Triceratops

Dual FIFO Serial,

Single Parallel

Expansion Card

for the Amiga 2000, 3000 and 4000

User's Guide

Introduction

Thank you for purchasing the Triceratops dual FIFO serial/single parallel card. It has been carefully designed and tested to provide years of reliable operation. The card has the following features:

** Two serial ports with 16-byte send and receive FIFO buffers.
** Support for hardware and XON/OFF handshaking protocols.
** A programmable baud clock for each serial port (including MIDI rate).
** A bidirectional parallel port.

The software package includes (on the **Triceratops_SW** floppy disk):

** cereal.device -- FIFO serial port device driver that supports up to 32 units (ports).
** extprint.device -- an output-only parallel port device driver for up to 8 units.
** Serprefs -- a utility for setting serial port parameters from a CLI window.
** IOWedge -- a 2.0 utility that enables the use of the Triceratops board with software that does not support external I/O ports.
** NewPortHandler -- a replacment port-handler that enables the use of cereal.device or extprint.device in MountList entries.
** FindTriceratops -- a utility for verifying that the Amiga recognizes all Triceratops cards installed.
** FastMode -- a 2.0 utility to set the "fast mode" option on the parallel ports.

For programming examples, the source files for *FindTriceratops* and *extprint.device* are included in the *sources* directory of the **Triceratops_SW** disk.

Hardware Installation

Do NOT install the board in your computer before reading this entire booklet. Unpack your Triceratops board and lay it on the pink antistatic plastic in which it was shipped (save the plastic and other packing materials). Look over the board so as to familiarize yourself with the various connectors. Notice that one end of the board says "Triceratops Mark II" and the opposite end has three connectors: a DB-25 female socket, a DB-9 male plug and a 10-pin dual in-line header just to the left of the DB-9 connector.

The 10-pin header is where the second serial port connects. Incuded in the box you will find a second DB-9 male connector attached to a metal backplane runner with a 6" length of flat cable. The free end of this cable connects to the Triceratops board; attach it by lining up the 10 pin header with the cable connector such that the RED wire on the flat cable is closest to the number "1" printed on the board. (The red wire will face away from the DB9 and DB25.)

ALWAYS MAKE CERTAIN YOUR AMIGA IS TURNED OFF BEFORE OPENING THE CASE!! Allow 30 seconds for internal fixed disks to spin down and park. If available, use a grounded wrist-strap while handling circuit boards. Otherwise, touch a water pipe or the faceplate screw of a grounded electrical socket with your fingers, then touch the metal chassis of your Amiga. This greatly reduces the chance of static electricity damaging the sensitive components in your Amiga as well as on the Triceratops card. Follow the instructions in your Amiga Users Manual to install the Triceratops card in any one of the unoccupied 100 pin Zorro slots. If you plan to use both serial ports, then remove the blank metal runner from another unused slot and install the DB-9 connector in its place.

Turn on the Amiga. After it has booted up, insert the Triceratops software disk in a drive. Double-click to open that disk's icon. You will see two icons in the disk window: *FindTriceratops* and *Install*. Double-click on *FindTriceratops*. This will run a small program

which will report the number of Triceratops cards found in the system. A small window will open and display the message "Triceratops board at address $nnnnnn" for each board found installed in your Amiga. The 'nnnnnn' is a number that will vary depending on what other cards are installed in your machine. If you have more than one Triceratops board, each one will have a different address. If the Amiga did not find the board, the message "No Triceratops board(s) found!" will be displayed.

Examples: (all addresses are for illustration only--yours may be different)

Triceratops board at address $ea0000 -> one Triceratops card in system

Triceratops board at address $e90000 \ two Triceratops cards in the system
Triceratops board at address $eb0000 /

No Triceratops board(s) found! -> No board installed or not found by system

If you have problems...

If you have difficulty getting the board to fit into a slot, make sure that the bottom tab on the metal backplane runner (below the DB-25 connector) is not bent slightly inward. This causes the tab to catch on the edge of the Amiga 2000 main board or the expansion daughter board of the Amiga 3000 and 4000. Gently bend the tab with a small pair of pliers until it is straight.

If the program *FindTriceratops* fails to report the card you installed, turn your machine off and check to see if the board is fully plugged into its slot. It is possible to get the board to miss the slot socket and still appear to be installed. Also, check to see if the contacts on the expansion slots are free of dirt and corrosion. Use a dry toothbrush to clean away any suspect areas. It is not recommended to use a pencil eraser on plated board connectors. If you still do not see the Triceratops board reported under *FindTriceratops*, try changing the board to another slot. If this does not work, try installing another Amiga autoconfig card into your machine, then run the AmigaDOS 2.0 tool *ShowConfig* (in the Tools directory of the System disk icon) to see if the system recognizes that card. If this card fails to be recognized, your Amiga may need to be serviced. If this card is recognized, then your Triceratops card may be defective. Please contact Scott Advanced Microdesigns for further assistance.

Software Installation

Once your Triceratops board is found by *FindTriceratops*, you can proceed to install the support programs and drivers on your system. You can do this one of two ways:

1) Open the Triceratops floppy icon (be sure the disk is still in a drive), then double-click on the *Install* icon. This will copy the files to the correct places on your system disk.

2) Copy the files manually using a CLI window:

```
copy Triceratops_SW:c/Serprefs to c:Serprefs
copy Triceratops_SW:c/IOWedge to c:IOWedge
copy Triceratops_SW:devs/cereal.device to devs:cereal.device
copy Triceratops_SW:devs/extprint.device to devs:extprint.device
copy Triceratops_SW:l/NewPortHandler to l:NewPortHandler
```

You may optionally copy two other files to your system disk (*Install* does not do this):

```
copy Triceratops_SW:FindTriceratops to c:FindTriceratops
rename devs:MountList to devs:MountListB4Triceratops
```

join devs:MountListB4Triceratops Triceratops_SW:devs/LPTMount to devs:MountList

This completes the hardware and software installation.

