

ATARI 1090 Serial/Parallel Card
Firmware Specifications
Joseph Decuir Rev 30 May 84

CONTENTS

- 1.0 Overview
 - 1.1 Scope
 - 1.2 Product History
 - 1.3 Firmware
 - 1.4 Specification Summary
- 2.0 Operating Environment
 - 2.1 Processor
 - 2.2 BOOXL Address Map
 - 2.3 ATARI Operating System
 - 2.4 OS I/O Subsystem
 - 2.5 I/O Data structures
 - 2.6 Firmware Vector tables
 - 2.7 Device Status buffer
- 3.0 Memory Allocation
 - 3.1 ROM
 - 3.2 Private RAM
 - 3.3 Data Buffers
 - 3.4 Scratchpad
- 4.0 Initialization
 - 4.1 Initialize private variables
 - 4.2 Initialize hardware
 - 4.3 Install "R" device
- 5.0 Serial Port Handler
 - 5.1 Open
 - 5.2 Close
 - 5.3 Get
 - 5.4 Put
 - 5.5 Status
 - 5.6 Special, entry processing
 - 5.7 Special:Force Short Block
 - 5.8 Special:Serial Port Controls
 - 5.9 Special:Configure Serial Port
 - 5.10 Special:Translate Serial Data
 - 5.11 Special:Start Concurrent I/O

6.0	Low Level Device Drivers
6.1	Entry processing
6.2	Serial Port Control Device Driver
6.3	Serial Port Configure Device Driver
6.4	Serial Port Status Device Driver
6.5	Serial Port Read Device Driver
6.6	Serial Port Write Device Driver
6.7	Serial Port Stream Device Driver (Concurrent I/O)
6.8	Parallel Port Status Device Driver
6.9	Parallel Port Write Device Driver
7.0	Interrupt Handlers
7.1	Entry processing
7.2	Default Serial Data
7.3	Default Ringin
7.4	Default Parallel ACK
7.5	Default 6532 Timer
7.6	Default 6532 PA7
7.7	Concurrent I/O Serial Data In
7.8	Concurrent I/O Serial Data Out
7.9	Concurrent I/O Break Key Handler
8.0	Hardware Programming Model
8.1	6532 RAM
8.2	6532 Parallel Ports & Timer
8.3	6551 Serial Ports
8.4	LSTTL Parallel Output Ports
9.0	Differences from ATARI 850
9.1	Limitations
9.2	Improvements
9.3	Possible Improvements
10.0	References

1.0 Overview

1.1 Scope

This document contains the specifications for the Atari Serial/Parallel Card (SPC) device handlers and device drivers, to be cut into ROM and included on the card.

References will be included to public and proprietary Atari documents, to avoid unnecessary duplication.

1.2 Product History

In 1979, Atari introduced the 400/800 personal computer family. Since this was before the FCC created Class B standards for RFI, it had to confine all high speed processor bus signals to the inside of a shield. External I/O expansion was much more difficult, implemented with smart devices which communicated over a shared serial bus. An example of these devices is the Atari 850 Interface Module, which adds four RS232 serial ports and a Centronics type printer port.

In 1983, Atari introduced the 600/800XL personal computer family, which included a processor bus port. In 1984, Atari will introduce the 1090 Expansion Box, a powered chassis to hold new I/O devices which attach to the processor (parallel) bus. An example of these devices is the Serial/Parallel Card (SPC?), which adds two RS232 serial ports and a parallel printer port.

1.3 Firmware

The Atari computers contain a ROM Operating System (OS), which includes firmware "handlers" for some standard devices: TV screen ("S"), keyboard ("K"), external disk drives ("D"), a cassette drive ("C"), and a printer ("P"). The 850 was the first device introduced with a non-resident handler ("R" for RS232), which is loaded into RAM at cold-start. It is called by the OS Central Input/Output Utility (CIO). The handler in turn speaks to the 850 control processor through the serial bus using the OS Serial Input/Output Utility (SIO). The 850 controller drives the physical ports. (Refer to section 2.4 or reference 10.1 for descriptions of "handlers" and "drivers".)

The new "Parallel Bus" devices are attached to the processor bus. Their handlers are memory resident, on bank selected ROM (D800-DFFF), called by the Generic Parallel Device Handler (GPDH), included in the OS. Their I/O ports are directly on the bus, mapped into the D1xx area, so the device drivers are also resident on the same bank selected ROM. The device drivers may be called either by the OS Parallel Input/Output Utility (PIO), or by the handlers on the same ROM.

1.4 Specifications Summary

The SFC Firmware has four tasks:

1.4.1 Initialization

When called by the Monitor, the Initialization code must set up the serial and parallel port hardware, the R: serial port device handler, the serial and parallel port device drivers, and the serial and parallel port interrupt handlers. See section 4 for details.

1.4.2 Serial Port Device Handlers

When called through one of six vectors: OPEN, CLOSE, GET, PUT, STATUS and SPECIAL, process "R" device I/O requests. See section 5 for details.

1.4.3 Device Drivers

When called either by the R (RS232) handler (see above) or the resident OS P (printer) handler through PIO, recognize and execute I/O for these devices, and reject other requests. See section 6 for details.

1.4.4 Interrupt Handlers

When called by the system parallel interrupt handler, identify the source, and do whatever is required. See section 7 for details.

2.0 Operating Environment

The Serial/Parallel Card is installed in a 1090 Expansion Box, attached to an Atari XL family personal computer, with 64K RAM (a 600XL needs the expansion memory). The Atari system is described in considerable detail in the documents listed in section 10. The most useful, though slightly obsolete, is reference 10.1, the Atari OS User's Manual. There is also a block diagram of the I/O system included in section 2.4. A summary of the operating environment is included here:

2.1 Processor

All Atari personal computers have a 6502 microprocessor, running at a 1.79 MHz clock rate. The effective clock rate is lower, however, due to cycle stealing by the ANTIC Video Display Processor; the useful rate is about 1.2 to 1.3 MHz.

2.2 800XL Address Map

Address range:	Description
0000 - 7FFF	RAM
8000 - 9FFF	RAM, or ROM cartridge if installed
A000 - BFFF	RAM, or BASIC, or ROM cartridge if installed
C000 - CFFF	OS ROM, or RAM if bank switched
D000 - D5FF	Hardware I/O devices, including parallel bus devices in D1xx
D600 - D7FF	RAM, reserved for parallel device firmware
DB00 - DFFF	OS Floating Point ROM, or Parallel Device Firmware ROM, or RAM (bank switched)
E000 - FFFF	OS ROM, or RAM if bank switched

See reference 10.1, section 4, p20

2.3 Atari ROM Operating System (OS)

The Atari XL OS contains the following major components:

acronym	description
-	Monitor Cold Start and Warm Start Initialization Self-Test
CIO	Central Input/Output
	Resident Device Handlers:
K	Keyboard
S	Screen
E	Editor
D	Disk (but not file management)
C	Cassette
P	Printer
GPDH	Generic Parallel Device Handler

PIO Parallel Input/Output Driver

SIO Serial Input/Output Driver

Interrupt Handlers

Floating Point Arithmetic

See reference 10.1, section 2, p15

2.4 I/O Subsystem

BASIC, the monitor, DOS, or any other applications programs, all communicate with I/O devices or disk files through the resident CIO utility. They exchange information with CIO through the 6502 A, X, and Y registers, and through an Input/Output Control Block (IOCB), section 2.5.3.

On OPEN, CIO locates the device handler, ROM or RAM resident, through the Handler table (HATABS), section 2.5.1, indexed by the device name: e.g. K, S, E, D, C, P, M (modem) or R.

Each HATABS entry contains a pointer to the handler vector table, section 2.5.2. CIO calls a device handler through one of seven entry points specified in this table.

On any call, CIO copies the appropriate IOCB into a fixed area in page zero, the ZIOCB, section 2.5.4, and calls the handler through the vector table. On return, CIO copies the ZIOCB back into the originating IOCB.

