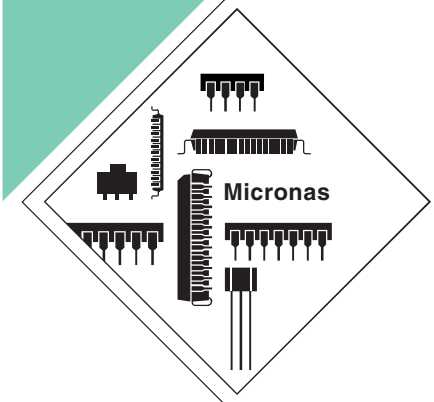


APPLICATION NOTE IC

CDC 32xxG
I^2C Port Expander Interfacing

Edition Feb. 6, 2003
6251-593-4-1AN

MICRONAS

Contents

I²C Port Expander Interfacing

1. Introduction

This document describes a basic application for the I²C master module of the Car Dashboard Controller family CDC 32xxG with ARM7TDMI core by means of connecting an I²C I/O expander IC (PCF8574A).

1.1. Device Overview

This application note can be used together with the devices listed below.

Version	Type	Package
CDC 32x5G-Bx	Emulator Chip	CPGA257
CDC 32x7G-Bx	MCM with 512k Flash	PQFP128
CDC 32x5G-Cx	Emulator Chip	CPGA257
CDC 32x7G-Cx	MCM with 512k Flash	PQFP128

Note: Micronas provides appropriate system files for each microcontroller version. This includes the start-up code, the header for register definitions, and the header for the HW option definitions.

2. I²C Bus

2.1. Overview

The I²C bus has been developed in the early 80's by Philips Semiconductors to provide an easy way to connect a CPU to peripheral ICs in a TV set. I²C is an acronym for Inter-IC bus. Its name literally explains the purpose: to provide a communication link between integrated circuits.

Today, the I²C bus is used in a wide field of applications, even in the automotive field. This document describes the use of one of the CDC 32xxG's I²C bus master together with two port expander ICs from Philips (PCF8574A). Such an application can be used, for instance, to save I/O pins of the microcontroller. In the example application one expander IC drives eight LEDs, the other is used as an 8-bit input with switches.

2.2. General I²C Features

The CDC 32xxG's I²C master interface comprises the following features:

– 2-wire, bidirectional bus

– Serial, synchronous data transfer

– 8-bit data blocks

– Up to 1.25 Mbit/s data transfer rate

It is possible to connect a certain number of same or different ICs to the I²C bus. Each IC has a separate, adjustable address.

The devices are connected in parallel to the I²C bus. In particular, all SDA respectively SCL pins are WIRED-OR connected. The resistors R_p pull up the lines SCL and SDA to $+V_{DD}$, which means a logical one (1). This condition remains until a device (e.g. the port expander) pushes SDA to GND, which means a logical zero (0). For the line SDA it makes no difference wether the device 1 or device 2 pushes SDA to GND. Thus, we are speaking about a WIRED-OR connection.

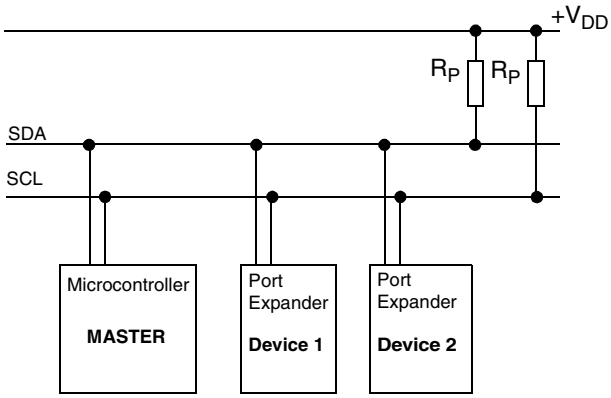


Fig. 2–1: Example for an I²C bus configuration

Note: The CDC 32xxG features two I²C master interface units. The modules are only operable as pure master. Multimaster buses are not possible.