The Serial Driver -- *cereal.device*

This driver is compatible with both the 1.3 and 2.0 operating system versions.

No mountlist entry is required to use the driver with a terminal program. Simply change the serial device in your terminal program to *cereal.device.* You do **not** need to use *binddrivers* or any other mounting program. AmigaDOS will automatically load the driver when it is first used.

The *cereal.device* supports up to 10 units in a five-slot system, numbered 0 through 9. Units 0 and 1 reside on the first Triceratops board, while units 2 and 3, 4 and 5, etc. are assigned to each additional Triceratops board installed in your machine. The even-numbered units (0,2,4,etc.) correspond to the DB9 AT-style serial connector located on the circuit board next to the DB-25 parallel connector; the legend 'SIO A' is printed on the circuit board. The odd-numbered units (1,3,5,etc.) correspond to the DB9 connector attached to the ribbon cable which in turn connects to the 10-pin header (SIO B).

The driver will automatically select first-in, first-out (FIFO) buffering at a depth of 8 characters unless this is changed under the utility *Serprefs* (see below). RTS/CTS and XON/XOFF hadshaking are supported and are selected through your terminal software and also via *Serprefs*.

The FIFO buffering of serial data has two significant advantages over the single-byte buffered Amiga internal serial port. The Triceratops hardware and *cereal.device* driver combination are smart enough to interrupt the CPU only after several characters have been sent or received (up to 14, with 2 additional bytes of 'safety space') instead of interrupting the CPU for every single byte. This means that the CPU can spend more time running its other tasks and can efficiently write to or read from the Triceratops serial ports in buffered blocks, even while the serial port hardware buffers are still sending or receivng. This greatly reduces the chance that characters will be lost, even at high baud rates. Under the utility *SerPrefs*, the FIFO buffer 'trigger' depth can be set for 1, 4, 8 or 14 characters. The default is 8 characters, meaning that the board will interrupt the CPU when the 16-byte buffer fills up halfway. Refer to the *SerPrefs* explanation to change this setting.

This driver has been tested with a variety of modems and Amiga-to-Amiga/Amiga-to-PC serial links at up to 57600 baud without any problems. However multitasking with certain programs at this speed can still cause your terminal program to lose characters. Any program that turns off interrupts for an extended time will affect your serial port throughput. Also, if your hard drive interface is on a DMA Zorro II card, you may experience character loss at high speeds. This occurs when the DMA interface occupies the bus for a long time, and refuses to allow the serial driver access. Further, the speed and size of your Amiga's CPU will affect performance; a 25MHz 68030 can service tasks much faster than a 7.159MHz 68000.

There are two software points to consider when running the serial ports at high speeds. The first is that some terminal programs use more overhead in talking to the serial ports, which takes time. A specific example is a terminal program that uses external transfer protocols (such as xprzmodem) versus a terminal program that has built-in protocols. The second point is in that AmigaDOS 2.0 is more efficient than 1.3 in terms of overhead, so one solution is to use AmigaDOS 2.0 to improve performance.

If you wish to access the new serial ports from the CLI, use the IOWedge program to redirect SER: to this driver (see the description of IOWedge below).

The Parallel Driver -- *extprint.device*

The *extprint.device* is a driver intended for printers. It only supports data output, not data input. Unit 0 corresponds to the DB-25 on the Triceratops board labeled 'PARALLEL I/O'. Units 1, 2, etc. correspond to the parallel ports on any additional Triceratops boards installed in your machine. To use the driver, run the IOWedge program to redirect the printer outputs of your word processor or other programs. It is not necessary to use a MountList entry.

If you wish to use a MountList entry, refer to *NewPortHandler* below.

This driver supports two operating modes: the standard mode where the printer uses the handshake signal **!ACK** to interrupt the CPU for each byte of data and the DOS 2.0 "fast mode" option, in which *extprint.device* polls the printer handshake signal **BUSY** to control data output. Since this fast mode polling involves "busy-waiting" (a tight loop that hogs CPU time) the *extprint.device* will automatically set itself to run only after more important system tasks are tended during the course of the polling loop. This prevents the Amiga from 'sleeping' during the printout.

Using MountList Entries -- *NewPortHandler*

NewPortHandler is a new version of "port-handler" that permits the device name and unit to be specified. It is useful for printing from the command line as in *copy file to LPT1:*, although you may prefer to use IOWedge.

To use *NewPortHandler*, create an entry in your devs:MountList file like this:

```
LPT1:
    FileSystem    =  L:NewPortHandler
    Device        =  extprint.device
    Unit          =  0
    StackSize     =  2000
    Priority      =  5
    GlobVec       =  -1
    Surfaces      =  0
    BlocksPerTrack = 0
    LowCyl = 0; HighCyl = 0;
#
```

The name (LPT1), device (extprint.device) and unit (0) may be changed to anything suitable, but the rest must be entered exactly as shown. The file LPTMount in the devs directory of the supplied software disk has the above MountList entry; you may append it to your MountList with the commands:

```
rename devs:MountList to devs:MountListB4Triceratops
join devs:MountListB4Triceratops Triceratops_SW:devs/LPTMount to devs:MountList
```

IOWedge

The program *IOWedge* allows you to use the Triceratops ports with programs that do not allow alternate serial or parallel drivers. **This program is for the 2.0 operating system and will not work under the 1.3 operating system.** It is CLI-only, but can be associated with an icon by using the *IconX* or *XIcon* utilities. The template for *IOWedge* is:

IOWedge SERIAL newserial.device PARALLEL newparallel.device [QUIT]

IOWedge intercepts any calls to begin using the internal serial or parallel ports and gives

you the option of substituting a new driver instead. When an access is attempted to the internal ports, *IOWedge* pops up a 2.0-look window with the name of the task that attempted the port access and a set of buttons. Clicking on the *internal* button allows the task to continue to use the internal port drivers. Clicking on one of the numbered buttons substitiues the new driver you specified, selecting the unit number corresponding to the button clicked on. For example, clicking on the button labeled "1" with *cereal.device* used as the new serial device would substitute unit 1 of the *cereal.device* in place of the internal serial port **for that task only**. Redirection to the new device is done for each task independently. *IOWedge* would pop up another window if another task attempts to use the internal port again. Note that copying data from the CLI to SER:, PAR: or PRT: will also cause *IOWedge* to pop up a window to redirect the data.

