A.N.A.L.O.G. #15, January 1984

TRANSPORTING ATARI COMPUTER PROGRAMS TO THE ATARI 5200

by Claus Buchholz

Annotations by Dan Boris 4/2002

When Atari designed the 5200 "Supersystem" as a successor to the aging 2600 VCS, they made use of the state-of-the-art hardware they put into their 400/800 home computers. As a result, the systems are quite similar. The differences are great enough, however, that transporting programs from one system to the other requires some effort.

The 5200 is a single-board machine with four controller jacks, a cartridge slot, an I/O expansion connector and a power/RF cable. It shares the same VLSI chips with the 400/800, except for the 6520 PIA (joystick/parallel I/O ports). The other chips (ANTIC, GTIA, POKEY and the 6502 CPU) are in the 5200, although some of their registers are in different memory locations than those in the 400/800. Also, some of the registers serve slightly different purposes. Let's look at each section of the hardware in turn.

When a specific register is mentioned in the article, the name is taken from the Hardware Manual in the 400/800 Technical Reference Notes. With the exceptions listed in this article, the Hardware Manual applies also to the 5200.

6502 CPU

Although it is a standard 6502 from the programmer's view, Atari has reworked this chip to save four support chips. Those four chips mediate access of the system bus by the 6502B and ANTIC in the 400/800, but that function is built into the 6502C in the 5200. Atari also uses the 6502C in the new XL series of computers.

16K RAM

The 5200 contains 16K bytes of RAM addressed from $0000 to $3FFF, just as in an unexpanded 400. The memory circuit is nearly identical to that in the 400, except it uses 4516s, a 5V-only version of the 4116 16K-bit dynamic RAM.

The 5200 monitor program reserves locations $0000 through $0018 and $0200 through $021B for shadows and RAM vectors. And, of course, page $01 is reserved for the 6502 stack. The rest of the RAM is available to the cartridge program.

ANTIC

This is the same ANTIC chip used in the 400/800. Since ANTIC shares the address bus with the CPU and has no chip select input, its registers reside in the same place in the 5200's memory as in the 400/800's, page $D4.

The 5200 has no System Reset key, so bit 5 of NMIST is useless and the 5200's interrupt handler ignores it.

GTIA

The GTIA and its registers perform the same functions in the 5200 and 400/800, except as noted below. The registers, however, reside at page $C0 in the 5200's memory, not at $D0 as in the 400/800.

The trigger inputs, TRIG0 through TRIG3, are wired to the controller ports, one to a port. The bottom button on either side of the leftmost controller zeroes the TRIG0 register when pressed, and likewise for the other ports.

The bits in CONSOL, the 400/800's console switch port (START, OPTION, SELECT and speaker), are used as outputs in the 5200, Bit 3, the 400/800's speaker control can still be toggled in the 5200 to produce sounds through the TV speaker. Bit 2 controls the pots in the joystick controllers. It must be set high to enable the pots.

Bits 1 and 0 select which controller port is to be active at one time. 00 selects port #1 (the leftmost), 01 selects #2, 10 selects #3, and 11 selects #4. The trigger buttons and pots are independent of this selection; it applies only to the keypads and top side buttons on the controllers.

POKEY's registers are all addressed at page $E8 in the 5200 as opposed to $D2 in the 400/800. Its functions are unchanged, however, except for two.

The eight pot inputs used for paddles in the 400/800 are wired to the 5200's controller ports, two to a port. Each controller has an analog joystick, using one pot to sense horizontal position and a second pot for vertical position. The even pots (POT0-POT6) give the horizontal positions of the joysticks and the odd pots give the vertical positions. The position values range from 1 to 228; the maximum readings are to the right for the horizontal pot and at the bottom for the vertical pot.

Figure 2: Pinout for the 5200 controllers.

Pin	Function
1	Keypad -- right column
2	Keypad -- middle column
3	Keypad -- left column
4	Start, Pause, and Reset common
5	Keypad -- third row and Reset
6	Keypad -- second row and Pause
7	Keypad -- top row and Start
8	Keypad -- bottom row
9	Pot common
10	Horizontal pot (POT0, 2, 4, 6)
11	Vertical pot (POT1, 3, 5, 7)
12	5 volts DC
13	Bottom side buttons (TRIG0, 1, 2, 3)

14	Top side buttons
15	0 volts -- ground

... 5200 to read the keypad keys to the one controller that is selected by bits 1 and 0 in CONSOL. Only four lines are used, though, so only bits 1 through 4 of KBCODE are valid. Table 1 gives the keycode for each key on the controller. The top side buttons on the selected controller act like the 400/800 shift keys and also cause a BREAK-key interrupt, if that's enabled. Bit 0 of SKCTL, the debounce enable bit need not be set in the 5200.

A couple of notes about the 5200 keypad:

1. Not only is the debounce enable not needed, if you do enable it you won't be able to read any keycodes back from the POKEY.

2. Since the low bit and high bit of the keyboard scan counter are not used any keypress will actually return 4 different keycodes during each scan cycle. For this reason it's always a good idea to mask bits 1-4 before using them.

