              Routine d'interruption I2C
//=================================================================
#INT_SSP
void I2C_interrupt()
{
    if (i2c_poll()) ;               // permet de sélectionner la donnée
    data = i2c_read();              // lit la donnée
}
//=================================================================
//              Programme principal
//=================================================================
main()
{
data = 00;
enable_interrupts(INT_SSP);     // Valide les interruptions I2C
enable_interrupts(GLOBAL);

while(1)
    {
    output_B(data);             // Affiche la donnée reçue sur le port B
    }
}
```

Programme de lecture de plusieurs données reçues

```
#use i2c(slave,SCL=PIN_C3,SDA=PIN_C4,address=0xA0)    // adresse choisie
int  data, data0, data1, data2;
enum {Adresse, Data_1, Data_2} i2c_etat;
//=================================================================
//              Routine d'interruption I2C
//=================================================================
#INT_SSP
void I2C_interrupt()
{
data = i2c_read();                          // lit la donnée
    if (i2c_etat == Adresse)                // adresse
            {       data0 = data;
                    i2c_etat = Data_1;
            }
    else if (i2c_etat == Data_1)            // donnée n°1
            {       data1 = data;
                    i2c_etat = Data_2;
            }
    else if (i2c_etat == Data_2)            // donnée n°2
            {       data2 = data;
                    i2c_etat = Adresse;
            }
}
//=================================================================
//              Programme principal
//=================================================================
main()
{
i2c_etat = Adresse;
data0 = 00;
data1 = 00;
data2 = 00;
enable_interrupts(INT_SSP);     // Valide les interruptions I2C
enable_interrupts(GLOBAL);

while(1)
    {
    output_B(data1);            // Affiche la donnée n°1 sur le port B
    delay_ms(1000);             // tempo de 1 seconde
    output_B(data2);            // Affiche la donnée n°2 sur le port B
    delay_ms(1000);             // tempo de 1 seconde
    }
}
```

Programme de lecture et d'envoi de données

```
#use i2c(slave,SCL=PIN_C3,SDA=PIN_C4,address=0xA0)    // adresse choisie
int         data, data0, data1, data2;
enum {Adresse, Data_1, Data_2} i2c_etat;
//=============================================================
//          Routine d'interruption I2C
//=============================================================
#INT_SSP
void I2C_interrupt()
{
if (!i2c_poll())            // pas de données reçues donc le maître attend une donnée
    {       i2c_write(input(pin_B0));           // envoi vers le maître
            I2C_ETAT = Adresse;         }
else
    {
    data = i2c_read(0);                         // lit la donnée
    if (i2c_etat == Adresse)                    // adresse
            {       data0 = data;
                    i2c_etat = Data_1;      }
    else if (i2c_etat == Data_1)                // donnée n°1
            {       data1 = data;
                    i2c_etat = Data_2;      }
    else if (i2c_etat == Data_2)                // donnée n°2
            {       data2 = data;