To get the usage template at any time type IOWedge with no arguments. Insert the alternate serial or parallel driver you want to use into the template. Example use to redirect normal serial port data to the cereal.device would be:

IOWedge serial cereal.device

To deactivate *IOWedge* and remove it from memory, type *IOWedge quit.*

Setting Serial Preferences -- *SerPrefs*

The *SerPrefs* program allows you to set the serial port preferences for the driver *cereal.device.* If you specify a set of preferences using this program, a 1K file is saved into your S: directory containing those perferences. Then, each time *cereal.device* is opened by any program, those preferences are loaded in. Note that you do not need this program if you are using any typical terminal program that sets the serial port to stored presets. Most terminal programs do this.

SerPrefs is useful, however, if you want to only copy data from the command line to SER: (using *IOWedge*) and have not previously opened *cereal.device* with a terminal program. To get example usage information, just type *SerPrefs* with no arguments. You will see the following information displayed:

```
Example use: SerPrefs myser u=0 baud=38400 parity=none handshake=cts
Options for PARITY: NONE, EVEN and ODD.
Options for HANDSHAKE: NONE, CTS and XOFF.
The FIFO option should normally be set to 0, which means "default mode".
Use the DUMP option to view all preferences (or just one if UNIT is used).
```

The *dump* option will list the current settings of the device and unit number you specify. You substitute the name of the new serial device in place of *myser* in the command line. The name can be typed in as either *cereal.device* or just *cereal.* To see the full template for *SerPrefs*, type *SerPrefs* ?. The following information will be listed:

```
DEVICE/A, U=UNIT/K/N, B=BAUD/K/N, D=DATA/K/N, S=STOP/K/N, P=PARITY/K,
H=HANDSHAKE/K, FIFO/K/N, DUMP/S
```

About the FIFO option: The default value FIFO=0 will set the FIFO trigger level at 8 characters. There are three other values: FIFO=1, FIFO=4 and FIFO=14. These select FIFO trigger levels of 1 character, 4 characters and 14 characters respectively.

Appendices

The following pages provide additional information.

Appendix A: Declaration of limited warranty

Appendix B: Connector pinouts for the Triceratops serial and parallel ports

Appendix C: Useful information for making DB9 to DB25 serial cables

Appendix D: Programmer's documentation on the Triceratops serial/parallel hardware

Appendix A -- Warranty

Scott Advanced Microdesigns, Inc. a West Virginia Corporation warrants that for a period of twelve (12) months from the date of shipment it will repair, or at its option replace, any new apparatus which proves defective in material or workmanship. All repairs and replacements shall be F.O.B. factory. All claims must be in writing to the manufacturer, Scott Advanced Microdesigns.

In no event and in no circumstances shall manufacturer be liable for (a) damages in shipment; (b) failures or damages due to misuse, abuse, improper installation or abnormal conditions of temperature, dirt or corrosives; (c) failures due to operation, intentional or otherwise, above rated capacities; and (d) non-authorized expenses for removal, inspection, transportation, repair or rework. Nor shall manufacturer ever be liable for consequential and incidental damages, or in any amount greater than the purchase price of this apparatus.

This warranty is in lieu of all other warranties, expressed or implied, including (but not limited to) any implied warranties of merchantibility or fitness for a particular purpose. The terms of this warranty constitute any buyer's and/or user's sole and exclusive remedy, and are in lieu of any right to recover for negligence, breach of warranty, strict tort liability or any other theory.

RETURNED GOODS - No goods will be accepted for return unless there is prior written authorization by the manufacturer. In all cases, transportation changes must be prepaid.

INTERPRETATION - There are no conditions or understandings whatsoever, verbal or otherwise, except as written herein and this instrument contains the complete and exclusive agreement between the buyer and seller. No waiver, alteration or modification of any of the provisions hereof shall be binding upon the seller unless made in writing and signed by a duly authorized officer of the seller.

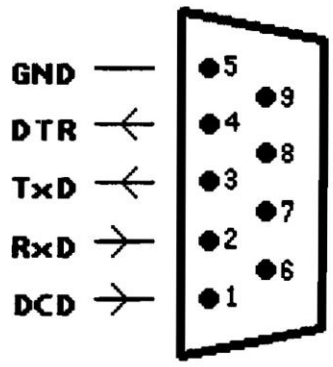
Manufacturer's address:

Scott Advanced Microdesigns
1618 Russell Court
Fairmont, WV 26554

TEL: +1 304 363 4286
FAX: +1 304 366 3201

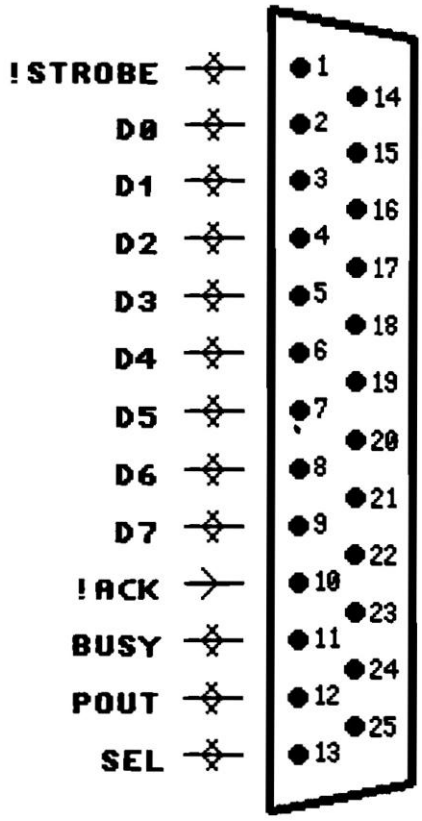
E-mail: srider@bix.com

SERIAL I/O DB9P (MALE) - (both serial ports)



