iec.doc

COLLABORATORS			
	TITLE : iec.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

iec.doc

1.1 iec.doc

```
--background()

ACPtr()

CIOut()

Listen()

Second()

Talk()

TkSA()

UnListen()

UnTalk()
```

1.2 iec.library/--background

```
                Disclaimer: information contained herein is not wholly proven ↩
                   reliable.
I didn't find enough official documentation about IEC interface and DOS
standards, so I examined the original ROM disassemble, some C64 disk
utilities and some Commodore 64 Emulators.
If you find something wrong, or own better specifications, please contact me.


Fabrizio Farenga
(f.farenga@agora.stm.it)



  THE SERIAL BUS
```

The IEC serial bus is a daisy chain arrangement designed to let the computer communicate with devices such as the VIC-1541 DISK DRIVE and the VIC-1525 GRAPHICS PRINTER. The advantage of the serial bus is that more than one device can be connected to the port. Up to 5 devices can be connected to the serial bus at once.

There are three types of operation over a serial bus: CONTROL, TALK, and LISTEN. A CONTROLLER device is one which controls operation of the serial bus. A TALKER transmits data onto the bus. A LISTENER receives data from the bus.

The computer is the controller of the bus. It also acts as a TALKER (when sending data to the drive, for example) and as a LISTENER (when loading a program from the disk drive, for example). Other devices may be either LISTENERS (the printer), TALKERS, or both (the disk drive). Only the computer can act as the controller.

All devices connected on the serial bus will receive the whole data transmitted over the bus. To allow the computer to route data to its chosen destination, each device has a bus ADDRESS. By using this device address, the computer can control access to the bus. Addresses on the serial bus range from 4 to 31.

The computer can COMMAND a particular device to TALK or LISTEN. When the computer commands a device to TALK, the device will begin putting data onto the serial bus. When the computer commands a device to LISTEN, the device addressed will get ready to receive data (from the computer or from another device on the bus). Only one device can TALK on the bus at once; otherwise, the data will collide and the system will crash in confusion. However, any number of devices can LISTEN at the same time to one TALKER.

A SAMPLE TRANSMISSION

The basic operations, during a trasmission from computer to drive are the following:

- OPEN TRASMISSION
- SEND DATA
- CLOSE TRASMISSION

This can occur when we want to send a single command to the drive, and we don’t ask for reply. As example we could send the "I" command to the drive unit 8 to reset it.

In Commodore Basic v2.0 we should use something like:

```
1 OPEN 1,8,15
2 PRINT #1,"I"
3 CLOSE 1
```

- In the first line we ask to open a logic file number (1), to the device number 8 (the drive) on the command channel (15).

- In the second line we send to the logic file 1 (after the previous OPEN, it refers to the device number 8) the "I" character.

- In the third line we CLOSE the logic file number 1.

Now the drive will be reset.

Doing the same operations, using the iec.library functions and 'C' language is easy:

```
/* Step 1 */
Listen(8);
Second(CMD_OPEN+15)
UnListen();

/* Step 2 */
Listen(8);
Second(CMD_DATA+15)
CIOut('I');
UnListen();

/* Step 3 */
Listen(8);
Second(CMD_CLOSE+15)
UnListen();
```

Step 1 the device 8 (we suppose a 1541 drive) will become a LISTENER. We send it a secondary address to perform an open command on the channel 15 (command channel). Then we end the opening session, ordering to the device to UNLISTEN.

Step 2 the device 8 will become a LISTENER again. We notify by using the secondary address that we are going to send data. Using the

```
CIOut()
```

function, we send the 'I' character, then close the data trasmission.

Step 3 we end up operations sending a secondary address that includes the CMD_CLOSE command.

Please refer to the example sources to obtain further information.

Note: don't issue any command to the 1541 if it is connected but turned off. Due to an odd bug, you'll not get the "DEVICE NOT PRESENT ERROR" and you'll lock up the Amiga! If the 1541 is physically disconnected, you'll get the proper error.

1.3 iec.library/ACPtr

NAME

ACPtr -- Input byte from serial port.

```
 SYNOPSIS
char = ACPtr()
D0

LONG ACPtr(void)


 FUNCTION
This is the function to use when you want to get information from a
device on the serial bus, like a disk drive. This function gets one
data byte from the serial bus using full handshaking. A -1 is
returned when EOF or an error is encountered. To prepare for this
function the
               Talk()
                function must be called first
to command the device on the serial bus to send data through the bus.
If the input device needs a secondary command, it must be sent by
using the
               TkSA()
                function before calling this function.


How to Use:

 1) Command a device on the serial bus to prepare to send data to
    the computer. (Use the
               Talk()
                and
               TkSA()
                functions.)
 2) Call this function
 3) Print or otherwise use the data.


 INPUTS


 RESULTS
char - character read (0-255)


 EXAMPLE

/*GET A BYTE FROM THE BUS*/

UBYTE c;

c=ACPtr();
printf ("Byte received - %d\n",c);


 BUGS


 SEE ALSO
```

```
               Talk()
               ,
               TkSA()
```