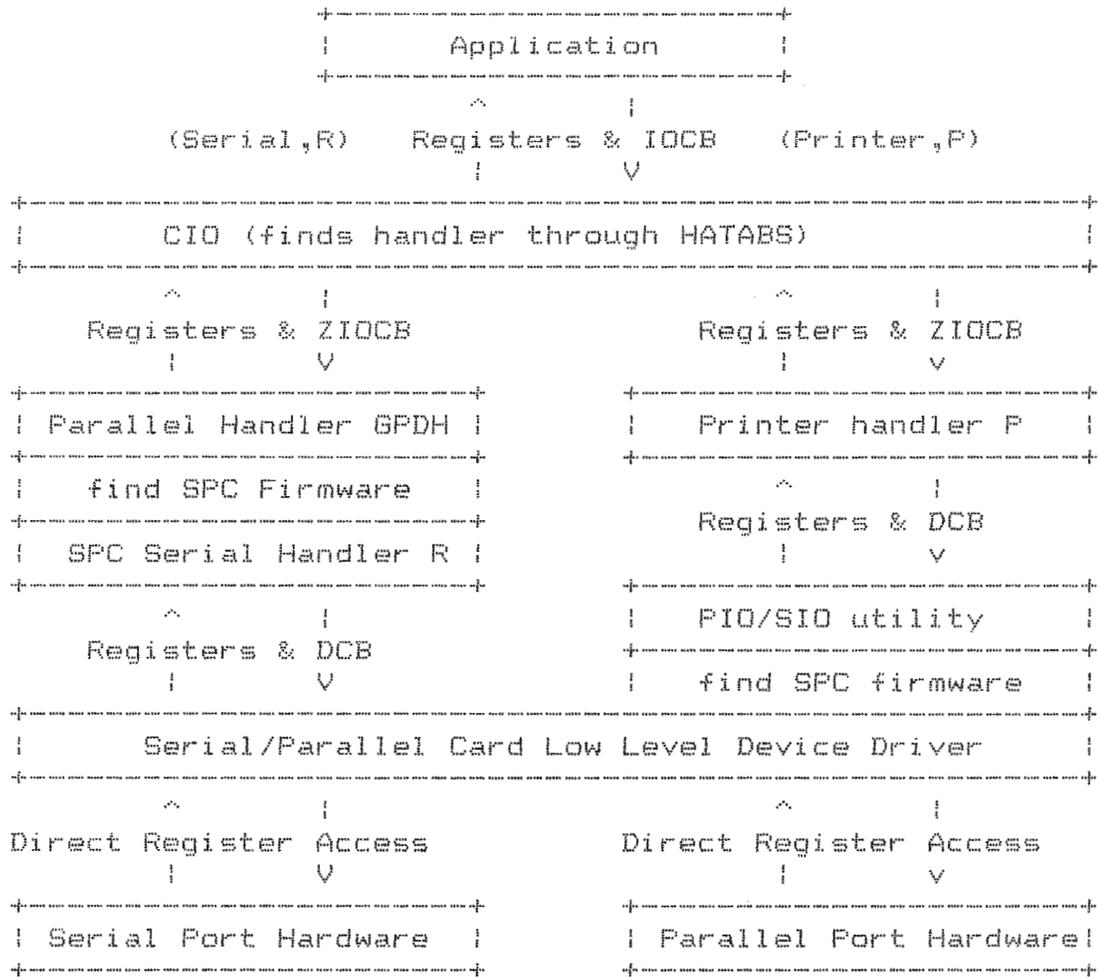
CIO may call the handler directly, if it is either a ROM resident handler included in the system, or a RAM resident handler loaded at cold start (e.g. the 850 Interface Module handler). In the case of a Parallel Bus device such as the R: handler included in the SPC firmware, CIO calls the ROM resident Generic Parallel Device Handler (GPDH). GPDH in turn uses a hardware register and some RAM variables to locate any parallel device handlers located in bank selected ROM. (These locations are described in section 2.5.5). GPDH calls each handler through that ROM's entry table (described in section 2.6). GPDH polls each handler it can find until one of them recognizes and processes the I/O request.

The handler (for example, either the resident P: printer handler or the R: RS232 handler included in the SPC firmware) operates on the information in the registers and the ZIOCB, including the buffers it may point to, and generates calls to a device driver. The handler communicates with the device driver through the Device Control Block (DCB), section 2.5.6, and the registers.

The handler may call the driver directly, if the target device is present within the system (e.g. the resident Keyboard and Screen devices, or the R serial ports devices, when called from the R handler). Otherwise, the handler must call the driver

through the PIO or SIO utilities (common entry through PIO). PIO will first look for a parallel device handler by polling the firmware on each parallel device with a call through the Low level I/O Vector. (The SPC firmware must recognize calls for printer or serial devices). If none of them recognize the request, it is passed on to SIO, which in turn hunts for the device on the external serial bus.

A block diagram of this process, for calls to the printer or a serial port, is shown here:



2.5.1 Handler Table (HATABS @ \$031A)

The Handler table consists of up to 12 three byte entries, one for each type of device. Each entry has two parts: a device type name (one character, such as P or R), and a pointer to the table of entry points for that device type's handler.

address	name	description
031A	HATABS	base address of table
031A+3n	dname	device name in ATASCII e.g. "P" = \$50, "R" = \$52
031B+3n	dhvtl	handler vector table pointer
031B+3n	dhvth	low byte and high byte e.g. the OS ROM resident printer handler is at \$E430, and the OS ROM resident Generic Parallel Device Handler (GPDH), used to enter the SPC Firmware serial port handler, is at \$E48F.

n can range from 0 to 11(\$0B), for twelve device names. Therefore, the index into HATABS can range from 0 to \$21.

2.5.2 Handler Vector Table

Each Handler Vector Table contains six two-byte vectors, a three byte JMP instruction, and an unused byte (usually 00). The six vectors are saved decremented by 1, because they are used by pushing them onto the stack and executing an RTS instruction.

address	name	description
base	OPENV	device OPEN vector -1
base+2	CLOSEV	device CLOSE vector -1
base+4	GETV	device GET vector -1
base+6	FUTV	device FUT vector -1
base+8	STATV	device STATUS vector -1
base+A	SPECV	device SPECIAL vector -1
base+C	INITV	JMP instruction (\$4C)
base+D		device INITIALIZATION vector
base+F	-	00 byte

2.5.3 Input/Output Control Blocks (IOCB)

There are eight IOCBs, of 16 bytes each, located from \$340 to \$3BF. Each OPEN device or disk file must have one. See reference 10.1 p 37 - 39 for a detailed description.

address	name	description
340+16n	ICHID	Handler I.D. an index into HATABS
341+16n	ICDNO	Device number e.g. 2 for R2
342+16n	ICCMD	Device Command: OPEN = 3, GETRECORD = 5, GETCHAR = 7, PUTRECORD = 9, PUTCHAR = #B, CLOSE = #C, STATUS = #D, SPECIAL >= #E.
343+16n	ICSTA	Device Status: 01 = OK, any negative number is an error (see reference 10.1 Appendix B for codes)
344+16n	ICBAL,	Buffer for data transfer, or file
345+16n	ICBAH	specification on OPEN
346+16n	ICPTL,	PUT address -1, for RTS type entry,
347+16n	ICPTH	from BASIC
348+16n	ICBLL,	Length of Buffer
349+16n	ICBLH	
34A+16n	ICAX1,	Auxilliary Information, device specific,
34B+16n	ICAX2	for the handler and/or device driver
34C+16n	ICAX3-	Spare auxilliary data, for use by handler.
34F+16n	ICAX6	(locate using ICIDNO, \$2E, in ZIOCB)

2.5.4 Zero Page I/O Control Block (ZIOCB)

Before CIO calls any particular device handler, it takes the first 12 bytes of the IOCB, and copies them down to the zero page IOCB, ZIOCB. On return, it copies them back again. Locations (340 - 34B)+16n are mapped to \$20 - \$2B.