                    i2c_etat = Adresse;     }
    }
}
//=============================================================
//          Programme principal
//=============================================================
main()
{
i2c_etat = Adresse;
data0 = 00;         data1 = 00;     data2 = 00;
enable_interrupts(INT_SSP);     // Valide les interruptions I2C
enable_interrupts(GLOBAL);

while(1)
    {       output_B(data1);            // Affiche la donnée n°1 sur le port B
            delay_ms(1000);             // tempo de 1 seconde
            output_B(data2);            // Affiche la donnée n°2 sur le port B
            delay_ms(1000);     }       // tempo de 1 seconde
}
```

PIC 16F87x en maître vers esclaves PICs

Programme de lecture et d'écriture de données vers des esclaves PICs

```
#use i2c(master,SCL=PIN_C3,SDA=PIN_C4,Force_HW)
int  data;

main()
Delay_ms(2000);          // tempo pour synchroniser par le maître
{
while(1)
    {
    i2c_start();
    i2c_write(0xA1);     // adresse d'un esclave F873 en lecture
    data = i2c_read();   // lecture de la donnée envoyée

    if (data == 0x01)
            {
            i2c_start();
            i2c_write(0xA0);     // adresse d'un esclave F873 en écriture
            i2c_write(0xFF);     // envoi la donnée n°1 vers l'esclave
            i2c_write(0x00);     // envoi la donnée n°2 vers l'esclave
            }
    else
            {
            i2c_start();
            i2c_write(0xA0);     // adresse d'un esclave F873 en écriture
            i2c_write(0xAA);     // envoi la donnée n°1 vers l'esclave
            i2c_write(0x55);     // envoi la donnée n°2 vers l'esclave
            }
    i2c_stop();
    }
}
```

Bibliographie

Le bus I2C : communication inter-IC	Elektor n° xxx (janvier 1991)
Interface I2C encartable pour PC	Elektor n° 163 (janvier 1992)
Convertisseur A/N-N/A et E/S pour I2C	Elektor n° 164 (février 1992)
Module à afficheurs 7 segments à LED	Elektor n° 165 (mars 1992)
I2C power switch	Elektor n° 187 (janvier 1994)
Affichage alphanumérique I2C	Elektor n° 188 (février 1994)
Prolongateur de bus I2C	Elektor n° 192 (juin 1994)
Interface parallèle I2C	Elektor n° 210 (décembre 1995)

Annexes

PCF 8573 : horloge/calendrier en temps réel avec alarme
PCF 8574 : extension pour 8 entrées-sorties parallèles.
PCF 8583 : horloge/calendrier avec RAM 2ko
PCF 8591 : convertisseur AN et NA
X24C04 : mémoire EEPROM 4ko
DS 1621 : thermomètre numérique et thermostat

Philips
Semiconductors

PHILIPS

PCF 8573

5 BLOCK DIAGRAM

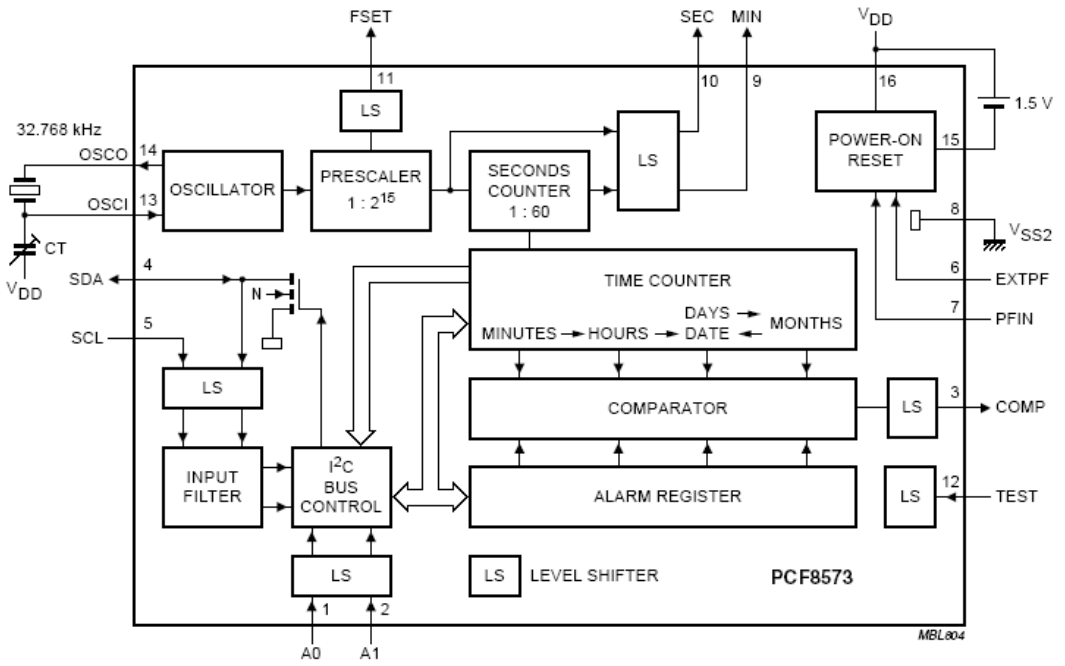


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
A0	1	address input
A1	2	address input
COMP	3	comparator output
SDA	4	serial data line; I²C-bus
SCL	5	serial clock line; I²C-bus
EXTPF	6	enable power fail flag input
PFIN	7	power fail flag input
V_{SS2}	8	negative supply 2 (I²C interface)
MIN	9	one pulse per minute output
SEC	10	one pulse per second output
FSET	11	oscillator tuning output
TEST	12	test input; connect to V_{SS2} if not in use
OSCI	13	oscillator input
OSCO	14	oscillator input/output
V_{SS1}	15	negative supply 1 (clock)
V_{DD}	16	common positive supply

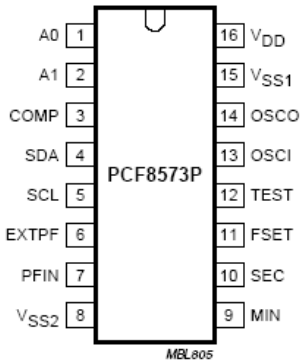


Fig.2 Pinning diagram (DIP16).