2.3. Communication Protocol

Concerning communication between two partners two different roles have to be considered:

The **Master** provides the clock signal and starts the transfer of data and stops it.

The **Slave** becomes activated when he detects its address on the bus.

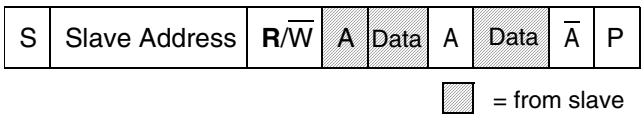
The roles of the devices are partly fixed by the hardware and can't be changed. The CDC 32xxG's I^2C module is a pure master module and can't be used as a slave. In this case, the port expander works as the slave.

The protocol consists of a start condition, slave address, n Bytes of data and a stop condition. Each Byte is followed by an acknowledge bit. A start condition is defined as a high-to-low transition on the SDA line while SCL is high. A stop condition is defined as a low-to-high transition on SDA while SCL is high. The addressed slave responds with an acknowledge bit to each Byte sent from the master. The acknowledge bit is a low logic level during the ninth clock period.

2.4. I^2C Read and Write

The software has to follow a determined sequence for read, respectively write operation, on the bus. The principle parts of a read and a write sequence are shown below:

Read Sequence for two data Bytes:



Write Sequence for two data Bytes:

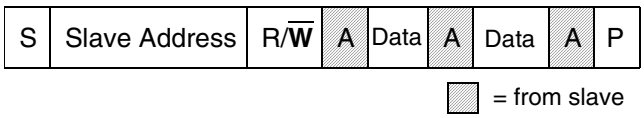


Fig. 2–2: Read/write sequence

S	Start condition from master
Slave Address	DeviceID + Address
R/$\overline{W}$	In case of read = 1, in case of write = 0
A	Acknowledge
Data	Data Byte to send/receive
$\overline{A}$	Not Acknowledge
P	Stop condition from the master

Note: The number of Bytes to be transferred is arbitrary. The read/write functions in the example software for this Application Note uses only one Byte for receiving/transmitting.

2.5. The Port Expander PCF8574(A)

The I/O expander IC from Philips provides an 8-bit wide, digital and bidirectional I/O-port. On an I^2C bus 16 of such port expander devices can be connected (8×PCF8574 and 8×PCF8574A). The difference between both versions are the slave address field bits 4 to bit 7 (see below for PCF8574A).

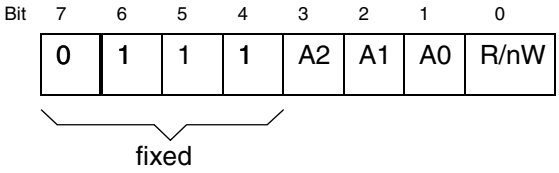


Fig. 2–3: Slave address of PCF8574A

The transmission of data to an I/O-Port Expander consists of:

START,slave-address, db1,db2,...,STOP

The receipt of the slave address and each data Byte is confirmed with an acknowledge.

To read data from the I/O-Port Expander the following sequence has to be followed:

START, slave-address, read db1, read db2,..., STOP

The slave gives now a receipt of the received address, afterwards the µC acknowledges each received data Byte. If the controller don't wants to receive anymore, it signals a not-acknowledge after the last Byte and sends a STOP.

Please refer to the Philips data sheet of the PCF8574A for more details.

3. Programming the I²C Bus Master

This chapter describes how to configure and initialize the I²C master module and how to implement some basic I²C read/write routines. The software uses the Micronas system files (startup, headers). The driver functions for the I²C bus can be found in file “i2c.c”. Hardware-dependent constants and I²C function prototypes are located in ”i2c.h”.

Note: The software example uses only I²C module 0 of the microcontroller. Before using the described I²C functions consider the initializations in this chapter!

3.1. Hardware Options

For the I²C master no hardware options have to be set.

