i2c.library

COLLABORATORS			
	TITLE : i2c.library		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

i2c.library

1.1 i2c.library

```
                                    i2c.library
                  A Postcard-Ware Library

     © Copyright 1993-96 by GizmoSoft Productions / Brian Ipsen
               © Copyright 1997-98 by Wilhelm Nöker

                        All rights reserved.
```


1.2 Table Of Contents

1.3 Disclaimer

The authors neither assume nor accept any responsibility for the use or misuse of this library, code or connected hardware.

The authors will not be liable for any damage arising from the failure of this library to perform as described, or any destruction of other programs or data residing on a system attempting to use the library functions. The user of this program uses it at his or her own risk.

No guarantee of any kind is given that the library described in this document is 100% reliable. You are using this material on your own risk.

1.4 About the I2C bus

History and background

Control programs

Two lines

Adressing

Applications

1.5 History and background

An increasing number of complex integrated circuits, ranging from real-time clocks to frequency synthesizers, is provided with an I2C bus interface. Not surprisingly, the I2C bus is found in a wide variety of electronic equipment, including telephones, car radios, television sets and video recorders.

The acronym I2C stands for Inter-IC Communication, and the network was developed by Philips to reduce the number of connections between integrated circuits. This proved feasible in pratice mainly because many ICs have a large number of pins that carry information that is not time-critical and, therefore, suitable for conveying via a relatively slow serial bus with fewer connections than would be required for a high-speed parallel interface. The implementation of the I2C bus on a real-time clock chip, for instance, may reduce the number of pins from 40 to as few as 8. This results in a much simpler PCB design with all benefits of lower production cost and smaller risks of faults developing in equipment. However, a number of connections, including those for the supply voltage, for clock signals, etc., can not be replaced by a serial communication protocol. It will be clear that these connections remain necessary as before.

All ICs that use the I2C bus are in principle connected to two lines (check figure 1). A central bus interconnects two microcontrollers, a memory, a gate array and an LCD driver.

In spite of their wide diversity as regards function and appli-cation, all I2C-compatible integrated circuits have one common feature: all control commands and data are conveyed via a serial bus, according a predefined communication protocol. The serial bus takes the form of three lines: ground, clock (SCL) and data (SDA).

Normally, any I2C configuration has at least one master (an IC capable of initiating the data exchange processes and generating a master clock signal) and one or more slaves (ICs that do the actual work). A master can be a microprocessor such as an 8048, an 8051 or a 68000, which are available in special versions with a built-in I2C bus interface. Two I/O port lines of the microprocessor are used as SDA and SCL lines. Together with the ground line, this implements an I2C bus which allows serial communication between 'bused' devices at a rate of up to 100 kbit per second.

It is also possible to emulate a I2C bus master on a computer with this library and some hardware. This requires 3 lines from a port, which is for example (in case you're using the parallel interface from Jan Leuverink's TeleText package) D2, SEL and POUT.

1.6 Control programs

The two communication lines, SDA and SCL, are connected to open-drain or open-collector outputs, and have one, common, pullup-resistor (check figure 2). This arrangement is called a wired AND-structure.

Adding or removing one or more I2C components on the bus therefore does not affect the operation of already connected ICs, nor does it affect the software that runs on the system. In fact, the software is capable of automatic detection of the hardware configuration. This allows programs to be written for complex systems that do not provide certain features unless the relevant chips are connected to the bus. The absence of these chips is automatically detected by the master controller which interrogates certain addresses.

Existing software may be extended with subroutines written for add-on ICs without affecting the operation of the ICs already installed. This allows existing control programs to be used for a long time without the need of a completely new version every time the hardware is modified. This high level of compability is achieved by virtue of the fixed adresses of the ICs on the I2C bus.

1.7 Two lines

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a pull-up resistor (see figure 2). When all output transistors of connected devices are off, the bus is free, and both lines are high. When an IC is ready to transmit a data block, it pulls SDA low to mark a start condition. From that moment, all other ICs 'know' that the bus is in use. Arbitration procedures come into effect should one or more ICs claim access to the bus simultaneously. When the start condition is recognized, the SDA line is available for carrying databits. The clock line, SCL, determines the validity of the data levels on the SDA line (check figure 3).

The start of any data exchange via the bus is marked by SDA going low while SCL is high, i.e., by a start condition (check figure 4). The level on the SDA line is read by all ICs on the bus during the positive part of the clock pulse. However, only the IC selected by the transmitted address-code responds to the information by actually loading the data and returning an acknowledge pulse. This pulse is generated by the addressed slave device by pulling the data line low for one clock period after the eight clock periods reserved for the databits (check figure 4).

When none of the ICs in the system responds to the transmitted data, the master does not receive an acknowledge pulse. This means that either the addressed slave is busy performing some real-time function, the address is wrong or there is no device that responds at that particular address. The bus is free again after the transmission of the last data bit. Both SCL and SDA revert to high, and the bus may be used to convey the next data block.

The function of the SCL line is to generate one clock pulse for every transmitted databit. Each master must generate its own SCL signal. Although the frequency of this signal is not fixed, certain minimum timing specifications must be preserved. In practice, the I2C bus allows a maximum data speed of about 100 Kbit/s.

1.8 Adressing

Each IC on the I2C bus has its own, unique 7-bit address, which is determined by the manufacturer and hard-coded into the chip. The type PCF8583 real-time clock chip for example, is selected by sending binary code 101000x. The last bit is user-preset (x is 0 or 1) to allow two identical ICs to be used in parallel by tying their inputs to ground or the positive supply to set the address to 1010000 or 1010001 respectively. Similary, certain ADCs, DACs chips and memories may be hard-wired to map them at one of up to eight adresses in a cluster.