PCF 8574

4 BLOCK DIAGRAM

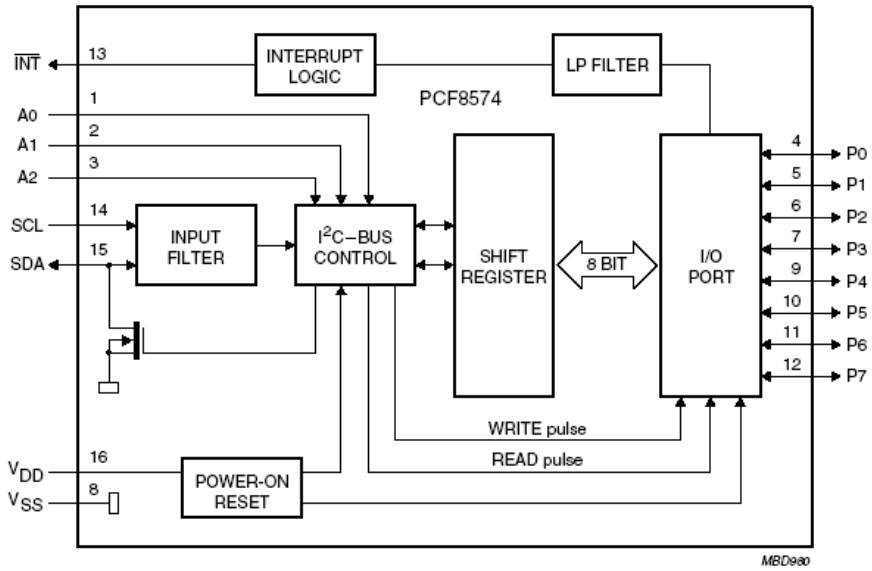


Fig.1 Block diagram (pin numbers apply to DIP16 and SO16 packages).

PCF 8583

5 BLOCK DIAGRAM

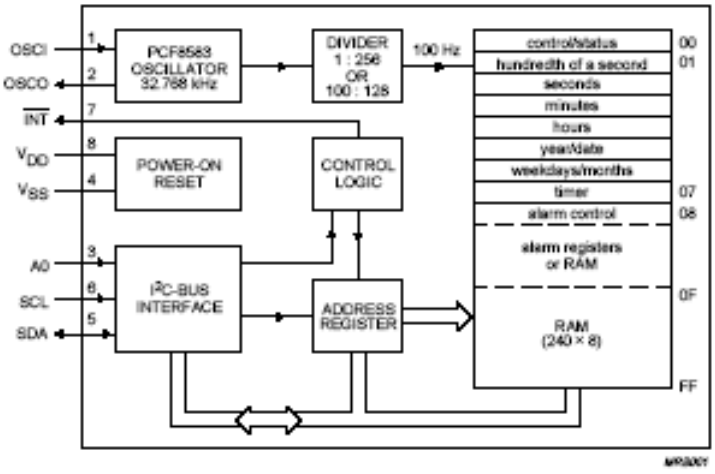


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

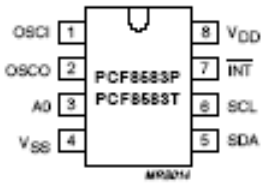


Fig.2 Pinning diagram.