The last four locations, \$2C - \$2F, are used by CIO, but can also be useful to the handler:

address	name	description
002C	ICSPRZ	spare pointer, used by CIO, and available
002D		to handler before return
002E	ICIDNO	Index back to originating IOCB (=16n)
002F	CIOCHR	Character for current byte operation

2.5.5 Parallel Device Control Locations

On any reference to the SPC firmware, Initialization, Handler, Device Driver, or Interrupt, the calling software has to determine which slots are active, and select them. There is one hardware read register used to identify Interrupt sources (PDVI), a second hardware write register used to select installed devices, (PDVS) and three RAM registers used to manage them. For all five of these locations, each board is mapped to a particular bit; for example, the SPC might be mapped to slot 5, in which case it would be enabled by bit 5 of PDVS, and report its interrupt status in bit 5 of PDVI. (See reference 10.6 section 3 for more details).

address	name	description
D1FF(W)	PDVS	Parallel Device Select (shadow at 0248)
D1FF(R)	PDVI	Parallel Device Interrupt Status
0247	FDVMSK	Parallel Device Select Mask
0248	SHPDVS	Parallel Device Select Shadow
0249	FDIMSK	Parallel Device Interrupt Mask

2.5.6 Device Control Block (DCB)

There is one Device Control Block, located from \$300 to \$30B. See reference 10.1, p130 for a detailed description.

address	name	description
0300	DDEVIC	Device Bus I.D. e.g. printer P = 40-4F, RS232 ports = 50-53 (50,51 for R1,R2)
0301	DUNIT	Device Unit e.g. 1 for R1, 2 for R2
0302	DCOMND	Device Command (see ref 10.1, Appendix I for all values)
0303	DSTATS	Device Status on driver call, D7 = write needed D6 = read needed on driver return, operations status: O1 = OK, any negative value is an error (see ref 10.1, Appendices B and C).
0304	DBUFLO,	pointer to source or destination buffer
0305	DBUFHI	
0306	DTIMLO	Device Timeout, in units of 1.067 seconds.
0308	DBYTLO,	count of size of data buffer
0309	DBYTHI	
030A	DAUX1,	Auxilliary bytes, with device specific
030B	DAUX2	meanings (often passed from ICAX1Z,2Z)

2.6 Required Firmware Tables

address	name	description
D800,	CHECKL,	ROM checksum: low byte,
D801	CHECKH	high byte
D802	RREV	ROM Revision number
D803	ID1	constant #80
D804	RNAME	name (optional, tbd)
D805	LLIOV	"Low-level I/O Vector" = JMP (#4C)
D806,7		Device Driver entry point address
D808	IRQHV	"IRQ Handler Vector" = JMP (#4C)
D809,A		IRQ Handler entry point address
D80B	ID2	constant #91
D80C	DNAME	device name, in ASCII ("R" = #52)
D80D,E	ROPENV	Serial Device Open Vector -1
D80F,10	RCLDSV	Serial Device Close Vector -1
D811,12	RGETV	Serial Device Get Vector -1
D813,14	RPUTV	Serial Device Put Vector -1
D815,16	RSTATV	Serial Device Status Vector -1
D817,18	RSPECV	Serial Device Special Vector -1
D819	RINITV	"Initialization Vector" = JMP (#4C)
D81A,1B		Device Initialization vector
D81C	-	constant 0

2.7 Device Status Buffer

When a device STATUS command is executed, the STATUS handler returns additional information beyond the operation status returned in the Y register. There is a four byte buffer in memory for this information to be returned:

Address	Label	Description
02EA	DVSTAT	base address for returned device status
02EB		
02EC		
02ED		

The SPC firmware implements a serial port and a printer STATUS command, and the codes they return in DVSTAT are described in sections 6.4.3 and 6.7.3.

3.0 Memory Allocation

The SPC Firmware will reside in 2K bytes of bank selected ROM, located at D800 - DFFF. The SPC card also has 128 bytes of RAM, bank selected with the ROM and I/O devices, at D100 - D17F. When the SPC firmware is called, the I/O system data structures (e.g. ZIOCB, DCD, see section 2) may contain pointers to data buffers in the main user RAM area 0700 - BFFF. Finally, since the device drivers operate in place of the OS Serial Input/Output Utility (SIO), they may be able to use some of SIO's scratchpad area in page zero.

3.1 ROM

The SPC Firmware: the Initialization Code, the Serial Handler, the Serial and the Parallel Device Drivers, the Interrupt Handlers, and any necessary tables, all must fit into the 2K byte block bank selected into D800 - D8FF. (This preempts the OS Floating Point Utilities, which should be no problem). The only other constraint is the required "Parallel Device ROM Data Table" in the first 29 bytes (D800 - D81C), as described in section 2.6, and in reference 10.6 section 2.0.

3.2 Private RAM

The SPC Firmware needs to maintain some private data storage, for status and control records, and data buffers. If at all possible, this must fit into the 128 byte private RAM at D100, to avoid problems of competing for use of user RAM.

The following is a tentative allocation of this 128 bytes, with some similarities to the Atari 850 Handler's private data structure. See reference 10.9, page 30,31 for a description of these data structures. A current listing of the private memory allocation is listed in reference 10.10.

Bytes	Description
64	Serial Port concurrent I/O buffers
2	Serial Port block output buffers
10	Serial Port data buffer pointers
12	Serial Port Handler status/control
2	Serial Port Device Driver status/control
2	Printer Device Driver Status/Control:
16	SPC Interrupt Vectors
20	spare

128	total

3.3 Buffers

The SPC Firmware must maintain a default input buffer and output buffer for the port supporting Concurrent I/O. There may also be a user-supplied buffer for Concurrent I/O input buffer, with a pointer and size passed in the IOCB.

3.4 Scratchpad

There are three places where small amounts of scratchpad RAM are available.

First, the 128 bytes of RAM on the SPC 6532 can be allocated by the SPC Firmware at will. A tentative allocation of these bytes is listed in reference 10.10.

Second, there are 4 spare bytes at the end of each IOCB reserved for the use of the corresponding device handler. Also, there are two bytes in the ZIOCB (ICSPRZ, #2C,D) which are used by CIO before and after handler calls.

Third, there is a block of 13 bytes in zero page used by the SID bus handler, which may be used by an SPC low-level device driver, since these drivers preempt and replace SID. In particular, buffer pointers BUFRL0,HI and BFENLO,HI, at #32 - #35, may be handy because they are already in page zero.

4.0 Initialization

The SPC Firmware will be called by a JSR instruction to location DB19. This code must return by an RTS instruction.

The initialization code has three tasks: set up any internal variables for the SPC Firmware, including the default settings of the serial port handler; initialize the SPC hardware, and patch the "R" serial port handler into the system handler table HATABS. See reference 10.6, section 4.0 for a description of the Parallel Device Initialization interface, reference 10.1, section 9, for details of installing device handlers, and reference 10.8 chapter 3 for a description of the serial device handler default settings.

4.1 Initialize hardware

The SPC hardware should be initialized with all interrupt sources inactive, and with the ports set to match the default conditions as described above. The 6532 RAM/IO/Timer chip should be set with both parallel ports as inputs. Both 6551 serial ports should have a program RESET by a write (anything) to the corresponding address (D1A1, D1A5). The Parallel printer port should default to data outputs active w/ASCII null (0) and control outputs inactive (write \$F0 to D1AC). Hardware initialization should include self-testing if practical.

4.2 Install Default Interrupt Handlers

The SPC device has seven sources of interrupts: two serial ports, two Ringin ports, a Parallel port ACKnowledge input, and a 6532 chip with a Timer and a PA7 edge detector. At initialization, the private RAM jump vectors should all be set to point to an RTS instruction, not an RTI.

4.3 Initialize Serial Port Handler and both device drivers

The SPC Firmware should default to particular specification and conditions, such as serial port data rates, translation modes, etc. Where possible, these should match those of the Atari 850 device. (Consult reference 10.8 for details).

4.4 Install "R" Serial Port Handler

An entry for the R Handler should be installed into the system device handler table HATABS, beginning at 031A (see section 2.5.1 for a description). At the first vacant entry (0, 0, 0 bytes), substitute the ASCII character "R", followed by the low and high bytes of the system's resident Generic Parallel Device Handler (GPDH). These bytes are: \$52, \$8F, \$E4.

5.0 Serial Port Device Handler

The SPC Firmware must provide a serial port handler, since the resident OS firmware does not include one. This handler will be called by CIO or by BASIC (or by an aggressive applications program).