3.2. Port Configuration

Since the I²C bus lines are wired-or connected (with external pull-up resistor), the used U-ports have to be configured to double-pull-down mode. In addition, the ports have to be set to slow-mode where a current limit is set and the output may even be shorted to either supply.

The port slow mode flag has to be set in the standby register SR0.

Example:

```
SR0_PSLW = 1;// set port slow mode
```

The I²C Modules use the U-port 2 [0:1] and U-port 5 [1:2] for SCL respectively SDA. The configuration is done as follows (for U2.1 and U2.0, I²C-Module 0):

Example:

```
U2NS |= 0x03; // U2.0 and U2.1 = special out
U2TRI &= 0xFC; // enable the output driver
U2SLOW |=  0x03; // set port slow mode
U2DPM  |=  0x03; // output driver pull down
```

3.3. Module Initialization

Additional to the port configuration, the following registers have to be initialized for proper function:

Port multiplexer: Bit U20 has to be set to 1 to be able to use SCL0 and SDA0. As default, after reset this bit is already set.

Example:

```
srPM_U20 = 1; // SCL0 at U2.0, SDA0 at U2.1
PM = srPM;
```

I²C mode register: The bit rate is set in the I²C mode register. Please consider the optimum bit rate depending to your application. Using the deglitcher, the maximum bit rate is limited (see CDC 32xxG specification). In the example, the resulting bit rate is 100 kBit/s.

Example:

```
#define I2C_SPEED 0x0a
.....
I2CM0 = 0x80 | I2C0_SPEED;
```

Standby register: The I²C module must be enabled in the standby register SR0. Below it is shown for the module 0.

Example:

```
SR0_I2C0 = 1; // enable the I2C master 0
```

The flow charts in Section 3.5. show the principal way to program a read, respectively write function using the appropriate I²C register of module 0. The same procedure can be used together with the module 1. For register names and more detailed description of the modules, please refer to the microcontroller specification.

3.4. Global Device Structure “I2C_Device”

The structure I2C_Device contains the device information for the slave:

```
struct I2C_Device
{
unsigned char DeviceID; // Device Ident
unsigned char Address; // Slave Address
};
```

For the software example two variables, “Port_Expander_1” and “Port_Expander_2”, are derived from the structure I2C_Device in main.c:

```
struct I2C_Device Port_Expander_1;
struct I2C_Device Port_Expander_2;
```

The initialization of the I²C device structure is done by the function “I2C_Device_Structure_init()”:

```
I2C_Device_Structure_init
(&Port_Expander_1,PCF8574A,0x01);
```

The parameters are a pointer to the appropriate structure (derived from I2C_Device), the device ID and the used device address. The address has to be set regarding to the hard-wired address (A0....A2) of the port expander chip.

3.5. Read from I²C

The function for reading from the I²C bus is based on the program flow chart in Fig. 3–1. The function is declared external and can be used e.g. from “main.c”. The parameter for the read function is a pointer to the declared I2C_Device structure:

```
extern unsigned char I2C_Device_read ( struct
I2C_Device* ptr );
```

The return value is a unsigned char and is the Byte read out from the I²C read FIFO.