PCF 8591

5 BLOCK DIAGRAM

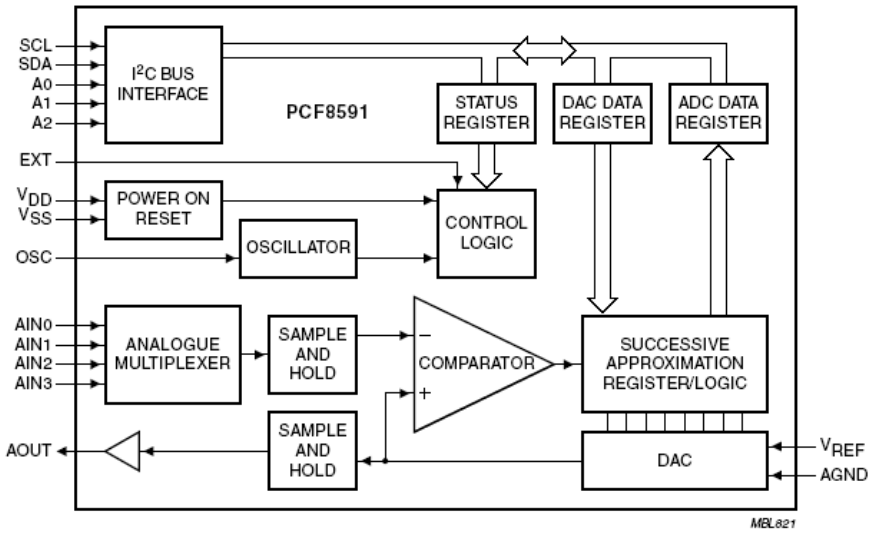


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I²C-bus data input/output
SCL	10	I²C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

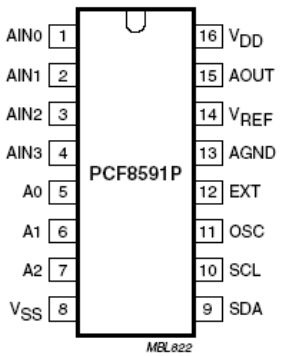


Fig.2 Pinning diagram (DIP16).

DS 1621

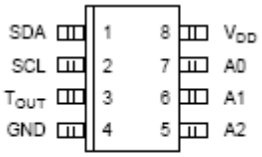
DALLAS SEMICONDUCTOR

DS1621
Digital Thermometer and Thermostat

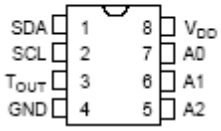
FEATURES

- Temperature measurements require no external components
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to 257°F in 0.9°F increments
- Temperature is read as a 9–bit value (two byte transfer)
- Wide power supply range (2.7V to 5.5V)
- Converts temperature to digital word in 1 second
- Thermostatic settings are user definable and nonvolatile
- Data is read from/written via a 2–wire serial interface (open drain I/O lines)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermal sensitive system.
- 8–pin DIP or SOIC package (150 MIL and 208 MIL)

PIN ASSIGNMENT



DS1621S 8–PIN SOIC (150 MIL)
DS1621V 8–PIN SOIC (208 MIL)
See Mech. Drawings Section



DS1621
8–PIN DIP (300 MIL)
See Mech. Drawings Section

PIN DESCRIPTION

SDA	–	2–Wire Serial Data Input/Output
SCL	–	2–Wire Serial Clock
GND	–	Ground
T_{OUT}	–	Thermostat Output Signal
A0	–	Chip Address Input
A1	–	Chip Address Input
A2	–	Chip Address Input
V_{DD}	–	Power Supply Voltage

DESCRIPTION

The DS1621 digital thermometer and thermostat provides 9–bit temperature readings which indicate the temperature of the device. The thermal alarm output, T_{OUT}, is active when the temperature of the device exceeds a user–defined temperature TH. The output remains active until the temperature drops below user defined temperature TL, allowing for any hysteresis necessary.