In all cases, commands, data, and pointers to data are passed through the registers and the zero page IO control block (ZIOCB, \$20 - \$2F), and status, data, and pointers to data are returned in the same places. The general conditions for device handler calls and returns are discussed in reference 10.1, section 5, in exhaustive detail. On a call, the A register may contain single output data byte, the X register contains the index to the source I/O control block (IOCB) (the original IOCB, not the copy in page zero). ($X = 16 \times \text{the IOCB number}$). The Y register contains a tentative return status of \$B2 = "non-existent device". On return, the A register may contain a single input data byte, the X register contains the same IOCB index, and the Y register should contain the actual operation status: 1 for OK, and some negative number for an error. (A list of error codes is contained in reference 10.1 Appendices B and C).

There are six standard entry points for a device handler: OPEN device, CLOSE device, PUT data, GET data, check STATUS, and SPECIAL. The application will set up the IOCB, including a command byte. (A list of standard command codes is included in reference 10.1, appendix A. the serial device specific commands are listed in the equate section of reference 10.9, and in section 5.6 of this document.) CIO will partially decode that command byte, to chose which entry point to use. For the SPC firmware, which resides in bank selected ROM, CIO will call the OS resident Generic Parallel Device Handler first, through its entry points, which will in turn call the firmware for each parallel device until the firmware on one card recognizes and processes the I/O request.

On entry at each of its six entry points, the SPC firmware serial device handler must first recognize a request for itself. If it handles the request, it must return with the processor carry flag (C) set; otherwise, it should return with the C flag clear.

At this writing, it appears that the way to recognize an intended call is a follows: On OPEN, follow the IOCB index (in reg X) back to the Handler I.D. (ICHID, \$340+X) entry, which in turn points to the device name in HATABS (\$31A+ICHID), and test for a match. If the OPEN entry code saves the X register for future use, the other five entry points can test the X register directly to recognize a call. In this case, the CLOSE entry code should clear the saved device I.D.

Some handler entry points have additional command interpreting to do, using the AUX1 and AUX2 bytes (ICAX1Z,2Z locations \$2A,B in the ZIOCB). The SPECIAL command entry point must also further decode the command byte (ICCMDZ location \$22 in

the ZIOCB) to identify device specific commands:

command name		description
#20	FORCE	force short block output
#22	CONTROL	operate serial port control lines
#24	CONFIGURE	configure serial port
#26	TRANSLATE	configure serial data translation
#28	CONCURRENT	set up and start concurrent I/O

The specifications for each of the ten serial device handler commands are described in sections 5.1 through 5.11. Each section will list the Inputs, the handler Tasks required, and the Outputs, with references where helpful.

NOTE: Due to an artifact of the design of the original Atari BASIC cartridge, the PUT handler can be entered from BASIC through a vector in the IOCB, bypassing the CIO utility. In this case, the IOCB is not copied down into the ZIOCB, nor is the ZIOCB copied back to the IOCB on exit. This has three consequences:

- 1) The PUT byte handler cannot count on the ZIOCB on entry.
- 2) If the PUT handler modifies the first 12 bytes of the ZIOCB, it should do so to the originating IOCB as well, and vice versa..
- 3) It is recommended to save the IOCB index (in reg X on entry) into ICIDNO, location #2E, in the ZIOCB, as CIO does.

In the following sections, the inputs and outputs from the IOCBs will be referenced to the ZIOCB for all handlers except PUT.

5.1 OPEN Serial Port

5.1.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICHIDZ	\$0020	Handler I.D. (index into Handler Table)
ICDNOZ	\$0021	Device number
ICAX1Z	\$002A	Auxilliary byte #1 bit 3 = Write/Output request bit 2 = Read/Input request bit 0 = Concurrent I/O request

5.1.2 Tasks:

- 1) Test for device = "R" (see section 5.0). If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (\$82), and Carry set.
- 3) Test for port already open. If so, return with port already open error status (\$96), and Carry set.
- 4) Test for requests for Read, or Write I/O (Aux1). If not Read (bit 2) or Write (bit 3), then return no permissions granted error code (\$84), with Carry set.
- 5) set up serial port to default conditions.
- 6) return success status (1) and carry set.

5.1.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "

5.2 CLOSE Serial Port

5.2.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICHIDZ	\$0020	Handler I.D.
ICDNOZ	\$0021	Device number

5.2.2 Tasks:

- 1) Test for device = "R" or valid Handler I.D. If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (\$B2), and Carry set.
- 3) If concurrent I/O is in process, finish emptying transmit buffer buffer, shut down device, report status: either success (1) or device timeout (\$BA), and return with carry set.

5.2.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "

5.3 GET BYTE

5.3.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICHIDZ	#0020	Handler I.D.
ICDNDZ	#0021	Device number

5.3.2 Tasks:

- 1) Test for device = "R" or valid Handler I.D. If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (#82), and Carry set.
- 3) Test for Read permission (from OPEN command, see section 5.1). If not, return with write only error status (#83), and carry set.
- 4) Test for Concurrent Input (from OPEN and START CONCURRENT commands, see sections 5.1 and 5.11). If not, return with error status (#9A), and carry set.
- 5) Wait for and fetch next input character from concurrent I/O interrupt handler.
- 6) For each input byte, count time in seconds, and check for device timeout (nominal 30 seconds). If device timeout, return with device timeout error status (#8A) with carry set.
- 7) Check parity on input character, as specified under TRANSLATION command (see section 5.10), and log error status for query by STATUS command (see section 5.5). (NOTE: 6551 chip does parity checking, see section 8.3).
- 8) Translate input character, as specified under TRANSLATION command (see section 5.10).
- 9) Return input character in A register, return success status (1), and set carry flag.

5.3.3 Outputs

Label	Address	Description
C	flag	performed operation flag
A	reg	returned input data character
Y	reg	returned operation status
ICSTAZ	#0023	" " "

5.4 PUT BYTE

5.4.1 Inputs

Label	Address	Description
A	reg	"put" output data character
X	reg	index to originating I/O Control Block
ICHID	\$0340+X	Device I.D.
ICDNO	\$0341+X	Device number

NOTE: PUT can be entered from BASIC without first entering CID, which copies the IOCB to the ZIOCB.

5.4.2 Tasks:

- 1) Test for device = "R" or valid Handler I.D. If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (\$82), and Carry set.
- 3) Test for Write permission (from OPEN command, section 5.1). If not, return with read only error status (\$87), and with carry set.
- 4) Translate output character, as specified by TRANSLATE command (see section 5.10).
- 5) Determine parity, as specified by TRANSLATE command (see section 5.10). (NOTE: 6551 chip does parity generation, see section 8.3).
 - 6a) if not concurrent mode, pass character to low-level device driver using a Device Control Block (DCB).
 - 6b) if concurrent mode, pass character to interrupt handler buffer.
 - 7a) if not concurrent I/O, set the device timeout to 30 seconds, and pass that to the device driver through DTIMLO and DTIMHI in the DCB.
 - 7b) For concurrent I/O, for each output byte, count time in seconds, and check for device timeout (nominal 30 seconds). If device timeout, return with device timeout error status (\$8A) with carry set.
 - 8a) if not concurrent I/O, copy status from DCB, and return with carry set.
 - 8b) if concurrent I/O, return with success status (1) and carry set.

5.4.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "
DDEVIC	\$0300	Device Bus I.D. = \$50, \$51 for R1, R2
DUNIT	\$0301	Unit number = 1, 2 for R1, R2
DCOMND	\$0302	Bus Command = \$57 ('W') for write
DBUFLO,	\$0304	Data Buffer Pointer: low byte = ^OUTBi,
DBUFHI	\$0305	high byte = \$D1
DBYTLO,	\$0308	Data Buffer Size: low byte = 1
DBYTHI	\$0309	high byte = 0
DTIMLO,	\$0306	data transfer timeout limit (seconds)
DTIMHI	\$0307	low byte and high byte
OUTBi	tbd	output byte

5.5 Check STATUS

5.5.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICHIDZ	\$0020	Handler I.D.
ICDNOZ	\$0021	Device number

5.5.2 Tasks:

- 1) Test for device = "R" or valid Handler I.D. If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (\$B2), and Carry set.
- 3) Set up Device Control Block to query status from the Low-level Device Driver.
- 4) Compute status as necessary (e.g. buffer sizes, as described in section 5.5.3 below.)
- 5) Load status into DVSTAT device status table.
- 6) Return with success operations status (Y = 1), and carry set.