3. SKCTL bit 2 (last key still pressed) does not behave as expected on the 5200. I have not yet determined it's actual behavior.

4. SKCTL bit 1 enables the keyboard scanning. Note that this also enabled/disables the POT scanning.

KBCODE

Key	Bits	Keypad code
none	0000	$FF
#	0001	$0B
0	0010	$00
*	0011	$0A
Reset	0100	$0E
9	0101	$09
8	0110	$08
7	0111	$07
Pause	1000	$0D
6	1001	$06
5	1010	$05
4	1011	$04
Start	1100	$0C
3	1101	$03
2	1110	$02
1	1111	$01

POKEY's serial I/O lines are used in the 5200, but they are wired to the I/O expansion connector, an edge connector hidden behind a small door in the rear of the 5200. This connector allows for more hardware registers addressed at page $E0, and for peripherals using the serial port. See Figure 3 for the pinout of this connector. Its existence demonstrates Atari's original plans to expand the 5200 system.

Top		Bottom	
	Pin	Pin	
+5V DC	1	36	+5V DC
Audio Out (2 port)	2	35	Not connected
Ground	3	34	Ground
R/W Early	4	33	Not connected
Enable E0-EF	5	32	D7
D6	6	31	D5
D4	7	30	D3
D2	8	29	D1
D0	9	28	Ground
IRQ	10	27	A0
Ground	11	26	A1
Serial Data In	12	25	A2
Serial In Clock	13	24	A3
Serial Out Clock	14	23	A4
Serial Data Out	15	22	A5
Audio In	16	21	A6
A14	17	20	A7
System Clock	18	19	A11

Figure 3: Expansion Connector

ROM

The 5200 has a 2K ROM on board which holds the character set and monitor program. The character set, which is an exact copy of the 400/800's set, resides at pages $F8 through $FB, and the monitor sits at $FC through $FF.

The cartridge ROM can be 32K bytes long and resides in memory from $4000 to $BFFF. Figure 4 shows the pinout of the cartridge slot. The two interlock connectors are wired together on a cartridge board. The 5200 uses this as a switch for the cartridge's power connections and as a Reset signal. Therefore, a cartridge may be safely removed or inserted while the 5200 is powered on.

Function	Pin	Pin	Function
DO	1	36	Interlock
D1	2	35	A11
D2	3	34	A12
D3	4	33	A10
D4	5	32	A13
D5	6	31	A9
D6	7	30	Audio In (2 port)
D7	8	29	A8
-Enable 80-BF	9	28	Not Connected
-Enable 40-7F	10	27	A7
Not Connected	11	26	+5 VDC
Ground	12	25	Ground
Ground	13	24	Ground (Video In on 2 port)

Ground (System Clock 02 on 2 port)	14	23	Ground
A6	15	22	A4
A5	16	21	A3
A2	17	20	A1
Interlock	18	19	A0

Figure 4: Cartridge Pinout

Note on cartridge configuration

There is room for two ROMS on a standard 5200 cart. One ROM is enabled by pin 9, address $8000-$BFFF and the other is enabled by pin 10, $4000-$7FFF. If you are building your own carts and want to use a single 8K or 16K EPROM you must connect it to the $8000-$BFFF enable because this is where the reset vector is fetched from. It is possible to use a single 32K EPROM on a cart but it requires a little extra logic circuitry to be added to the cart.

The 5200 monitor program

The 1K monitor program in ROM has three functions: (1) to initialize the system before running the cartridge program, (2) to service interrupts as they occur, and (3) to maintain shadows of some important hardware registers. Recall that the 400/800 Operating System is 10K bytes long and performs the above functions. It also provides peripheral handlers, predefined graphics modes, a screen editor, and floating point math routines. Those utilities do not exist in the 5200.

Table 2 shows the RAM locations used by the monitor for shadows and RAM vectors.

Table 2. 5200 Monitor RAM Locations.

Page Zero Locations

$00	Shadow for IRQEN
$01	Real time clock (high byte)
$02	Real time clock (low byte)
$03	Critical code flag (if non-zero, VBI routine is abbreviated)
$04	Attract mode timer/flag
$05	Shadow for DLISTL
$06	Shadow for DLISTH
$07	Shadow for DMACTL
$08-$10	Shadows for COLPMO through COLBK
$11-$18	Shadows for POT0 through POT7

Page Two Vectors

$200	Immediate IRQ vector
$202	Immediate VBI vector
$204	Deferred VBI vector
$206	DLI vector
$208	Keyboard IRQ vector

$20A Keypad routine continuation vector
$20C BREAK key IRQ vector
$20E BRK instruction IRQ vector
$210 Serial Input Data Ready IRQ vector
$212 Serial Output Data Needed IRQ vector
$214 Serial Output Finished IRQ vector
$216 POKEY Timer 1 IRQ vector
$218 POKEY Timer 2 IRQ vector
$21A POKEY Timer 4 IRQ vector

Upon Reset, the 6502 vectors through $FFFC to the initialization routine. This routine performs the following sequence.