DCD (Data Carrier Detect)
RxD (Receive Data)
TxD (Transmit Data)
DTR (Data Terminal ready)
GND (Signal Ground)
DSR (Data Set Ready)
RTS (Request To Send)
CTS (Clear To Send)

PARALLEL I/O DB25S (FEMALE)



!STROBE (Printer strobe or I/O)
D0-D7 (Printer data or I/O)
!ACK (Acknowledge, input only)
BUSY (Printer busy or I/O)
POUT (Paper out or I/O)
SEL (Line printer selected or I/O)
+5V@100mA (Accessory power)
!INIT (Initialize printer reset out)
GND (Signal grounds)
n/c (pin not connected)

Indicate Input (toward connector) or Output (away from connector)
Bidirectional signal

Appendix C.1 -- Making DB9 to DB25 adapter cables

Many serial peripherals (such as modems) use a 25 pin D connectors as for their serial interface. This conforms to the standard created under EIA spec. RS232-C. In order to most efficiently use the space in the limited area on the Triceratops card's backplane, the "AT" style 9 pin D connector is used for serial interfacing. Here is how to create a suitable adapter cable for interfacing to devices that have a 25 pin D connector.

Materials needed:

9 pin female D connector (DB9S)
25 pin male D connector (DB25P)
Desired length of 10-conductor cable (easier to find than 9-conductor)
Tools: soldering iron, pliers, electrical tape, etc.

Assembly:

Prepare the cable by stripping 2" of the outer insulation from the bundled wires. Then, strip 1/4" of insulation from each conductor. Tin each of the exposed conductors with a small bit of solder. Do this for both ends of the cable. The extra 10th wire is not needed and may be clipped off short or taped over. The rest of the wires are soldered from pins on the DB9S to the DB25P in the following order:

DB9 pin	DB25 pin	Description
1	8	DCD (Data Carrier Detect)
2	3	RxD (Receive Data)
3	2	TxD (Transmit Data)
4	20	DTR (Data Terminal Ready)
5	7	GND (Signal Ground)
6	6	DSR (Data Set Ready)
7	4	RTS (Request To Send)
8	5	CTS (Clear To Send)
9	22	RI (Ring Indicator)

Appendix C.2 -- Assembly of Null-Modem Cables

A null-modem cable is used when one wishes to connect devices of the same configuration; the most frequent application is to connect two computers together to perform serial file transfers or serial networking. For a DTE-to-DTE (computer-to-computer) null-modem cable, use two female D connectors. For a DCE-to-DCE cable, use two male D connectors. Wire them as follows using 7 wire cable:

DB9 pin	DB9 pin
2	3
3	2
4	6
5	5
6	4
7	8
8	7

DB25 pin	DB25 pin
2	3
3	2
4	5
5	4
6	20
7	7
20	6

DB9 pin	DB25 pin
2	2
3	3
4	6
5	7
6	20
7	5
8	4

Appendix D -- Technical Information for Programmers

Manufacturer ID / Product ID: $0850 / $1 (2128 / 1 decimal)

Number of serial ports: 2

Number of parallel ports: 1

Type of I/O chip: Startech ST16C552. Each serial port register set compatible to National Semiconductor 16550. ST16C552 includes bidirectional parallel port.

Card configuration: Amiga Zorro II, autoconfigurable

The card will autoconfigure as a 64K peripheral in the $E90000 to $EF0000 Zorro II configure space. All chip registers occupy the UPPER (even) byte on the data bus (D8-D15). Refer to the datasheets on the following pages for full details on the functions of these registers. Offset addresses to the registers are given below.

SERIAL PORTS

1) Serial port wiring: Ports A and B are RS-232C 9-pin AT style (see appendix B). Port A is accessed via the connector labeled "SIO A" on the card; port B is accessed via the connector labeled "SIO B" to which attaches the accessory cable. The device driver *cereal.device* assigns SIO A to be Unit 0 and SIO B to be Unit 1. In the case of multiple Triceratops cards residing in the system, each extra port is assigned the next-higher unit number. e.g., a second Triceratops board in the system would have its serial ports A and B accessed as Units 2 and 3; a third board would have Units 4 and 5, and so on.

2a) **SIO A** (referred to as "channel B" in datasheets) has the base of its registers at **offset $00** from the autoconfig base address assigned to the board at boot time.

2b) **SIO B** (referred to as "channel A" in datasheets) begins at **offset $80** from the autoconfig base address.

3) Register offsets: (again, refer to datasheets)

At offset **$00**; *if **DLAB** bit of Line Control Register = **0***:
Receiver buffer (**RBR**) [read only]
Transmitter Holding Register (**THR**) [write only]

At offset **$00**; *if **DLAB** bit = **1***: Divisor Latch LS byte (**DLL**)

At offset **$02**; *if **DLAB** = **0***: Interrupt Enable Register (**IER**)

At offset **$02**; *if **DLAB** = **1***: Divisor Latch MS byte (**DLM**)

At offset **$04**; *if **DLAB** = **0***:
Interrupt Identification Register (**IIR**) [read only]
FIFO Control (**FCR**) [write only]

At offset **$04**; *if **DLAB** = **1***: Alternate Function Register (**AFR**)

At offset **$06**; Line Control (**LCR**)

At offset **$08**; MODEM Control (**MCR**)

At offset **$0A**; Line Status (**LSR**)

At offset **$0C**; MODEM Status (**MSR**)

At offset **$0E**; Scratch Register (**SCR**)

4) The transmit and receive **interrupt** signals of *both* serial ports *and* the parallel port interrupt signal are open-collector OR tied to the Amiga system bus signal **INT6**. Reading the IIR of the UARTs or the Status register of the parallel port (see below) will determine exactly what is pulling the INT6 line.

PARALLEL PORT

1) The printer port decodes as three registers at offsets **$c0**, **$c2** and **$c4** from the base address assigned by the system at boot time.