5.5.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "
DDEVIC	\$0300	Device Bus I.D. = \$50, \$51 for R1, R2
DUNIT	\$0301	Unit number = 1, 2 for R1, R2
DCOMND	\$0302	Bus Command = \$53 ('S') for STATUS
(set by low-level device driver)		
DVSTAT	\$02EA	Error flags (bits):
	7	RX data framing error
	6	RX data overrun error
	5	RX data parity error
	4	RX data buffer overflow error
	3	Illegal option combination attempted
	2	External device not ready
	1	TX block data transfer error
	0	error on command to device driver

(if not concurrent I/O mode):

```
-      $02EB  Serial port Status line conditions:
          7  DSR, Data Set Ready
          6  DSR bit has changed flag
          5  CTS, Clear to Send
          4  CTS bit has changed flag
          3  CRX, (aka DCD) carrier detect
          2  CRX has changed flag
        * 1  RINGIN state
          0  RCV, (aka RXD) Received data
             (set to 1 unless BRK detect)
```

NOTE: * RINGIN line is not available from Atari 850.

```
(if concurrent I/O mode):
-      $02EB,C number of characters in Input buffer in
          Concurrent mode (low byte, high byte)
-      $02ED  number of characters in Output buffer in
          Concurrent mode (one byte only)
```

5.6 SPECIAL Command, Entry processing

5.6.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICHIDZ	\$0020	Handler I.D.
ICDNOZ	\$0021	Device number
ICCMNDZ	\$0022	Device Command

5.6.2 Tasks:

- 1) Test for device = "R" or valid Handler I.D. If not, return with Carry flag clear.
- 2) Test for valid device number. If not, return with invalid device number status (\$82), and Carry set.
- 3) Test for valid SPECIAL command (\$20, \$22, \$24, \$26, and \$28). If not, return invalid command error (\$84), with carry set.
- 4) Branch to appropriate Special Command handler, as described in sections 5.7 to 5.11 below.

5.6.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "

5.7 SPECIAL: FORCE Short Block

This command is decoded for compatability with the Atari 850 only; in principle, it is unnecessary with the SPC device. The only work it must do is to validate the command, and report success. A discussion of why this is the case is described in section 5.7.4 below.

5.7.1 Inputs

Label	Address	Description
-------	---------	-------------

X	reg	index to originating I/O Control Block
---	-----	--

ICDNOZ	\$0021	Device number
--------	--------	---------------

1) Test for port OPEN. If not, return not-open error status (\$85) and carry set.

2) Test for write permission, as set by OPEN command (see section 5.1). If not, return read-only error status (\$87) and carry set.

3) Return success status (1), and set carry.

5.7.3 Outputs

Label	Address	Description
-------	---------	-------------

C	flag	performed operation flag
---	------	--------------------------

Y	reg	returned operation status
---	-----	---------------------------

ICSTAZ	\$0023	"	"	"
--------	--------	---	---	---

5.7.4 Differences from ATARI 850 handler

The Atari 850 handler, which resides in the main computer's RAM, must communicate with the physical 850 device through the Serial Bus. This is shared resource, for which this handler must compete with any other Serial Bus device handler, such as the disk drives, etc; i.e. it is a bottleneck!

The 850 handler copes with this bottleneck in the common way: it maintains an internal buffer for output data, which is flushed to the 850 over the Serial Bus under any of three conditions: a) buffer full, b) <CR> character, or c) FORCE Short Block command. The SPC card doesn't have to compete for the Serial Bus, so it doesn't need to buffer more than a byte of output data before the device driver is called. Therefore, there is no buffer to flush; it is always emptied immediately.

The FORCE Short Block command is included here for compatability with programs written to work with the 850 serial port handler, at the handler level.

5.8 SPECIAL: Serial Port CONTROLS

5.8.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICDNOZ	\$0021	Device number
ICAX1Z	\$002A	Auxilliary byte #1: Port controls: 7 Write DTR line 6 New DTR state 5 Write RTS line 4 New RTS state 3,2 Unused 1 Write XMT (aka TXD) line 0 New XMT (aka TXD) state

5.8.2 Tasks:

- 1) Transfer Control Specifications from IOCB input (ICAX1Z) to DCB (Device Control Block)
- 2) call low-level device driver with CONTROL command (\$53) in DCMND
- 3) Report Status from DCB to ZIOCB, and return with carry set.

5.8.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "
DDEVIC	\$0300	Device Bus I.D. = \$50, \$51 for R1, R2
DUNIT	\$0301	Unit number = 1, 2 for R1, R2
DCOMND	\$0302	Bus Command = \$41 ('A') for CONTROL
DAUX1	\$030A	Auxilliary Byte (from ICAX1)

5.9.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "
DDEVIC	\$0300	Device Bus I.D. = \$50, \$51 for R1, R2
DUNIT	\$0301	Unit number = 1, 2 for R1, R2
DCOMND	\$0302	Bus Command = \$42 ('B') for CONFIGURE
DAUX1	\$030A	Serial Port Configuration (from ICAX1)
DAUX2	\$030B	Line Monitoring Requests (from ICAX2)

5.10 SPECIAL: TRANSLATE Serial Data

In PUT and GET operations, the serial port handler must translate characters between their Atari computer internal representation (ATASCII) and their external representation (typically ASCII). The primary differences are as follows:

- 1) ASCII is seven bits. ATASCII uses the eighth bit for inverse video.
- 2) ASCII uses the first 32 characters as control codes. ATASCII uses many of them as graphics characters.
- 3) ASCII uses <CR> (\$0D) as a line delimiter. ATASCII use an EOL (End-Of-Line, \$9B).

The PUT and GET handlers have three options:

- 1) Light Translation (default):
 - a) MSB on output and input set to zero
 - b) EDL substituted for CR on input
 - c) CR (and LF) substituted for EOL on output
- 2) Heavy Translation: same as above, plus:
 - d) ATASCII \$00 - \$1F, \$7D - \$FF suppressed for output
 - e) ASCII \$00 - \$1F replaced by substitution character in input (see ICAX2Z below)

5.10.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICDNOZ	\$0021	Device number
ICAX1Z	\$002A	Translation Specifications: 7 -unused- 6 Append <LF> to <CR>, from EOL 5 Translation inhibit 4 Heavy vs Light ATASCII/ASCII translation 3,2 Input Parity: 00 = Ignore parity 01 = check for odd parity, then clear 10 = check for even parity, then clear 11 = clear parity 1,0 Output parity: 00 = Ignore parity 01 = set odd parity 10 = set even parity 11 = set parity bit to 1
ICAX2Z	\$002B	Suppression Character, used to replace illegal ASCII as translated from ATASCII

5.10.2 Tasks:

1) Log Serial Data Translation Specifications, as described in section 5.10.1 above, for use by the PUT and GET handlers.

2) Return success status (1), with carry set.

5.10.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "

5.11 SPECIAL: Start CONCURRENT I/O

Concurrent I/O is initiated by this command. It is halted by either a CLOSE, or by pressing the BREAK key on the keyboard (an IRQ), or by pressing the SYSTEM RESET key (an NMI).

5.11.1 Inputs

Label	Address	Description
X	reg	index to originating I/O Control Block
ICDNDZ	\$0021	Device number
ICAX1Z	\$002A	= 00: specifies default buffer, (use permissions from OPEN requests) /=00: specifies user supplied buffer I/O requests: bit 3 = Write/Output bit 2 = Read/Input bit 0 = Concurrent I/O
ICBALZ,	\$0024	Input Buffer Pointer, low byte,
ICBAHZ	\$0025	
ICBLL,	\$0028	Input buffer size, low byte,
ICBLH	\$0029	

5.11.2 Tasks:

- 1) Test for port OPEN. If not return port-not-open status (\$85), with carry set.
- 2) Test if the other port is already in concurrent I/O mode (there may only be resources for one at a time). If true, report error code (\$99), with carry set.
- 3) Test if concurrent I/O requested (AUX1 or from OPEN). If not, report error status (\$97), with carry set.
- 4) Test for user supplied input buffer: if ICAX1 = 0, then use default buffer. Set up input buffer for user supplied or default 32 byte input buffer.
- 5) Test for output request. If so, set up output buffer pointers.
- 6) Point appropriate SPC serial port interrupt vector at the corresponding Concurrent I/O interrupt handler.
- 7) Intercept main IRQ vector, to filter for BREAK Key interrupts from POKEY chip.
- 8) Return success (1), with carry set.

