Entex Adventurevision Technical Specs V1.2

By Daniel Boris 12/18/2005

Disclaimer: All the information in this document comes from studying the actual Adventurevision hardware. The only technical document I have is the Intel 8-bit Embedded Controller Handbook. I have made every attempt to assure the accuracy of this information, but there are bound to be errors and omissions in this document. ***Use this information at your own risk.***

1.0 Processor

The Adventurevision (AV) is based on the Intel 8048 microcontroller. The 8048 is clocked by an 11 MHz crystal which is divided by 15 to produce a 733 KHz (1.36us) instruction cycle clock. (Note: This is not as slow as it seems since each 8048 instruction takes only one or two cycles.) The 8048 has 64 bytes of internal RAM, and 1K of internal ROM that contains the system BIOS. The 8048 has two, 8-bit I/O ports, an internal timer/counter, an interrupt input (which is not used in the AV), and two single bit testable inputs.

1.1 I/O port 1

All the pins (P10-P17) on this port are used as outputs. The function of each pin as follows:

P1.0, P1.1: RAM bank switch
These pins are used to select which of the four, 256 bytes external RAM banks to use.

P12: BIOS enable
This pin is used to enable or disable the 8048's internal ROM. When set to 0 the internal BIOS ROM is enabled from $000-$3ff. When set to 1 the beginning of the cartridge ROM appears at $000-$3ff.

P1.3..P1.7: Controller read. See section 5.0.

1.2 I/O port 2

P2.0..P2.3: Address bus A8..A11.
P2.4..P2.7: Sound and video control

2.0 Cartridge

The AV cartridges are basically 4K 2532 EPROMS sealed in a special case. The pin outs of the cartridge socket are as follows:

1	A7	24	+5v
2	A6	23	A8
3	A5	22	A9
4	A4	21	Vpp
5	A3	20	Output Enable
6	A2	19	A10
7	A1	18	A11
8	A0	17	D7
9	D0	16	D6
10	D1	15	D5
11	D2	14	D4
12	Ground	13	D3

The cartridge appears in program memory space from $0 - $FFF. Bit P1.2 is used to swap addresses $0-$3FF between the cartridge and the internal BIOS.

3.0 External RAM

Besides the RAM that is internal to the processor there are four banks of 256 bytes of RAM external to the processor. P1.0 and P1.1 are used to select one of the four banks. The memory can be read and written to using the MOVX command.

4.0 Video

The AV has a built in video display system. This system is composed of a vertical column of 40 LEDs, and a continuously spinning mirror. The light from the LEDs bounces off the mirror to form the display that the player sees.

4.1 The Mirror

The mirror is driven by a belt connected to a DC electric motor which is run directly from the DC input voltage to the system. The motor runs at approximately 450 rpm (7.5 rotations per second). The mirror has 2 sides which produces 2 video frames per rotation thus giving the system a display rate of 15fps.

4.2 Sync

Each end of the mirror has a plastic tab that passes through a photo interrupter which signals to the processor the start of each video frame. The state of the photo interrupter can be read via the T1 input on the processor. When the interrupter is blocked, T1 will go low, and when it’s not blocked it will go high.

4.3 LEDs

The LED display has five, 8-bit registers that control the 40 LEDs. Each bit either turns an LED on when the bit is low or off when the bit is high. The register address is controlled by P2.5, P2.6 and P2.7 as follows:

P2.5	P2.6	P2.7	LED numbers (starting from the top)
1	0	0	1 – 8
0	1	0	9 – 16
1	1	0	17 – 24
0	0	1	25 – 32
1	0	1	33 – 40
0	1	1	Unused
1	1	1	Unused

The actual write to the LED registers occurs when there is a read from external memory. So if register one is selected, and $55 is read from memory, this will cause $55 to be written to LED register one. The reason for this odd arrangement is to increase the speed at which data can be transferred to the display, since all the data can be written to external RAM between video frames, the rapidly read back directly into the LED registers.

In the BIOS video routine, after all five registers have been written to, the BIOS sets P2.4 high. I assume this is used to latch in the LED data.

4.4 Video memory

The AV does not have any dedicated video memory, but the BIOS display routine does treat part of the external RAM as video memory. Bytes 6-255 of memory pages one, two and three are used by the BIOS to draw a single frame of video. This makes a total of 750 bytes of display memory, which gives a resolution of 150 vertical lines of 40 pixels (5 bytes) each.

5.0 Game Controls

The system has a 4 position joystick and two sets of four buttons on either side of the joystick The 2 sets of buttons are wired together so the right button one and the left button one are the same, etc.

The controls are read through processor port P1, bits 3 to 7. The following table shows the bit pattern when each button is pressed.