2) Register **$c0** is the 8 bit **I/O port**. Bits 0 to 7 are connected to pins 2 to 9 on the 25 pin D connector. A "hi" bit in the output register corresponds to a 1 on the output port. When set to the appropriate mode, data may also be read in on these pins into the data register.

3) The **Status** register is **read** at offset **$c2**. Also, the **IOSelect** register is **written** at offset **$c2**. When *read*, the !ACK signal, pin 10 on the 25 pin D connector, appears as bit 6. *Take care when reading this register as any pending interrupt will be cleared.*

4) The **Control** register is **written** and the **Command** register is **read** at offset **$c4**. When *read*, the bits of the Command register reflect the state of the corresponding pins on the 25 pin D connector. When written, the DB25 connector pins reflect the state of the Control register. These control lines are open-drain. Here is the list:

ST16C552 signal	bit in register	printer signal	pin on DB25 connector	signal polarity
!STROBE	0	!STROBE	1	(inverted signal)
!AUTOFDXT	1	BUSY	11	(inverted)
INIT	2	POUT	12	(not inverted)
!SLCTIN	3	SEL	13	(inverted)

Bits 4-7 are used for internal operation, refer to the datasheets for further information.

5) The remaining pins on the DB-25 are wired per the Amiga 2000B spec. The following additional signals appear on the 25 pin D connector:

port signal name	pin(s) on DB25 connector
+5V@100mA	14
SIGNAL GROUND	17--25
!INIT	16

Pin 15 is not connected to anything.

(Datasheets follow this page)

PROGRAMMING TABLE

CSB	CSA	DLAB	A2	A1	A0	READ MODE	WRITE MODE
1	0	0	0	0	0	Receive Holding Register A	Transmit Holding Register A
1	0	0	0	0	1		Interrupt Enable Register A
1	0	x	0	1	0	Interrupt Status Register A	FIFO Control Register A
1	0	x	0	1	1		Line Control Register A
1	0	x	1	0	0		Modem Control Register A
1	0	x	1	0	1	Line Status Register A	
1	0	x	1	1	0	Modem Status Register A	
1	0	x	1	1	1	Scratchpad Register A	Scratchpad Register A
1	0	1	0	0	0		LSB of Divisor Latch A
1	0	1	0	0	1		MSB of Divisor Latch A
0	1	0	0	0	0	Receive Holding Register B	Transmit Holding Register B
0	1	0	0	0	1		Interrupt Enable Register B
0	1	x	0	1	0	Interrupt Status Register B	FIFO Control Register B
0	1	x	0	1	1		Line Control Register B
0	1	x	1	0	0		Modem Control Register B
0	1	x	1	0	1	Line Status Register B	
0	1	x	1	1	0	Modem Status Register B	
0	1	x	1	1	1	Scratchpad Register B	Scratchpad Register B
0	1	1	0	0	0		LSB of Divisor Latch B
0	1	1	0	0	1		MSB of Divisor Latch B

REGISTER FUNCTIONAL DESCRIPTIONS

TRANSMIT AND RECEIVE HOLDING REGISTER A/B

The serial transmitter section consists of a Transmit Hold Register A/B and Transmit Shift Register A/B. The status of the transmit hold register is provided in the Line Status Register A/B. Writing to this register will transfer the contents of the data bus (D7-D0) to the transmit holding register A/B whenever the transmitter holding register A/B or transmitter shift register A/B is empty. The transmit holding register empty A/B flag will be set to "1" when the transmitter is empty or data is transferred to the transmit shift register A/B. Note that a write operation should be performed when the transmit holding register empty flag is set.

On the falling edge of the start bit, the receiver internal counter will start to count 7 1/2 clocks (16x clock) which is the center of the start bit. The start bit is valid if the RXA/B is still low at the mid-bit sample of the start bit. Verifying the start bit prevents the receiver from assembling a false data character due to a low going noise spike on the RXA/B input. Receiver status codes will be posted in the Line Status Register A/B.

FIFO INTERRUPT MODE OPERATION

When the receive FIFO (FCR BIT-0=1) and receive interrupts (IER BIT-0=1) are enabled, receiver interrupt will occur as follows:

A) The receive data available interrupts will be issued to the CPU when the FIFO has reached its programmed trigger level; it will be cleared as soon as the FIFO drops below its programmed trigger level.

B) The ISR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared when the FIFO drops below the trigger level.

C) The data ready bit (LSR BIT-0) is set as soon as a character is transferred from the shift register to the receiver FIFO. It is reset when the FIFO is empty.

FIFO POLLED MODE OPERATION
When FCR BIT-0 = 1; resetting IER BIT 3-0 to zero puts the ST16C550 in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately either one or both can be in the polled mode operation by utilizing the Line Status Register.
A) LSR BIT-0 will be set as long as there is one byte in the receive FIFO.
B) LSR BIT4-1 will specify which error(s) has occurred.
C) LSR BIT-5 will indicate when the transmit FIFO is empty.
D) LSR BIT-6 will indicate when both transmit FIFO and transmit shift register are empty.
E) LSR BIT-7 will indicate when there are any errors in the receive FIFO.

PROGRAMMABLE BAUD RATE GENERATOR
The ST16C550 contains a programmable Baud Rate Generator that is capable of taking any clock input from DC-16 MHz and dividing it by any divisor from 2 to 2^{16} -1. Customize Baud Rates can be achieved by selecting proper divisor values for MSB and LSB of baud rate generator.

INTERRUPT ENABLE REGISTER A/B
The Interrupt Enable Register A/B masks the incoming interrupts from receiver ready, transmitter empty, line status and modem status registers to the INTA/B output pin.

IER BIT-0:
0 = disable receiver ready interrupt
1 = enable receiver ready interrupt

IER BIT-1:
0 = disable transmitter empty interrupt
1 = enable transmitter empty interrupt

IER BIT-2:
0 = disable receiver line status interrupt
1 = enable receiver line status interrupt

IER BIT-3:
0 = disable modem status register interrupt
1 = enable modem status register interrupt

IER BIT 7-4:
All these bits are set to logic zero.