5.11.3 Outputs

Label	Address	Description
C	flag	performed operation flag
Y	reg	returned operation status
ICSTAZ	\$0023	" " "

6.0 Low Level Device Drivers

The SPC Firmware must provide two sets of low-level device drivers: one set for the two serial ports, and one set for the parallel printer port. The primary entry point for these drivers is a single JMP instruction located at D805, as specified in section 2.6 above. The primary specification for these device drivers is that they respond to calls by the FIO/SIO utility identically to the way the ATARI 850 Interface Module responds to SIO commands (exceptions will be noted).

Commands, data, and pointers are passed to the device drivers in the Device Control Block (DCB), described in section 2.5. Status, data, and pointers are returned in the DCB and in the Device status buffer, described in section 2.7. The specific uses of these data structures is described in sections 6.1 - 6.8 below.

At entry, the Low-level Device Driver code must first examine the DCB to recognize appropriate calls and device commands and execute them, and reject any others. The device driver entry processing is described in detail in section 6.1. The individual device command specifications are described in sections 6.2 to 6.8. The appropriate device I.D.s (DDEVIC \$0300) and commands (DCOMND \$0302) are listed here:

DDEVIC	DCOMND	Description	Section
\$50	\$41	Serial CONTROL command	6.2
\$50	\$42	Serial CONFIGURE command	6.3
\$50	\$53	Serial STATUS command	6.4
\$50	\$57	Serial WRITE command	6.5
\$50	\$58	Serial STREAM command	6.6
\$40	\$53	Printer STATUS command	6.7
\$40	\$57	Printer WRITE command	6.8

The serial port device handlers, resident in the same ROM, may bypass the FIO utility and the resulting Entry processing, and enter the individual drivers directly.

6.1 Low Level Device Driver Entry processing

6.1.1 Inputs

Label	Address	Description
DDEVIC	\$0300	Device I.D. (From Serial Bus I.D)
DUNIT	\$0301	Device Unit number
DCOMND	\$0302	Device Command

6.1.2 Tasks

1) Test the device I.D. for the printer code \$40 (\$4X), or the serial port code \$50 (\$5X). If neither, return with carry clear.

2) If printer, test if a printer is attached and ready (i.e. is printer "Error" inactive). If so, ignore printer call: return with carry clear.

3a) For printer, test for device unit number of 2, for P2. If not, ignore printer call: return with carry clear.

3b) For serial port, test for device unit number of 1 or 2, for R1 or R2. If not, ignore serial port call: return with carry clear.

4a) For printer, test for valid command (\$53 or \$57). If valid, jump to appropriate printer device driver: see section 6.7 or 6.8.

4b) For serial port, test for valid command (\$41, \$42, \$53, \$57, or \$58). If valid, jump to appropriate printer device driver: see sections 6.2 through 6.6.

5) If invalid command, return invalid-command status (\$84), with carry set.

6.1.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned error status

6.2 Serial Port Control Device Driver

6.2.1 Inputs

Label	Address	Description
DUNIT	\$0301	Device unit number
DAUX1	\$030A	Auxilliary Inputs: port controls: 7 Write DTR line 6 New DTR state 5 Write RTS line 4 New RTS state 3,2 Unused 1 Write XMT (aka TXD) line 0 New XMT (aka TXD) state

6.2.2 Tasks

- 1) Locate ACIA chip from DUNIT
- 2) Test DAUX1 bit 7: if set, write bit 6 to DTR.
- 3) Test DAUX1 bit 5: if set, write bit 4 to RTS
- 4) Test DAUX1 bit 1: if set, write bit 0 to TXD (XMT)
- 5) Return success (1) in DSTATS, with carry set.

6.2.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status

6.3.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status

6.4 Serial Port Status Device Driver

6.4.1 Inputs

Label	Address	Description
DUNIT	\$0301	Device unit number
DSTATS	\$0303	Device Status: D6 = read request

6.4.2 Tasks

1) Test DSTATS for Read request. If not, return error status in DSTATS, with carry set. (Device NACK, \$8B).

2) Copy serial data status of specified serial port into DVSTAT (\$02EA), in format specified in section 6.4.3 below.

3a) If not Concurrent I/O mode, sample and copy serial port status line conditions into DVSTAT+1 (\$02EB), in format specified in section 6.4.3 below.

3b) If Concurrent I/O mode, compute and copy number of characters in input buffer into DVSTAT+1 and DVSTAT+2 (\$2EB,C) low byte/high byte, and copy number of characters in output buffer into DVSTAT+3 (\$2ED) one byte.

4) Return success (1) in DSTATS, with carry set.

6.4.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status
DVSTAT	\$02EA	Error flags (bits): (reference)
	7	RX data framing error (7.7.3)
	6	RX data overrun error (7.7.3)
	5	RX data parity error (7.7.3)
	4	RX data buffer overflow error (7.7.3)
	3	Illegal option combination (?)
	2	External device not ready (6.5.3)
	1	TX block data transfer error (6.5.3 or 7.8.3)
	0	error on command to device driver

```
(if not concurrent I/O mode):  
- $02EB Serial port Status line conditions:  
  7 DSR, Data Set Ready  
  6 DSR bit has changed flag  
  *1 5 CTS, Clear to Send ( = 1)  
  *1 4 CTS bit has changed flag ( = 0)  
  3 CRX, (aka DCD) carrier detect  
  2 CRX has changed flag  
  *2 1 RINGIN state  
  *3 0 RCV (aka RXD) Received data ( = 1)
```

NOTES:

```
*1 6551 cannot sense CTS directly, default to 1.  
*2 RINGIN line is not available from Atari 850.  
*3 6551 cannot sense RCV directly, default to 1.
```

```
(if concurrent I/O mode):  
- $02EB,C number of character in Input buffer in  
  Concurrent mode (low byte, high byte)  
- $02ED number of characters in Output buffer in  
  Concurrent mode (one byte only)
```

6.5 Serial Port Write Device Driver

6.5.1 Inputs

Label	Address	Description
DUNIT	\$0301	Device unit number
DSTATS	\$0303	Device Status: D7 = write request
DBUFLO, DBUFHI	\$0304 \$0305	pointer to output buffer, low byte, high byte
DBYTLO, DBYTHI	\$0308, \$0309	output buffer size, low byte, high byte
DTIMLO, DTIMHI	\$0306, \$0307	device timeout, low byte, high byte

6.5.2 Tasks

- 1) Test DSTATS for write request. If not, return error (Device NACK \$8B) status in DSTATS, with carry set.
- 2) Transfer contents of data buffer of size specified by DBYTLO,DBYTHI, pointed to by DBUFLO,DBUFHI to serial port specified by DUNIT, until buffer is empty: no bytes left or EOL character found.
- 3) On each byte transfer, if character cannot be transferred within timeout period specified by DTIMLO,DTIMHI, exit with Timeout Error (\$8A) in DSTATS, and with carry set.
- 4) Return success (1) in DSTATS, with carry set.

6.5.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status

6.6 Serial Port Stream Device Driver (Concurrent I/O)

This device driver is a dummy, for compatibility with the Atari 850. Since the SPC does not use the Atari serial bus at all, the Stream function cannot be implemented.

However, just what this code should do for compatibility is unspecified. The choices are: truthfully report an error, for invalid command (\$84), or fake success (1).

At this writing, this driver should return an error. The streaming function was intended to be called by the serial device handler (section 5.11), which in this case does not need to send a stream command to the driver. Any 850-dependent software which tried to use the 850 in stream mode at the driver level would not work. The error condition will notify the user of a problem.

6.6.1 Inputs

Label	Address	Description
		none

6.6.2 Tasks

1) Return an invalid-command status (\$84), with carry set.

6.6.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status

6.7 Parallel Port Status Device Driver

DSTATS \$0303 Device status

6.7.1 Inputs

Label	Address	Description
-------	---------	-------------

DSTATS	\$0303	Device Status: D6 = read request
--------	--------	----------------------------------

6.7.2 Tasks

1) Test DSTATS for Read request. If not, return error status in DSTATS, with carry set. (Device NACK \$8B).