The data clock conveyed via the bus invariably consist of 8 bits. The bit that follows the address indicates the start of a read or write operation with the selected IC. Bit 8 is low for a write operation, and high for a read operation.

1.9 Applications

There is much more to the concept of the I2C bus than can be described here. The full specification of the system may be found in the I2C-bus Specifications by Philips Components. The I2C bus is relatively simple to implement on almost any microcomputer system that has at least one user port. If nessecery, external buffers may have to be added to make such a port bidirectional. Some microcomputers, including the Acorn Archimedes, even have an I2C interface a a standard feature.
Developers of small stand-alone microprocessor systems may find the I2C version of the 8048, the PCF84C00T, a good starting point for the design of a dedicated control system.

1.10 About the Library

1.11 Using i2c.library

Using the library should not be that complicated. Take a look at the AutoDoc file (named i2c.doc) for the library and at the example-code. This should give you enough hints on how to use the supplied routines.

In case you’re not much of a programmer, but a hardware hobbyist, you will probably like the contents of the "bin" directory: "SendI2C" and

"ReceiveI2C" are simple Shell commands for controlling and querying
I2C chips, "I2Cscan" is a diagnostic tool.

1.12 Library limitations

As you may recall from earlier chapters, I2C-bus clock timing isn't
really critical or something. However, the I2C-bus specification by
Philips describes that the maximum clock rate for SCL is 100 kHz.
Exceed that speed, and your chips don't reply any more. That's a fact.
The problem is, that i2c.library design was always aimed at a good
utilisation of this maximum bandwidth, and the solution was to make
timing adjustable, rather than simply reliable. (Reliable would have
meant using timer.device, which gives a hopelessly huge overhead when
it comes to 5 microsecond intervals.) This has turned out to be a
major reason for confusion among first-time users.

"Adjustable" means for you as an end user, that you need to know a
timing parameter suitable for your system. (Hint: try 1 for ECS, 5 for
AGA, or simply 10 to be really on the safe side.) The installer script
will ask you for this parameter, test it (if possible) and store it in
an environment variable (I2CDELAY). Note however, that application
programs may override this environment value, and many actually do
(Wilhelm Nöker's "VideoText" for example), because that was the only
way to adjust the timing with older versions of the library. So don't
be surprised if this value is asked of you (one way or the other) more
than once.

1.13 History

i2c.library came into being a while ago in connection with a hardware
project named TeleText (see hard/hack/tt120.lha on Aminet). As can
still be seen from those documents, Jan Leuverink was planning to
implement the I2C-bus routines for his program in a shared library.
However, Brian Ipsen, who helped him with testing at that time,
apparently was faster (or had more time) and was the one to actually
write the library.

Since those days, i2c.library has undergone quite a few changes. As of
v39.0, these changes have been made by Wilhelm Nöker.

```
v40.0
  - Configuration by I2CDELAY environment variable
  - I2C_Base exports performance counters and some interface type
    information

v39.4
  - Bugfix: Misplaced delay loops would slow down the overall clock
    rate, but would not resolve all timing problems.
  - Better version string
```

```
   v39.2
     - Protected SendI2C / ReceiveI2C by a semaphore, so that more than
       one client at a time can use the library. (Useful for I$^2$C buses
       with more than one chip attached.)
     - Made functions AllocI2C, InitI2C, FreeI2C obsolete.
     - Replaced bus timing by timer.device / by empty loops
       with timing by loops of CIA read accesses.
     - SendI2C / ReceiveI2C return more detailed error codes.
     - SendI2C / ReceiveI2C ignore the supplied R/W address bit and set
       it to its proper value.
     - Added functions I2CErrText, ShutDownI2C, BringBackI2C.

   v38.0
     - Added function GetI2COpponent.

   v37.2
     - Moved code to set timerdelay to zero into the library initcode.

   v37.1
     - First official release
```

1.14 About the Authors

1.15 Who did it?

Original library design and documentation by Brian Ipsen.

Brian Ipsen bipsen@usa.net
Ved Andebakken 10, 4.tv. fido: 2:238/67.97
DK-2000 Frederiksberg
Denmark

Current binary implementations by Wilhelm Nöker.

Wilhelm Nöker wnoeker@t-online.de
Hertastr. 8
D-44388 Dortmund
Germany

1.16 Where Are They Now?

Brian Ipsen:

> His Amiga unfortunately almost died around February 1st, 1996. Since that time he hasn't used it much, but instead jumped to a Pentium-based PC, which he's currently using. No plans about upgrading/repairing the Amiga (unless he sees a pretty good price on a working A2000 motherboard or a cheap A4000 :-)).

Wilhelm Nöker:

> Incorrigible Amiga user, however with a daytime job that involves writing device drivers for Windows 95/NT. %-|

1.17 License

Using i2c.library in a commercial or shareware project will require explicit permission from Wilhelm Nöker. (Permission will most likely be granted in exchange for a keyfile or a fully functional copy of the finished product.)

Using i2c.library in a non-commercial project, no matter if it is to be published or not, will require a short thank-you message, either postcard or e-mail, to either Brian Ipsen or Wilhelm Nöker.

1.18 Acknowledgements

This guide file was created using Text2Guide by Stephan Sürken.

Thanks to Thorsten Marquardt <thom@kaupp.chemie.uni-oldenburg.de>, who made the Maxon include files.

And thanks goes especially to Jan Leuverink, whose work got all this started.