INTERRUPT STATUS REGISTER A/B
The ST16C552 provides four level prioritized interrupt conditions to minimize software overhead during data character transfers. The Interrupt Status Register A/B provides the source of the interrupt in prioritized manner. During the read cycle, the ST16C552 provides the highest interrupt level to be serviced by the CPU. No other interrupts are acknowledged until the particular interrupt has been serviced. The following are the prioritized interrupt levels:

Priority level				Source of the interrupts
P	D2	D1	D0	
1	1	1	0	LSR (Receiver Line Status Register)
2	1	0	0	RXRDY (Received Data Ready)
3	0	1	0	TXRDY(Transmitter Holding Register Empty)
4	0	0	0	MSR (Modem Status Register)

ISR BIT-0:
0 = an interrupt is pending and the ISR contents may be used as a pointer to the appropriate interrupt service routine
1 = no interrupt pending

ISR BIT 1-2:
Logical combination of these bits, provides the highest priority interrupt pending.

ISR BIT 3-7:
These bits are not used and are set to zero in ST16C450 mode. **BIT 6-7**: are set "1" in ST16C550 mode.

FIFO CONTROL REGISTER (FCR)

This register is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signalling.

FCR BIT-0:
0 = Disable the transmit and receive FIFO.
1 = Enable the transmit and receive FIFO.

FCR BIT-1:
0 = No change.
1 = Clears the contents of the receive FIFO and resets its counter logic to 0 (the receive shift register is not cleared or altered). This bit will return to zero after clearing the FIFOs.

FCR BIT-2:
0 = No change.
1 = Clears the contents of the transmit FIFO and resets its counter logic to 0 (the transmit shift register is not cleared or altered). This bit will return to zero after clearing the FIFOs.

FCR BIT-3:
0 = No change.
1 = Changes RXRDY and TXRDY pins from mode "0" to mode "1".

FCR BIT 4-5:
Not used.

FCR BIT 6-7:
These bits are used to set the trigger level for the receiver FIFO interrupt.

BIT-7	BIT-6	FIFO trigger level
0	0	01
0	1	04
1	0	08
1	1	14

LINE CONTROL REGISTER A/B
The Line Control Register is used to specify the asynchronous data communication format. The number of the word length, stop bits, and parity can be selected by writing appropriate bits in this register.

LCR BIT1-0:
These two bits specify the word length to be transmitted or received.
00 = 5 bits word length
01 = 6 bits word length
10 = 7 bits word length
11 = 8 bits word length

LCR BIT-2:
The number of stop bits can be specified by this bit.
0 = 1 stop bit , when word length = 5, 6, 7, 8 bits
1 = 1 and 1/2 stop bit , when word length = 5 bits
1 = 2 stop bits, word length = 6, 7, 8 bits

LCR BIT-3:
Parity or no parity can be selected via this bit.
0 = no parity
1 = a parity bit is generated during the transmission; receiver also checks for received parity

LCR BIT-4:
If the parity bit is enabled, LCR BIT-4 selects the even or odd parity format.
0 = odd parity is generated by calculating odd number of 1's in the transmitted data; receiver also checks for same format.
1 = an even parity bit is generated by calculating the number of even 1's in the transmitted or received data.

LCR BIT-5:
If the parity bit is enabled, LCR BIT-5 selects the forced parity format.
LCR BIT-5 = 1 and LCR BIT-4 = 0 parity bit is forced to "1" in the transmitted and received data
LCR BIT-5 = 1 and LCR BIT-4 = 1 parity bit is forced to "0" in the transmitted and received data

LCR BIT-6:
Break control bit.
1 = forces the transmitter output (TXA/B) to go low to alert the communication terminal
0 = normal operating condition

LCR BIT-7:
The internal baud rate counter latch enable (DLAB).
0 = normal operation
1 = select divisor latch register

MODEM CONTROL REGISTER A/B
This register controls the interface with the MODEM or a peripheral device (RS232).

MCR BIT-0:
0 = force DTR~ output to high
1 = force DTR~ output to low

MCR BIT-1:
0 = force RTS~ output to high
1 = force RTS~ output to low

MCR BIT-2:
Not used.

MCR BIT -3:
INTA/B output control.
0 = INTA/B outputs disabled
1 = INTA/B outputs enabled

MCR BIT -4:
0 = normal operating mode
1 = enable local loop-back mode (diagnostics). The transmitter output (TXA/B) is set high (Mark condition), the Receiver inputs (RXA/B , CTSA/B~, DSRA/B~, CDA/B~, and RIA/B~) are disabled. Internally, the transmitter output is connected to the receiver input and DTRA/B~, RTSA/B~ are connected to modem control inputs. In this mode , the receiver and transmitter interrupts are fully operational. The Modem Control Interrupts are also operational, but the interrupt sources are now the lower four bits of the Modem Control Register instead of the four Modem Control Inputs. The interrupts are still controlled by the IERA/B .

MCR BIT 5-7:
Not used. Are set to zero permanently.

LINE STATUS REGISTER A/B
This register provides the status of data transfer to CPU.

LSR BIT-0:
0 = no data in receive holding register
1 = a data has been received and saved in the receive holding register

LSR BIT-1:
0 = no overrun error (normal)
1 = overrun error, next character arrived before receive holding register was empty

LSR BIT-2:
0 = no parity error (normal)
1 = parity error, received data does not have correct parity information

LSR BIT-3:
0 = no framing error (normal)
1 = framing error received, received data did not have a valid stop bit

LSR BIT-4:
0 = no break condition (normal)
1 = receiver received a break signal (RX was low for one character time frame)

LSR BIT-5:
0 = transmit holding register is full. ST16C552 will not accept any data for transmission.
1 = transmit holding register is empty. CPU can load the next character.

LSR BIT-6:
0 = transmitter holding and shift registers are full
1 = transmitter holding and shift registers are empty

LSR BIT-7:
0 = Normal.
1 = At least one parity error, framing error or break indication in the FIFO. This bit is cleared when LSR is read.