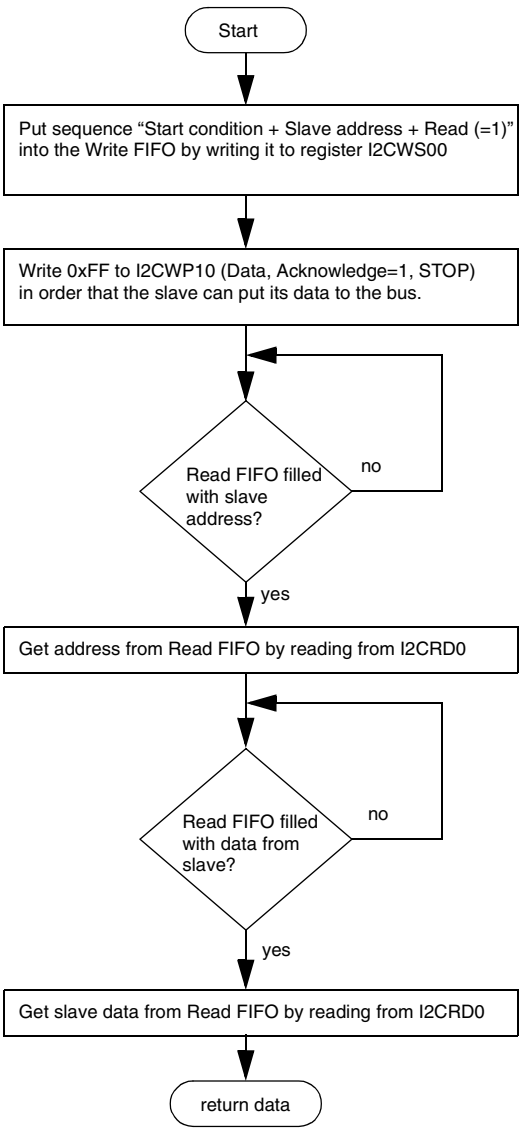


Fig. 3–1: Flow chart for read sequence

3.6. Write to I²C

The function to write to the I²C bus is based on the program flow chart in Fig. 3–2. The parameters for the write function are the pointer to the I²C device structure and the data to write to the I²C bus. The function has no return value:

```
extern void I2C_Device_write ( struct I2C_Device
*ptr,unsigned char Data );
```

The full implementation of both write and read function is given in the code listing in Section 5..

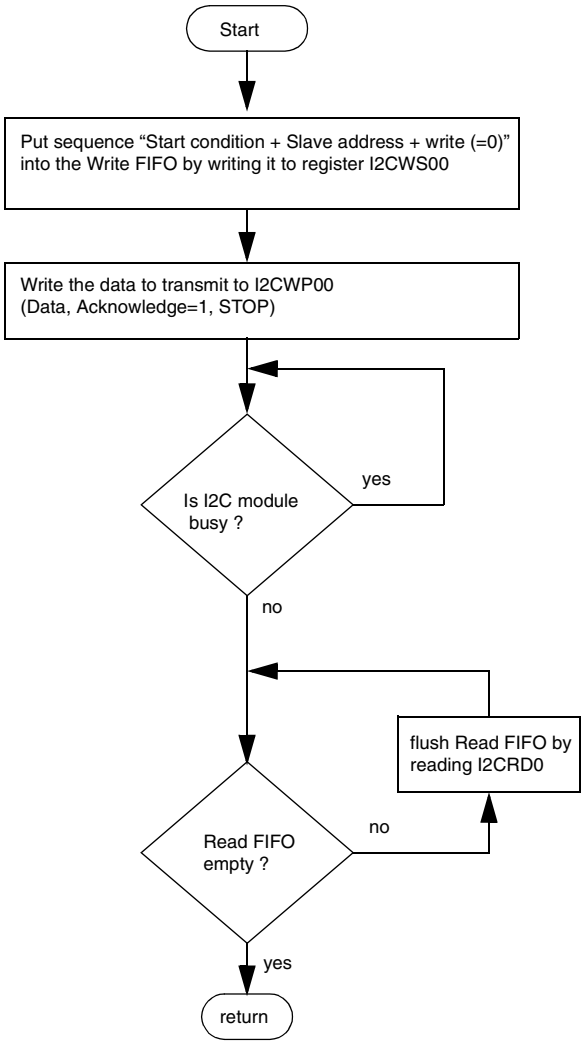


Fig. 3–2: Flow chart for write sequence

4. Application Hardware with PCF8574A

This chapter demonstrates how to connect two port expander chips PCF8574A to the I^2C bus.