2) Sample and transfer printer flags into DVSTAT (\$02EA), as specified in 6.7.3.

3) Transfer last Write DAUX1 byte into DVSTAT+1.

4) Save the nominal printer timeout low byte (\$30) into DVSTAT+2, with a 00 into DVSTAT+3. (from 850 controller listing..)

5) Return success (1) in DSTATS, with carry set.

6.7.3 Outputs

Label	Address	Description
-------	---------	-------------

C	flag	operation performed flag
---	------	--------------------------

DSTATS	\$0303	returned operation status
--------	--------	---------------------------

DVSTAT	\$02EA	Printer Flags: 7 = 1 not 40 column printer 2 = printer BUSY 1 = data error (TBD) 0 = command error
--------	--------	--

DVSTAT+1	\$02EB	Last AUX1 sent on Write command (see 6.8.2)
----------	--------	---

DVSTAT+2,	\$02EC	Printer device timeout, low byte = \$1E,
DVSTAT+3	\$02ED	high byte = 0

6.8 Parallel Port Write Device Driver

6.8.1 Inputs

Label	Address	Description
DSTATS	\$0303	Device Status: D7 = write request
DBUFLO,	\$0304	pointer to output buffer, low byte,
DBUFHI	\$0305	high byte
DTIMLO,	\$0306	device timeout, low byte,
DTIMHI	\$0307	high byte
DBYTLO,	\$0308,	output buffer size, low byte,
DBYTHI	\$0309	high byte
DAUX1	\$030A	AUX1, 'N' (\$4E) for normal printing

6.8.2 Tasks

- 1) Test DSTATS for write request. If not, return error (Device NACK \$8B) status in DSTATS, with carry set.
- 2) Save DAUX1 for reporting by the Status driver (see section 6.7.3).
- 3) Test DAUX1 for normal printing. If not, return error (TBD) status in DSTATS, with carry set.
- 4) Transfer contents of data buffer of size specified by DBYTLO,DBYTHI, pointed to by DBUFLO,DBUFHI to printer port, until buffer is empty: no bytes left or EOL character found.
- 5) On each byte transfer, if printer does not respond within specified timeout period, exit with Timeout Error (\$8A) in DSTATS, and with carry set.
- 6) Return success (1) in DSTATS, with carry set.

6.8.3 Outputs

Label	Address	Description
C	flag	operation performed flag
DSTATS	\$0303	returned operation status

7.0 Interrupt Handlers

A resident OS ROM Parallel IRQ Handler will intercept any interrupts from Parallel Bus devices, including the SPC device. It will locate the interrupt source by reading the request location PDVI (\$D1FF), and mask it with the RAM register PDIMSK (\$0249). It will then call the Firmware for that device through the specified location D808, by a JSR instruction. The A and X registers will have been saved, and will be restored upon exit by an RTS instruction.

There are seven sources of interrupts on the SPC device. The IRQ handler entry code must identify the source, and then jump through a vector in the SPC private RAM to code which handles that specific interrupt. On initialization, these seven vectors should point to code which disables the source of interrupt, and returns by RTS.

The seven default interrupt handlers are described in sections 7.2 through 7.6.

The SPC Firmware will contain three interrupt handlers associated with Concurrent I/O: serial data input, serial data output, and Break key. These will be described in sections 7.7 through 7.9. (as of this date, 30 May 84, these sections are TBD.)

7.1 Entry processing

7.1.1 Inputs

Label	Address	Description
PPSTAT	D180	Parallel Port Status bit 4 = -Acknowledge input bit 1 = Serial Port #2 Ringin bit 0 = Serial Port #1 Ringin
TPSTAT	D185	6532 Timer/PA7 Status bit 7 = Timer underflow bit 6 = PA7 Edge Detect
SPSTAT1	D1A1	Serial Port #1 Status bit 7 = Interrupt request
SPSTAT2	D1A5	Serial Port #2 Status bit 7 = Interrupt request

NOTE: the A and X registers are save before entry, and will be restored after an RTS. The Y register must be saved and restored if used.

7.1.2 Tasks

- 1) Test TPSTAT: if Bit 7, JMP through TIMERV.
if Bit 6, JMP through PA7V.
- 2) Test SPSTAT1: if bit 7, JMP through SERP1V.
- 3) Test SPSTAT2: if bit 7, JMP through SERP2V.
- 4) Test PPSTAT: if bit 4, JMP through PPACKV.
if bit 1, JMP through RING2V.
if bit 0, JMP through RING1V.
- 5) If none of above, return by RTS

7.1.3 Outputs

-no outputs-

7.2 Serial Ports Interrupts

This section describes the Default serial port handler. It should only be entered under error conditions i.e. a serial port is generating unintended interrupts.

7.2.1 Inputs

Label	Address	Description
SPSTAT1	D1A1	Serial Port #1 Status bit 4 = TX buffer empty bit 3 = RX buffer full
SPSTAT2	D1A5	Serial Port #2 Status bit 4 = TX buffer empty bit 3 = RX buffer full

7.2.2 Tasks

- 1a) Entry for Serial port #1, set X reg = 0.
- 1b) Entry for Serial port #2, set X reg = 4.

- 2) Determine interrupt source from status register.
 - 3a) if bit 4, disable TX interrupts, and reset TX interrupt flag: set bit 3 and clear bit 2 of the command register; transmit null character (00) to the TX data register.
 - 3b) if bit 3, disable RX interrupts, and reset RX interrupt flag: clear bit 1 of the command register; read and discard the contents of the RX data register.

- 4) return by RTS instruction

7.2.3 Outputs

Label	Address	Description
SPCMD1	D1A2	Serial Port #1 Command Register bit 3,2 = TX controls 1,0 = TX ON, IRQ OFF 0,1 = TX ON, IRQ ON bit 1 = RX interrupt enable
SPCMD2	D1A6	Serial Port #1 Command Register bit 3,2 = TX controls 1,0 = TX ON, IRQ OFF 0,1 = TX ON, IRQ ON bit 1 = RX interrupt enable
SPDAT1	D1A0	Serial Port Data
SPDAT2	D1A4	Serial Port Data

7.3 Serial Port Ringin Interrupts

7.4 Parallel Port ACKnowledge Interrupt

This is the Default interrupt handler for RS232 Ringin input interrupts, which should be called only if this is an error.

This is the Default interrupt handler for the Printer port ACKnowledge input interrupt, which should be called only if this is an error.

7.3.1 Inputs

Label	Address	Description
PPSTAT	D180	Parallel Port Status bit 4 = -Acknowledge input bit 1 = Serial Port #2 Ringin bit 0 = Serial Port #1 Ringin

7.3.2 Tasks

- 1) Locate appropriate enable bit to clear from entry.
- 2) Clear corresponding IRQ enable bit.
- 3) return by RTS instruction.

7.3.3 Outputs

Label	Address	Description
PPCSHD	D15C	Parallel Port Control Shadow
PPCTRL	D1AC	Parallel Port Control bit 2 = Printer ACKnowledge IRQ enable bit 1 = Serial Port #2 Ringin IRQ Enable bit 0 = Serial Port #1 Ringin IRQ Enable

7.5 6532 Timer Interrupt

This is the Default interrupt handler for the 6532 Timer, which should be called only if this is an error.

7.5.1 Inputs

Label	Address	Description
-------	---------	-------------

-no inputs-

7.5.2 Tasks

- 1) Disable Timer interrupts, by reading state.
- 2) return by RTS instruction.

7.5.3 Outputs

Label	Address	Description
-------	---------	-------------

TIMEDI	D194	Read TIMER, and Disable timer IRQ
--------	------	-----------------------------------

7.6 6532 PA7 Interrupt

This is the Default interrupt handler for the 6532 PA7 edge detector, which should be called only if this is an error.

7.6.1 Inputs

Label	Address	Description
-------	---------	-------------

-on inputs-

7.6.2 Tasks

- 1) Disable PA7 interrupt by writing to the control register PA7NDI (D194).
- 2) return by RTS instruction.