MODEM STATUS REGISTER A/B
This register provides the current state of the control lines from the modem or peripheral to the CPU. Four bits of this register are used to indicate the changed information. These bits are set to "1" whenever a control input from the MODEM changes state. They are set to "0" whenever the CPU reads this register.

MSR BIT-0:
Indicates that the CTS~ input to the ST16C552 has changed state since the last time it was read.

MSR BIT-1:
Indicates that the DSR~ input to the ST16C552 has changed state since the last time it was read.

MSR BIT-2:
Indicates that the RI~ input to the ST16C552 has changed from a low to a high state.

MSR BIT-3:
Indicates that the CD~ input to the ST16C552 has changed state since the last time it was read.

MSR BIT-4:
This bit is the compliment of the CTS~ input. It is equivalent to RTS in the MCR during loop-back mode.

MSR BIT-5:
This bit is the compliment of the DSR~ input. It is equivalent to DTR in the MCR during loop-back mode.

MSR BIT-6:
This bit is the compliment of the RI~ input.

MSR BIT-7:
This bit is the compliment to the CD~ input.

SCRATCHPAD REGISTER A/B
ST16C552 provides a temporary data register to store 8 bits of information for variable use.

BAUD RATE GENERATOR PROGRAMMING TABLE (1.8432 MHz CLOCK):

BAUD RATE	16 x CLOCK DIVISOR	% ERROR
50	2304	
75	1536	
110	1047	0.026
150	768	
300	384	
600	192	
1200	96	
1800	64	
2000	58	0.69
2400	48	
3600	36	
4800	24	
7200	16	
9600	12	
19.2K	6	
38.4K	3	
56K	2	2.86
112K	1	

ST16C552 EXTERNAL RESET CONDITION

REGISTERS	RESET STATE
IERA/B	IERA/B BITS 0-7 = 0
ISRA/B	ISRA/B BIT 0 = 1, ISRA/B BITS 1-7 = 0
LCRA/B	LCRA/B BITS 0-7 = 0
MCRA/B	MCRA/B BITS 0-7 = 0
LSRA/B	LSRA/B BITS 0-4 = 0, LSRA/B BITS 5-6 = 1, LSRA/B BIT 7 = 0
MSRA/B	MSRA/B BITS 0-3 = 0,
MSRA/B	BITS 4-7 = Input signals
CR	CR BIT 4 = 0

SIGNALS	RESET STATE
TXA/B	High
RTSA/B~	High
DTRA/B~	High
INTA/B	Three state
INTP	Three state
PD7-PD0	Output mode, PD7-PD0 = 0
STROBE~	Output mode, high

ST16C552 ACCESSIBLE REGISTERS

A2 A1 A0	Register	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0
0 0 0	RHR	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0 0 0	THR	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0 0 1	IER	0	0	0	0	modem status interrupt	receive line status interrupt	transmit holding register	receive holding register
0 1 0	FCR	RCVR trigger (MSB)	RCVR trigger (LSB)	0	0	DMA mode select	XMIT FIFO reset	RCVR FIFO reset	FIFO enable
0 1 0	ISR	0/ FIFOs enabled	0/ FIFOs enabled	0	0	0	int priority bit-1	int priority bit-0	int status
0 1 1	LCR	divisor latch enable	set break	set parity	even parity	parity enable	stop bits	word length bit-1	word length bit-0
1 0 0	MCR	0	0	0	loop back	OP2~	OP1~	RTS~	DTR~
1 0 1	LSR	0/ FIFO error	trans. empty	trans. holding empty	break interrupt	framing error	parity error	overrun error	receive data ready
1 1 0	MSR	CD	RI	DSR	CTS	delta CD~	delta RI~	delta DSR~	delta CTS~
1 1 1	SPR	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0 0 0	DLL	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0 0 1	DLM	bit-15	bit-14	bit-13	bit-12	bit-11	bit-10	bit-9	bit-8

PRINTER PORT PROGRAMMING TABLE:

A1	A0	IOW~	IOR~
0	0	PORT REGISTER	PORT REGISTER
0	1	I/O SELECT REGISTER	STATUS REGISTER *
1	0	CONTROL REGISTER	COMMAND REGISTER

* Reading the status register will reset the INTP output.

PARALLEL PORT DIRECTION SELECT REGISTER (WRITE ONLY)

CONTROL REGISTER (D5)	BIDEN	I/O SELECT REGISTER (D7-D0)	PORT MODE
X	0	xxxxxxxx exp. AA Hex	OUTPUT
X	0	10101010	INPUT
0	1	xxxxxxxx	OUTPUT
1	1	xxxxxxxx	INPUT

REGISTER DESCRIPTIONS

PORT REGISTER
Bidirectional printer port.
Writing to this register during output mode will transfer the contents of the data bus to the PD7-PD0 ports . Reading this register during input mode will transfer the states of the PD7-PD0 to the data bus. This register will be set to the output mode after reset.

PR BIT 7-0:
PD7-PD0 bidirectional I/O ports.

STATUS REGISTER
This register provides the state of the printer outputs and the interrupt condition.

SR BIT 1-0:
Not used. Are set to "1" permanently.

SR BIT-2:
Interrupt condition.
0 = an interrupt is pending
This bit will be set to "0" at the falling edge of the ACK~ input.
1 = no interrupt is pending
Reading the STATUS REGISTER will set this bit to "1"

SR BIT-3:
ERROR~ input state.
0 = ERROR~ input is in low state
1 = ERROR~ input is in high state

SR BIT-4:
SLCT input state.
0 = SLCT input is in low state
1 = SLCT input is in high state

SR BIT-5:
PE input state.
0 = PE input is in low state
1 = PE input is in high state

SR BIT-6:
ACK~ input state.
0 = ACK~ input is in low state
1 = ACK~ input is in high state

SR BIT-7:
BUSY Input state.
0 = BUSY input is in high state
1 = BUSY input is in low state

COMMAND REGISTER
The state of the STROBE~, AUTOFDXT~, INIT, SLCTIN~ pins, and interrupt enable bit can be read by this register regardless of the I/O direction.