4.1. Application Circuit

The hardware used for this software example is shown in the figure below:

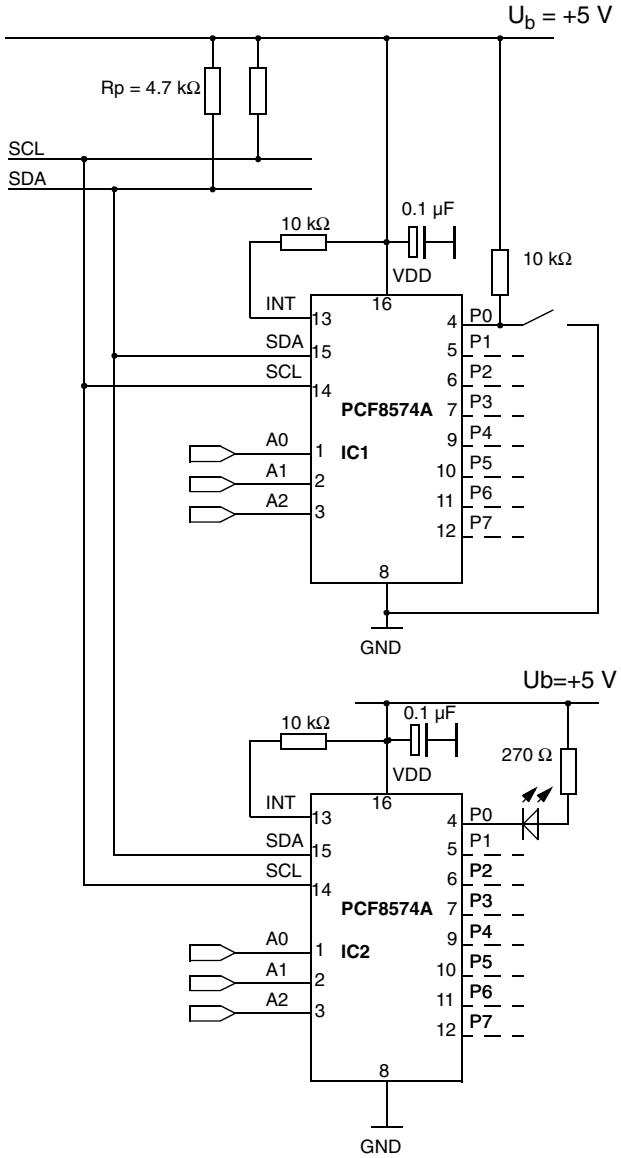


Fig. 4–1: Application circuit

The I^2C SCL and I^2C SDA pins of the PCF8574A are connected to the open-drain configurated I/O pins of the microcontroller. The address selection pins A0...A2 have to be connected either to GND or to $+U_B$, depending on the used address. For this example, the following addressing is used:

IC 2: Address = 0x01

A0 = $+U_B$, A1 = A2 = GND

IC 1: Address = 0x00

A0 = A1 = A2 = GND

All the switches on P0...P7 can be connected as shown for P0. The same can be done for the LEDs on IC2 from P0...P7.

It is possible to drive a transistor with the port pins in order to enable the connection of devices with higher current consumption (e.g. a relay).

Note: The external pull-up termination resistors R_p are required on the I^2C bus. The value depends on the capacitive load on the bus and is 4.7 kΩ for this example.

5. Source Code

The code listing does not contain the source for the microcontroller specific start-up files and header files.