User defined temperature settings are stored in non–volatile memory, so parts may be programmed prior to insertion in a system. Temperature settings, and temperature readings are all communicated to/from the DS1621 over a simple 2–wire serial interface.

X24C04

4K **X24C04** **512 x 8 Bit**

Serial E²PROM

FEATURES

- 2.7V to 5.5V Power Supply
- Low Power CMOS
  - —Active Read Current Less Than 1 mA
  - —Active Write Current Less Than 3 mA
  - —Standby Current Less Than 50 µA
- Internally Organized 512 x 8
- 2 Wire Serial Interface
  - —Bidirectional Data Transfer Protocol
- Sixteen Byte Page Write Mode
  - —Minimizes Total Write Time Per Byte
- Self Timed Write Cycle
  - —Typical Write Cycle Time of 5 ms
- High Reliability
  - —Endurance: 100,000 Cycles
  - —Data Retention: 100 Years
- 8 Pin Mini-DIP, 8 Pin SOIC and 14 Pin SOIC Packages

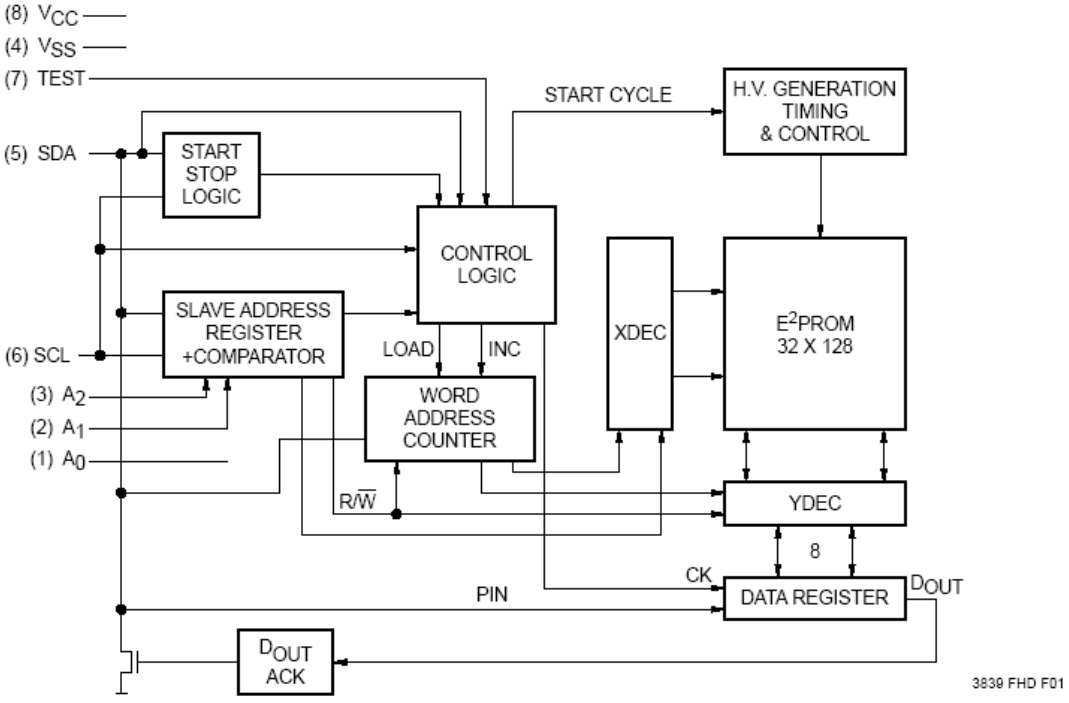
DESCRIPTION

The X24C04 is a CMOS 4096 bit serial E²PROM, internally organized 512 x 8. The X24C04 features a serial interface and software protocol allowing operation on a simple two wire bus.

The X24C04 is fabricated with Xicor's advanced CMOS Textured Poly Floating Gate Technology.

The X24C04 utilizes Xicor's proprietary DirectWrite™ cell providing a minimum endurance of 100,000 cycles and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



DirectWrite™ is a trademark of Xicor, Inc.