i2c.library Manual

COLLABORATORS			
	TITLE : i2c.library Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 31, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

i2c.library Manual

1.1 i2c.library Manual

```
                                        i2c.library
                A Postcard-Ware Library
                written by Brian Ipsen,

```


1.2 Disclaimer

```
                Disclaimer.
~~~~~~~~~~~~
  The author neither assumes nor accepts any responsibility for the
 use or misuse of this library, code or connected hardware.
  The author  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.

i2c.library is Postcard-Ware (concept introduced by Klaus Seistrup, DK). If you like this software and use it regulary, you are obliged to send a postcard to the author. The address can be found at
How to reach the author
.

COMMERCIAL USAGE
~~~~~~~~~~~~~~~~
Commercial usage is allowed if the following conditions are met:

a) You state in your documentation that your program uses i2c.library and that the i2c.library is Copyright (c) Brian Ipsen.

b) You send me a copy of your finished product(s) using i2c.library.

If these conditions are met you are allowed to include the library with your commercial product.

All of the files copyrighted by the author must remain unmodified. None of these files may be distributed on its own, the entire package must be distributed as one whole. 'test_lib.c' is full public domain and can be used in any way you like.

Whether your program is freely distributable or commercial, you must state in your documentation that your program uses i2c.library and that the i2c.library is Copyright (c) Brian Ipsen.

1.3 Introduction

Introduction.
~~~~~~~~~~~~
I have helped Jan Leuverink with testing the hardwareproject named TeleText (See
Acknowledgements
), and I saw in the documents to his hardware, that he planed to implement the I^{2}C-bus routines for his program in a shared library. I guess I was faster or had more time, because this is the result of some hours of programming and some sourcecode, that he sent to me.

1.4 About the I^{2}C bus

```
                                    About the I$^2$C bus.
                          ~~~~~~~~~~~~~~~~~~

                       History and background

                    Control programs for the I$^2$C bus

                               Two lines

                               Addressing

                              Applications
```

1.5 History and background

```
History and background.
~~~~~~~~~~~~~~~~~~~~~~
 An increasing number of complex integrated circuits, ranging from
real-time  clocks to frequency  synthesizers, is  provided with an
I$^2$C bus interface.  Not surprisingly,  the I$^2$C bus  is found  in a
wide  variety of  electronic equipment,  including telephones,  car
radios, television sets and video recorders.
 The acronym I$^2$C 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 I$^2$C 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 I$^2$C bus  are in  principle  connected to two
lines (check  figure 1 ). A central bus interconnects two microcon-
trollers, a memory, a gate array and an LCD driver.
 In spite  of their wide  diversity as regards  function and appli-
cation,  all I$^2$C-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 I$^2$C  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
I$^2$C 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 I^{2}C 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 I^{2}C bus master on a computer with this library and some hardware. This requires 3 lines from a port, which is D2, SEL and POUT with this library.

1.6 Control programs

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 I^{2}C 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 vitue of the fixed adresses of the ICs on the I^{2}C bus.

1.7 Two lines

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 SDA line is read by all ICs on the bus during the position 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 pulling the data line low for one clock period after the eight clock periods reserved by 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 I^{2}C bus allows a maximum data speed of about 100 Kbit/s.

1.8 Adressing

Adressing.
~~~~~~~~~~
Each IC on the I^{2}C bus has its own, unique 7-bit address, which is determined by the manufacturer and burned 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 identfy 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

Applications.
~~~~~~~~~~~~
There is much more to the concept of the I^{2}C bus than can be described here. The full specification of the system may be found in the I^{2}C-bus Specifications by Philips Components. The I^{2}C 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 I^{2}C interface a a standard feature. Developers of small stand-alone microprocessor systems may find the I^{2}C version of the 8048, the PCF84C00T, a good starting point for the design of a dedicated control system.

1.10 Using i2c.library

```
Using i2c.library.
~~~~~~~~~~~~~~~~~~
  Using the  library should not be that complicated. Take a look in
 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.
  To use    the    I$^2$C-bus    interface    you    need    to    have    the
 TeleText-project  connected  to  your  Amiga,  or you need to make
 some other hardware, so the library can be used.

Schematics for the hardware
```

1.11 Library limitations

```
Library limitations.
~~~~~~~~~~~~~~~~~~~~
  Because of  the  hardware  and the way to operate the needed bits
 on  the parallel-port, there's a disadvantage in speed. The I$^2$C-bus
 specifications  by  Philips  describes  that the maximum clockrate
 for  SCL is 100 KHz. With the used hardware and this software, the
 max.  speed  on SCL will be about 25 KHz on a plain un-accelerated
 68000.
```

1.12 History

```
History.
~~~~~~~~
  v38.0
    - Added function GetI2COpponent.

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

  v37.1
    - First official release
```

1.13 Acknowledgements

```
Acknowledgements
~~~~~~~~~~~~~~~~
  The following  names  are either trademarks or the efforts of the
 person and/or company listed (in random order):

  - Amiga and AutoDoc by Commodore-Amiga, Inc.

  - I$^2$C-Bus by Philips Components
```

- TeleText project by Jan Leuverink

- Dice and fdtolib by Matt Dillon

- SAS/C by SAS Institute

- Postcard-Ware by Klaus Seistrup

Thank-you's:
~~~~~~~~~~~~
Thanks goes especially to Jan Leuverink, because most of the code in the i2c.library is based on some C sourcecode I received from him. The schematic for the hardware was also introduced to me by Jan Leuverink.
He's also the only beta-tester of the library (at this moment).

1.14 Author's notes on distribution

Author's notes on distribution
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1.15 How to reach the author

How to reach the author
~~~~~~~~~~~~~~~~~~~~~~~

The author (me) may be reached in any of the following ways:

Postal service:

Brian Ipsen
Lyngens Kvarter 274
DK-7400 Herning
Denmark

BBS:
For a while at least, I will be picking up mail on MicroAmiga BBS in Randers/DK, (2:230/813.0@fidonet & 2:230/814.0@fidonet), an official MEBBSNet Support Board in Europe (I'm also working as CoSysop there).
~~~~~~~~~~~~~~~~~~~~~~~