5.1. C-entry (main.c)

```
/*
*******************************************************************************
** Copyright(C) Micronas GmbH
** This program is the property of Micronas GmbH. Its contents is proprietary
** information and is not to be reproduced in whole or part by any means or is
** to be disclosed to anyone except employees of Micronas GmbH, or as agreed in
** writing signed by an authorized person of Micronas GmbH.
** The software is delivered "AS IS" without warranty or condition of any
** kind, either express, implied or statutory. This includes without
** limitation any warranty or condition with respect to merchantability or
** fitness for any particular purpose, or against the infringements of
** intellectual property rights of others.
** ----------------------------------------------------------------------------
** file:                main.c
**
** description:Main program I2C demo
** ----------------------------------------------------------------------------
** author:   Adriano De Rosa, August 12, 2002
** modified: Adriano De Rosa, August 12, 2002
**
** Version:        V0.01
*******************************************************************************
*/

#define SRAM_C

#include "cdc3205g-c2.h"
#include "cdc3205g-c2-hwo.h"
#include "i2c.h"

/*=============================================================================*/
/* definitions                                                                 */
/*=============================================================================*/

/*=============================================================================*/
/* external variables and constants                                            */
/*=============================================================================*/

struct I2C_Device Port_Expander_1; // Struct for Port Expander Device 1
struct I2C_Device Port_Expander_2; // Struct for Port Expander Device 2

/*=============================================================================*/
/* external functions                                                          */
/*=============================================================================*/

/*****************************************************************************/
void main( void )
/*****************************************************************************/
{
      unsigned char i;

      //-------------------------------------------------
      //    setup all used ports
      //-------------------------------------------------
      srPM_U20 = 1;      // Port Multiplexer => SCL0 at U2.0, SDA0 at U2.1
      PM = srPM;

      //-------------------------------------------------
      // Initialize I2C Module 0 and the used port U2 [0:1]
      //-------------------------------------------------
      U2NS   |=  0x03; // set U2.1 (SDA0) and U2.0 (SCL0) to special out
      U2TRI  &=  0xFC; // Enable Output Driver U2[0:1]
      U2SLOW |=  0x03; // Set U2[0:1]to slow mode
      U2DPM  |=  0x03; // Enable double pull-down mode U2[0:1]

      SR0_I2C0 = SR0_I2C1 = 1;// enable I2C
      SR0_PSLW = 1;                   // set port slow mode
      I2CM0 = 0x80 | I2C0_SPEED; // set bitrate I2C clock prescaler

      //-------------------------------------------------
      // Initialize I2C-Device Structures
      //-------------------------------------------------
      I2C_Device_Structure_init (&Port_Expander_1, PCF8574A, 0x01 ); // Port Exp.1, Addr. 0x01
      I2C_Device_Structure_init (&Port_Expander_2, PCF8574A, 0x00 ); // Port Exp.2, Addr. 0x00

      while(1)
       {
                       i = I2C_Device_read  ( &Port_Expander_2 ); // Read Byte   from Input Switch
                        I2C_Device_write ( &Port_Expander_1, i );  // Write back   to the LEDs
       }
}
/*****************************************************************************/

/* End of main.c */
```

5.2. I²C Functions (i2c.c)

```
/*
*****************************************************************************
** Copyright(C) Micronas GmbH
** This program is the property of Micronas GmbH. Its contents is proprietary
** information and is not to be reproduced in whole or part by any means or is
** to be disclosed to anyone except employees of Micronas GmbH, or as agreed in
** writing signed by an authorized person of Micronas GmbH.
** The software is delivered "AS IS" without warranty or condition of any
** kind, either express, implied or statutory. This includes without
** limitation any warranty or condition with respect to merchantability or
** fitness for any particular purpose, or against the infringements of
** intellectual property rights of others.
** ---------------------------------------------------------------------------
**
** file: i2c.c
**
** description: Driver functions for I2C I/O expander PCF8574A
**
**
** ---------------------------------------------------------------------------
** author: Adriano De Rosa, August 08, 2002
**
** modified: Adriano De Rosa, August 08, 2002
**
**
** Version:      V0.01
*****************************************************************************
*/

#include "cdc3205g-c2.h"
#include "cdc3205g-c2-hwo.h"
#include "i2c.h"

/*=========================================================================*/
/* external functions                                                      */
/*=========================================================================*/

/*****************************************************************************/
void I2C_Device_Structure_init  ( struct I2C_Device* ptr,unsigned char DeviceID_par,
                                  unsigned char Address_par)
/*****************************************************************************/
{
       ptr -> DeviceID = DeviceID_par; // Init Device ID
       ptr -> Address = Address_par;   // Init the Device Address
};

/*************************************************************************/
void I2C_Device_write ( struct I2C_Device* ptr,
                        unsigned char Data )
/*************************************************************************/
{
      unsigned char buffer;

      I2CWS00 = (ptr -> DeviceID) | ((ptr -> Address) *2 );// start, DEV SEL, Port Select, ACK=1
      I2CWP00 = Data;                                      // data, ACK=1, stop

      while( I2CRS0_BUSY );   // Wait for I2C data send

      while( !I2CRS0_RFE )
                  buffer = I2CRD0; // Flush the read FIFO
};
```

```
/*********************************************************************/
unsigned char I2C_Device_read  ( struct I2C_Device* ptr )
/*********************************************************************/
{
      unsigned char buffer;

      I2CWS00 = ptr->DeviceID | (ptr->Address * 2 + 1);// start, DEV SEL, read=1, ACK=1

      I2CWP10 = 0xff;       // Write dummy

      while( I2CRS0_RFE );  // Check for empty read FIFO

      buffer = I2CRD0;                        // Read device address

      while( I2CRS0_RFE );  // Check for data in the read FIFO

      buffer = I2CRD0;                        // Get the data from out of the read-FIFO

      return (buffer);       // Return I2C data

};

/*********************************************************************/

/* End of i2c.c */
```