7.6.3 Outputs

Label	Address	Description
-------	---------	-------------

PA7NDI	D194	6532 Port PA7 Interrupt control: Negative edge, Disable Interrupts
--------	------	---

8.0 Hardware Programming Model

All of the Serial/Parallel Card (SPC) I/O devices and private memory are mapped into addresses D100 - D1AC. These are implemented with one 6532 RAM/I/O/TIMER, two 6551 ACIAs, and a pair of LSTTL latches. The address map is shown below:

Label	Address	Bits	Description
8.1	6532 RAM		
	D100 - D17F		128 bytes SPC private RAM
8.2	6532 Parallel I/O and Timer		
FPSTAT	D180(R)		SPC Status Port: (6532)
		7	+SLCT printer Select input
		6	-ERROR printer Error input
		5	+BUSY printer Busy input
		4	-ACK printer Acknowledge input
		3	+PE printer Paper End input
		2	+RXD2 serial port 2 RX Data
		1	RING2 serial port 2 RINGIN input
		0	RING1 serial port 1 RINGIN input
PPSDDR	D181		Status Port Data Direction Register
FPDIN	D182(R)		Printer Data Port (6532)
FPDDR	D183		Printer Data Port Data Direction Register
TFSTAT	D185(R)		6532 Interrupt Status Register
		7	Timer Interrupt
		6	PA7 Edge Interrupt
		5-0	unused, =0
	D184-7(W)		6532 PA7 Edge Detect Interrupt Controls:
PA7NDI	D184		IRQ OFF, Negative Edge
PA7PDI	D185		IRQ OFF, Positive Edge
PA7NEI	D186		IRQ ON, Negative Edge
PA7PEI	D187		IRQ ON, Positive Edge
TIMEDI	D194(R)		Read 6532 Timer, IRQ Disabled
	D194-7(W)		Set 6532 Timer, IRQ Disabled, modulo:
TIM1DI	D194		559nsec
TIM8DI	D195		4.47usec
TIM64DI	D196		35.76usec
TIM1KDI	D197		572.1 usec
TIMEEI	D19C(R)		Read 6532 Timer, IRQ Enabled
	D19C-F(W)		Set 6532 Timer, IRQ Enabled, modulo:
TIM1EI	D19C		559nsec
TIM8EI	D19D		4.47usec
TIM64EI	D19E		35.76usec
TIM1KEI	D19F		572.1 usec

8.3 6551 Serial Ports

Address	Bit	Description
D1A0-3		Serial Port 1:
SPDATA1	D1A0(R/W)	Serial Data (TXD out, RXD in)
SPRES1	D1A1(W)	Serial Port Programmed Reset
SPSTAT1	D1A1(R)	Serial Port Status Register
	7	IRQ Interrupt Request
	6	DSR Data Set Ready
	5	DCD Carrier Detect
	4	TX Data Empty
	3	RX Data Full
	2	RX Overrun Error
	1	Rx Framing Error
	0	Rx Parity Error
SFCMD1	D1A2(R/W)	Serial Port Command Register
	7,6	Parity specification: 00 = Odd parity, RX & TX 01 = Even parity, RX & TX 10 = TX parity mark (1), RX disabled 11 = TX parity space (0), RX disabled
	5	Parity enable
	4	Echo Mode (? set to 0)
	3,2	Transmit control: 00 = TX OFF, TX IRQ OFF, RTS OFF 01 = TX ON, TX IRQ ON, RTS ON 10 = TX ON, TX IRQ OFF, RTS ON 11 = TX BRK, TX IRQ OFF, RTS ON
	1	RX Data Full IRQ disable (enable = 0)
	0	DTR Data Terminal Ready 0 = RX OFF, DTR OFF 1 = RX ON, DTR ON
SPCTL1	D1A3(R/W)	Serial Port Control Register
	7	Stop Bits: (0 = 1 bit, 1 = 2 bits)
	6,5	Word Length: 00 = 8 bits, 01 = 7 bits, 10 = 6 bits, 11 = 5 bits
	4	RX clock source (set to 1 = internal)
	3-0	Baud Rate Selector: 0000 = 16x RXC pin 1000 = 1200 baud 0001 = 50 baud 1001 = 1800 baud 0010 = 75 baud 1010 = 2400 baud 0011 = 110 baud 1011 = 3600 baud 0100 = 134.6 baud 1100 = 4800 baud 0101 = 150 baud 1101 = 7200 baud 0110 = 300 baud 1110 = 9600 baud 0111 = 600 baud 1111 = 19200 baud

8.3 Serial Ports (continued)

	Address Bits	Description
	D1A4-D1A7	Serial Port 2:
SPDATA2	D1A4	Serial Data
SPSTAT2	D1A5	Status(Read)/Reset(Write)
SFCMD2	D1A6	Command Register
SPCTL2	D1A7	Control Register

all bit descriptions are identical to Serial Port 1

8.4 LSTTL Parallel Ports

PPDOUT	D1A8(W)	Printer Port Data Output
PPCTRL	D1AC(W)	Printer Port Control Output
	7	-STROBE Printer Port Data Strobe
	6	-INIT Printer Initialization Command
	5	-AUTO FEED XT Printer Automatic Paper Feed
	4	-SLCT IN Printer DC1/DC3 code disable
	3	-PDOE Printer Data Output Enable
	2	+ACKIEN Printer Acknowledge IRQ Enable
	1	+RG2IEN Serial Port Ringin 2 IRQ Enable
	0	+RG1IEN Serial Port Ringin 1 IRQ Enable

9.0 Differences from the ATARI 850

9.1 Limitations

- 1) Only two serial ports, R1 and R2.
- 2) 45.45 and 56.875 baud missing (first version).
- 3) Streaming/concurrent I/O NOT available directly through Serial Bus (i.e. command #28 is an error).
- 4) 6551 ACIA cannot report RXD (RCV) or CTS pins directly, as called for in STATUS device driver.

9.2 Improvements

- 1) 25-pin standard RS-232 serial ports.
- 2) Full set of control and status lines on both ports.
- 3) RINGIN additional status on both serial ports.
- 4) Block I/O mode unnecessary on output: each byte is sent as soon as the handler gets it. As a consequence, the FORCE Short Block Mode command (#20) is a dummy, for compatibility with the 850.
- 5) No restrictions on word size vs baud rate.
- 6) 19.2K baud available.
- 7) Concurrent I/O does not compete with Serial Bus I/O.
- 8) Serial Handler does not use main memory.
- 9) 25 pin IBM compatible printer port.
- 10) Bi-directional parallel port (not supported at device driver level).

9.3 Possible Improvements

These would be subject to ROM and RAM space restrictions on technical feasibility, and to time and budget restrictions on available development resources.

- 1) Restore 45.45 and 56.875 baud using processor to generate serial data clock under timer interrupt control.
- 2) Modify serial port device drivers to respond to P2 device I.D., for serial printers.
- 3) Add a new Printer (P) Handler, with new features:
 - a) Concurrent Output, using default 40 byte buffer in main memory, or user supplied buffer specified in IOCB.
 - b) Reporting of additional printer port status lines: Paper End, ACK, Select.
 - c) Control of additional printer port control lines: INIT, Auto Feed, Select.
 - d) Printer Port input (TBD)
 - e) Redirect printer data to serial port.
- 4) Modify Serial Handler so that a second SPC device could implement R3 and R4.

- 10.0 References
- 10.1 ATARI Personal Computer System Operating System User's Manual, and Hardware Manual, C016555, Atari 1980.
- 10.2 DE RE ATARI, A Guide to Effective Programming, Atari Program Exchange (APX) 1981.
- 10.3 ATARI OS Program Listing, Atari 1979.
- 10.4 ATARI XL OS Program Listing, Atari 1984.
- 10.5 ATARI XL Expansion System Technical Specification (Preliminary) 4/13/84.
- 10.6 The Software Implementation of Parallel Device Handlers and Drivers, Revision 4, 3/15/84.
- 10.7 Your Atari Computer, Osborne 1982.
- 10.8 ATARI 850 Interface Module Operator's Manual, C015953, Atari 1980.
- 10.9 ATARI 850 Resident Handler 1.0A Program Listing, Atari.
- 10.10 SPC Firmware Private Memory Allocation, Decuir, May 1984.