COM BIT-0:
STROBE~ Input pin.
0 = STROBE~ pin is in high state
1 = STROBE~ pin is in low state

COM BIT-1:
AUTOFDXT~ Input pin.
0 = AUTOFDXT~ pin is in high state
1 = AUTOFDXT~ pin is in low state

COM BIT-2:
INIT Input pin.
0 = INIT pin is in low state
1 = INIT pin is in high state

COM BIT-3:
SLCTIN~ Input pin.
0 = SLCTIN~ pin is in high state
1 = SLCTIN~ pin is in low state

COM BIT-4:
Interrupt mask.
0 = Interrupt (INTP output) is disabled
1 = Interrupt (INTP output) is enabled

COM BIT 7-5:
Not used. Are set to "1" permanently.

CONTROL REGISTER.
Writing to this register will set the state of the STROBE~, AUTOFDXT~, INIT, SLCTIN pins, and interrupt mask register.

CON BIT-0:
STROBE~ output control bit.
0 = STROBE~ output is set to high state
1 = STROBE~ output is set to low state

CON BIT-1:
AUTOFDXT~ output control bit.
0 = AUTOFDXT~ output is set to high state
1 = AUTOFDXT~ output is set to low state

CON BIT-2:
INIT output control bit.
0 = INIT output is set to low state
1 = INIT output is set to high state

CON BIT-3:
SLCTIN~ output control bit.
0 = SLCTIN~ output is set to high state
1 = SLCTIN~ output is set to low state

CON BIT-4:
Interrupt output control bit.
0 = INTP output is disabled
1 = INTP output is enabled

CON BIT-5:
I/O select. Direction of the PD7-PD0 can be selected by setting or clearing this bit.
0 = PD7-PD0 are set for output mode
1 = PD7-PD0 are set for input mode

CON BIT 7-6:
Not used.

I/O SELECT REGISTER
Software controlled I/O select.
Bidirectional mode can be selected by keeping the BIDEN input in high state and setting CON BIT-5 to "zero or one"
Hardware/software I/O select.
Bidirectional mode can be selected by keeping the BIDEN input in low state and setting I/O SELECT register to "AA" Hex for input or any other value for output.

ST16C552 PRINTER PORT REGISTER CONFIGURATIONS

PORT REGISTER (READ/WRITE)

D7	D6	D5	D4	D3	D2	D1	D0
PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

STATUS REGISTER (READ ONLY)

D7	D6	D5	D4	D3	D2	D1	D0
BUSY~	ACK	PE	SLCT	ERROR STATE	IRQ	1	1
					1 = No interrupt 0 = Interrupt		

COMMAND REGISTER (READ ONLY)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	IRQ ENABLE	SLCTIN~	INIT	AUTO-FDXT~	STROBE~
			0 = IRQ disabled 1 = IRQ enabled				

CONTROL REGISTER (WRITE ONLY)

D7	D6	D5	D4	D3	D2	D1	D0
X	X	I/O SELECT	IRQ MASK	SLCTIN~	INIT	AUTO-FDXT~	STROBE~
	0 = Output 1 = Input		0 = INTP output disabled 1 = INTP output enabled				

Addendum

The freely-distributable programs FastMode, FastVBR, MoveSSP and Ex5Ret0 are included in the Triceratops SW:c directory to permit the user to gain the most performance out of Triceratops serial and parallel I/O. The complete distribution archive of each program is in the "distribution archive" directory on the Triceratops SW disk. Here is a brief description of how and when to use each utility:

FastMode

FastMode, by Sebastiano Vigna, is a DOS2.0 utility to set the PARB_FASTMODE bit of either the Amiga parallel.device or the Triceratops extprint.device. When this flag is on, printing will be much faster (2.5 times faster on a 68010 Amiga 2000 printing to a HP LJIIP), but the printer must be fast enough to keep up. Be sure to run IOWedge first. Usage is 'FastMode ON' or 'FastMode OFF'.

FastVBR

FastVBR, by Michael Sinz, is a utility to relocate the vector base register (VBR) which by default points to low (CHIP) memory into FAST memory. Since the Amiga relies heavily on the use of the vectors pointed to by the VBR for system operation, moving them from CHIP memory to FAST memory results in better efficiency and performance. NOTE: This program only works on 68010 processors and up. It will NOT work on a 68000. This program is particularly useful for Amiga 2000 users with a 68010 on the motherboard or 68020/030 board with 32-bit RAM in the accelerator slot. If you are experiencing character loss on the Triceratops serial ports at high baud rates, try using FastVBR.

MoveSSP

MoveSSP, by Chris Wichura (from Roger Uzun's C original and enhanced by Ed Hanway), is a utility for relocating the system Stack pointer (SSP) from CHIP memory to FAST memory. The program will work on all 680x0 CPUs (including 68000). Like FastVBR, moving the SSP increases system performance as all exception handling and supervisor operations (of which there is a LOT on an Amiga) use the system stack. This utility should be run first in your startup-sequence; see the archive docs for details. As with FastVBR, this program should help the Triceratops serial ports work better at very high baud rates.

FastVBR and MoveSSP are not necessary if the 2.x utility "CPU" is used.

NOTE: As of this writing, two programs have been found to crash the system while printing to the Triceratops parallel port. These are DeluxePaint 4.0 and FinalCopy. Apparently there is an invalid structure data element passed to IOWedge through the printer.device which (in the case of DPaintIV) halts the system with a CPU privilege violation #5, zero divide. This problem does NOT occur with DPaintII or DPaintIII.

Ex5Ret0

I have included a small exception handler I wrote to cure this problem. If a zero-divide is encountered, this program will return a zero as the result of the operation. This prevents the crash in DPaintIV and allows the printer to operate normally. Ex5Ret0 (Exception 5, Return zero) is placed in the list of user-startup or startup-sequence commands. If you wish to remove it later, just run the program again and it will remove itself and restore the old handler.

Scott Rider 7-Mar-1993