1.4 iec.library/CIOut

```
                   NAME
CIOut -- Output byte to serial port.

 SYNOPSIS
CIOut(CHAR)
      D0

void CIOut(char)


 FUNCTION
This function is used to send information to devices on the serial
bus. A call to this function will put one data byte onto the serial
bus using full serial handshaking. Before this function is called,
the
               Listen()
                function must be used to command a device on the
serial bus to get ready to receive data (if a secondary address is needed,
this must also be sent by using the
               Second()
                function). Device must
be listening or the status word will return a timeout. This function
always buffers one character (the function holds the previous
character to be sent back). So when a call to
               UnListen()
                is issued
to end the data transmission, the buffered character is sent with an
End Or Identify (EOI) set, then UNLISTEN is sent to the device.

How to Use

 1) Use
               Listen()
                function (and
               Second()
                if needed).
 2) Call this function to send the data byte.


 INPUTS
char - character to transmit (0-255)


 RESULTS


 EXAMPLE
```

```
/*SEND A BYTE TO THE BUS*/

ACPtr(0xff);


 BUGS


 SEE ALSO

              Listen()
              ,
              Second()
```

1.5 iec.library/Listen

```
                  NAME
Listen -- Command devices on the serial bus to LISTEN.

 SYNOPSIS
Listen(device)
       D0

void Listen(UBYTE)


 FUNCTION
This function will command a device on the serial bus to receive
data. A device number between 0 and 31 must be provided before
calling the routine. LISTEN will OR the number bit by bit to convert
it to a listen address, then transmit this data as a command onto the
serial bus. The specified device will then go into listen mode, and
be ready to accept information.


 INPUTS
char - character to transmit (0-255)


 RESULTS


 EXAMPLE

/*SEND LISTEN TO DEVICE NUMBER 8*/

Listen(8);


 BUGS


 SEE ALSO

              Second()
```

,
UnListen()

1.6 iec.library/Second

```
                  NAME
Second -- Command devices on the serial bus to LISTEN.

 SYNOPSIS
Second(secondary_address)
       D0

void Second(UBYTE)


 FUNCTION
This function is used to send a secondary address to an I/O device
after a call to the LISTEN function has been made, and the device is
commanded to LISTEN. The function cannot be used to send a secondary
address after a call to the TALK function. A secondary address is
usually used to give setup information to a device before I/O
operations begin. When a secondary address is to be sent to a
device on the serial bus, the address must first be ORed with one
of the low level command codes: CMD_DATA, CMD_CLOSE or CMD_OPEN.
The secondary address can be any number from 0 to 14. These refer
to channels used to communicate with the disk drive. Channel 15 is
reserved as command channel.


 INPUTS
secondary_address - a valid secondary address (0-15)


 RESULTS


 EXAMPLE

/*OPEN DEVICE #8 WITH COMMAND (SECONDARY ADDRESS) #15 */

Listen(8);
Second(CMD_OPEN|15);


 BUGS


 SEE ALSO

              Listen()
```

1.7 iec.library/Talk

```
                  NAME
Talk -- Command serial bus device to TALK.

 SYNOPSIS
Talk(device)
     D0

void Listen(UBYTE)


 FUNCTION
To use this function a device number between 0 and 31 must be
      provided. When called, it ORes its argument bit by bit to
convert this device number to a talk address. Then data is
transmitted as a command on the serial bus.


 INPUTS
device - a valid device number (8-31)


 RESULTS


 EXAMPLE

/*SEND TALK TO DEVICE NUMBER 8*/

Talk(8);


 BUGS


 SEE ALSO

              TkSA()
              ,
              UnTalk()
```

1.8 iec.library/TkSA

```
                  NAME
TkSA -- Send secondary address after TALK.

 SYNOPSIS
TkSA(secondary_address)
     D0

void TkSA(UBYTE)


 FUNCTION
This function transmits a secondary address onto the serial bus for
```

```
a TALK device. The function sends the number as a secondary address
command over the serial bus. This function can only be called after
a call to the TALK function. It will not work after a LISTEN. A
secondary address is usually used to give setup information to a
device before I/O operations begin.
When a secondary address is to be sent to a device on the serial
bus, the address must first be ORed with one of the low level
command codes: CMD_DATA, CMD_CLOSE or CMD_OPEN.
The secondary address can be any number from 0 to 14. These refer
to channels used to communicate with the disk drive. Channel 15 is
reserved as command channel.


 INPUTS
secondary_address - a valid secondary address (0-15)


 RESULTS


 EXAMPLE

/*OPEN DEVICE #8 WITH COMMAND (SECONDARY ADDRESS) #2 */

Talk(8);
TkSA(CMD_OPEN|2);


 BUGS


 SEE ALSO

              Talk()
```

1.9 iec.library/UnTalk

```
                  NAME
UnTalk -- Command serial bus to UNTALK

 SYNOPSIS
UnTalk()


void UnTalk(void)


 FUNCTION
This function transmits an UNTALK command onto the serial bus. All
      devices previously set to TALK will stop sending data when this
command is received.


 INPUTS
```

```
 RESULTS


 EXAMPLE

/*SEND UNTALK COMMAND TO ALL DEVICES */

UnTalk();


 BUGS


 SEE ALSO

              Talk()
```

1.10 iec.library/UnListen

```
                  NAME
UnListen -- Command serial bus to UNLISTEN

 SYNOPSIS
UnListen()


void UnListen(void)


 FUNCTION
This function commands all devices on the serial bus to stop
receiving data from the computer (i.e., UNLISTEN). Calling this
function results in an UNLISTEN command being transmitted onto the
serial bus. Only devices previously commanded to listen are
affected. This function is normally used after the computer has
finished sending data to external devices. The sending of UNLISTEN
commands listening devices to get off the serial bus so it can be
used for other purposes.



 INPUTS


 RESULTS


 EXAMPLE

/*SEND UNLISTEN COMMAND TO ALL DEVICES */

UnListen();
```

BUGS

SEE ALSO

Listen()