5.3. Header File for I²C Functions (i2c.h)

```
/*
*********************************************************************
** Copyright(C) Micronas GmbH
** This program is the property of Micronas GmbH. Its contents is proprietary
** information and is not to be reproduced in whole or part by any means or is
** to be disclosed to anyone except employees of Micronas GmbH, or as agreed in
** writing signed by an authorized person of Micronas GmbH.
** The software is delivered "AS IS" without warranty or condition of any
** kind, either express, implied or statutory. This includes without
** limitation any warranty or condition with respect to merchantability or
** fitness for any particular purpose, or against the infringements of
** intellectual property rights of others.
** ------------------------------------------------------------------------
**
** file:i2c.h
**
** description: Header with definitions for I2C.c
**
**
** ------------------------------------------------------------------------
** author: Adriano De Rosa, May 24, 2002
**
** modified:Adriano De Rosa, May 24, 2002
**
**
** Version:      V0.01
*********************************************************************
*/

/*====================================================================*/
/* definitions                                                        */
/*====================================================================*/

#define PCF8574A      0x70  // Device ID from Port Expander Chip
#define I2C0_SPEED    10     // set bitrate to 100 kbit/s @ f1 = 4 MHz

/*====================================================================*/
/* macros                                                             */
/*====================================================================*/

/*====================================================================*/
/* external variables and constants                                   */
/*====================================================================*/

// DEVICE STRUCTURE
struct I2C_Device
{
      unsigned char DeviceID;     // Device Ident
   unsigned char Address;       // Slave Address (Offset)
};
```

```
/*====================================================================*/
/* prototypes of external functions                                   */
/*====================================================================*/

/**********************************************************************/
extern void I2C_Device_Structure_init   ( struct I2C_Device* ptr,
            unsigned char DeviceID_par, unsigned char Address_par);
/**********************************************************************
Description:  Initialization of the slave device structure

Parameters:    - DeviceID of the corresponding I2C device
              - Used Address of I2C device

Return value: -/-
**********************************************************************/

/**********************************************************************/
extern void I2C_Device_write  ( struct I2C_Device* ptr,
                                unsigned char Data );
/**********************************************************************
Description:  Write one Byte to I2C bus

Parameters:    - Pointer to the device structure
              - Data Byte to write

Return value: -/-
**********************************************************************/

/**********************************************************************/
extern unsigned char I2C_Device_read( struct I2C_Device* ptr );
/**********************************************************************
Description:  Read from I2C device

Parameters:    - Pointer to the device structure

Return value: Read Byte from slave
**********************************************************************/

/**********************************************************************/

/* End of i2c.h */
```

6. Application Note History

1. Application Note IC: “CDC 32xxG I^2C Port Expander Interfacing”, Feb. 6, 2003, 6251-593-4-1AN. First release of the application note IC.

Micronas GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@micronas.com
Internet: www.micronas.com

Printed in Germany
Order No. 6251-593-4-1AN