1. Disable maskable interrupts, clear the 6502 decimal flag, and set the stack pointed to $01FF.

2. If the cartridge address $BFFD contains $FF, then jump immediately through then vector at $BFFE (diagnostic cartridge).

3. Zero all hardware registers and page $00, set CHBASE to point to the character set at $F8, and initialize the first six RAM vectors starting at $0200.

4. Set up the Atari logo rainbow display. The cartridge title (20 characters) and copyright year (2 characters) in ANTIC mode 7 display code are taken from cartridge addresses $BFE8 through $BFFD.

5. Enable VBI (Vertical Blank Interrupt) and DLI (Display List Interrupt), and enable key scan.

6. Wait four seconds, then jump through the vector at $BFFE to the cartridge program.

When the 6502 receives a non-maskable interrupt (NMI), it vectors through $FFFA to the NMI handler. The following steps take place:

1. Check NMIST and strobe NMIRES to reset the interrupt status.

2. If a DLI is pending, jump through the DLI vector (initialized to point to the rainbow effect routine).

3. If a VBI is pending, jump through the immediate VBI vector (initialized to point to the VBI routine).

4. Else, return from the interrupt (no System Reset).

A cartridge program can change these vectors to point to its own DLI and VBI routines, if it must. The default VBI routine takes the following action.

1. Push A, X, and Y onto stack, increment the real time clock, and update the attract mode timer.

2. If the critical code flag byte is non-zero, then pop Y, X, and A from the stack and return from the interrupt.

3. Update DLISTL, DLISTH, and DMACTL from their shadows.

4. Maintain the attract mode flag and update the GTIA color registers from their shadows.

5. Update the pot shadows from POT0 through POT7, and strobe POTGO to start another pot scan.

6. Jump through the deferred VBI vector (initialized to point to the end of-interrupt routine, which pops Y, X, and A, and returns from the interrupt).

If maskable interrupts (IRQs) are enabled and one is received, the 6502 vectors through $FFFE to an instruction which jumps through the immediate IRQ vector. That vector is initialized to point to the IRQ routine, which performs the following tasks.

1. Push A and check IRQST.

2. For each of the eight bits in IRQST, check for a pending interrupt. If found, then clear the status bit, update IRQEN from its shadow, and jump through the appropriate IRQ vector.

3. If no interrupt found, then push X and check for a BRK instruction interrupt. If found, then jump through the BRK instruction IRQ vector.

4. Else, pop X and A and return from the interrupt.

The only IRQ vector that is initialized is the keyboard IRQ vector, which points to the keypad read routine. That routine does the following:

1. Push X and Y.

2. Read KBCODE and mask bits 1 through 4.

3. Convert to the keypad code given in Table 1, leaving that code in A.

4. Jump through the keypad routine continuation vector (initialized to point to the end-of-interrupt routine).

Comparing the 5200's monitor vectors to the 400/800's OS vectors, we see that Atari paid no attention to compatibility between the two. This further complicates the task of converting a program from one system to the other.

Transportability

It would not be difficult, given the information in this article, to write a program in two versions, one for the 400/800 and another for the 5200. Nor would it be difficult, given the source code, to convert a finished program from

the 5200 to the 400/800. The reverse is more difficult if the program takes advantage of special features in the 400/800 OS. Otherwise, the only task, aside from redefining some addresses, is to convert the keyboard/joystick input routines from one system to the other.

I acquired the information in this article by dissecting a 5200 and disassembling its ROM. The 400/800 schematics in the Hardware Manual were quite helpful. It is interesting to note the difference between the two machines and to guess Atari's motives for the design differences. But the similarities grossly outweigh the differences, so that a 5200 program can be developed and almost entirely debugged before testing on a 5200. With the addition of an EPROM burner, a 400/800 can be a powerful development system for 5200 programs. An adventuresome hacker can even bypass the EPROM by putting dual-port RAM on the cartridge board and downloading programs from the 400/800 development system into the 5200 for testing.

A.N.A.L.O.G. #16, February 1984

READER COMMENT

5200 Article Update. (ANALOG #15)

Newer releases of the 5200 incorporate some minor hardware changes. Controller ports 3 and 4 have been eliminated, making POT4 through POT7, TRIG2, TRIG3, and bit 1 of CONSOL useless. A few of the connector pins have been redefined. Pin 2 of the I/O expansion connector now carries POKEY's Audio Out signal. Three pins on the cartridge connector have changed to accomodate the new 2600 adapter. The system clock, 02, is output on pin 14, isolated through a diode. An alternate video input is taken from pin 24 and is also isolated through a diode. Pin 30 provides an alternate audio input.

There is space on the newer boards for circuitry for a PAL (European TV standard) version of the 5200. Also, on power- up, the monitor program checks for the PAL version by examining the GTIA register PAL after step 2 of the initialization routine. It also checks the cartridge program for PAL compatibility. The byte at $BFE7 should read $02 if compatible, or $00 if not. This is the only important change to the monitor program. There are some additional hardware changes, but none affects the machine's operation from the programmer's view.