	P1.3	P1.4	P1.5	P1.6	P1.7
Button 1	1	0	0	1	1
Button 2	1	0	1	0	1
Button 3	0	1	1	1	1

Button 4	1	0	1	1	0
Stick Up	1	0	1	1	1
Stick Down	1	1	0	1	1
Stick Right	1	1	1	0	1
Stick Left	1	1	1	1	0

6.0 Sound

The sound in the AV is created by a National Semiconductor COP411L Single-Chip Microcontroller. The COP411 contains 512 bytes of internal ROM, and 128 bits of internal RAM.

The clock for the COP411 is generated by an RC circuit which produces a nominal instruction clock frequency of 52.6Khz. Note that due to the nature of this clock circuit the frequency can actually vary up to +/- 15%, which will directly affect the frequency of the sound.

The sound processor interfaces to the main processor via I/O ports P2.4 to P2.7. The reset pin of the sound processor connects to a latch on the main board which has bit zero of the data bus as its data input. The clock input of the latch comes from the display LCD board and appears to be activated when P2 is set to $C0 which is outside the range of the LED registers.

This is the routine in the BIOS that is used by the games to control the sound chip:

Outputs 0xC0 onto the P2 port. Since 0xC0 is an address that the LED module doesn't use it's possible that this has something to do with enabling the reset latch as described above.

```
03A9: MOV A,C0
03AB: OUTL P2,A
```

Set RAM bank one so the commands later don't mess up the working RAM

```
03AC: MOV A,01
03AE: OUTL P1,A
```

Strobes 0 into the sound reset latch thus holding the sound processor in reset. Zero is written to memory, then read back so I assume the read triggers the latch through the LED module.

```
03AF: MOV R0,00
03B1: CLR A
03B2: MOVX @ R0,A
03B3: MOVX A,@ R0
```

Hold sound processor reset for 33 processor cycles.

```
03B4: MOV R2,14
03B6: DJNZ R2 B6
03B8: NOP
```

Strobes 1 into the sound reset latch this releasing the sound processor reset.

```
03B9: MOV A,01
03BB: MOVX @ R0,A
03BC: MOVX A,@ R0
```

Get the sound command from the R1 register.

```
03BD: MOV A,R1
```

Write the command to the sound processor. Since only the upper four bits of P2 are connected to the processor only the upper four bits of the command are written here.

```
03BE: OUTL P2,A
```

Wait for 55 processor cycles.

```
03BF: MOV R2,26
03C1: DJNZ R2 C1
03C3: NOP
```

Write the lower four bits of the sound command to the sound processor.

```
03C4: SWAP A
03C5: OUTL P2,A
```

Wait for 44 processor cycles.

```
03C6: MOV R2,21
03C8: DJNZ R2 C8
```

Set back to RAM bank 0 and clear the P2 outputs.

```
03CA: CLR A
03CB: OUTL P1,A
03CC: OUTL P2,A
03CD: RET
```

Each sound command is composed of two, four bit parts. I will call them the command value and the data value. There are three types of commands, control, pure tone, and sound effects.

6.1 Control Command

When the command value is 0, the data value is written into the sound control register in the COP411's RAM. The COP's RAM does not get cleared on a reset so this value will stay in memory until it is changed, or the system is powered off. The sound control registers effects each of the other sound commands in a different way. Issuing the control command will also turn off any sounds that are currently playing.

6.2 Pure Tones

When the command value is 0xE or 0xF, the sound chip generates a pure tone. The frequency is determined by the data value as show in the following table. Freq Nominal is the sound frequency based on the nominal clock frequency. Freq Scale is an equal tempered musical scale that corresponds pretty closely to the nominal frequencies especially when you take into account the range of error of the base frequenct. Note is the musical note that corresponds to each of the scale frequencies.

Value	Freq Nominal (Hz)	Freq Scale(Hz)	Note
0	239.23	233	A#
1	253.03	247	B
2	268.53	262	C
3	286.04	277	C#
4	302.48	294	D
5	320.92	311	D#
6	337.38	330	E
7	360.49	349	F
8	381.38	370	F#
9	404.85	392	G
A	424.44	415	G#
B	453.72	440	A
C	478.46	467	A#
D	506.07	494	B
E	537.05	523	C
F	572.08	554	C#

The sound control register controls the duration of the sound.

The sound is played back in two segments, and the duration of these segments is controlled by bit 0 of the sound control register as follows:

Bit 0	First Segment Duration (s)	Second Segment Duration (s)
0	0.117	0.240
1	0.046	0.104

The durations shown here are averages; the actual durations vary slightly from one frequency to the next.

Bits 1 and 2 of the control registers control the playback of these segments as follows

Bit 1	Bit 2	
0	0	Both segments are played at low volume
1	0	First segment is played at high volume, second is played at low volume
0	1	Both segments are played at high volume
1	1	Both segments are played at high volume

Bit 3 controls the looping of the sound. If this bit is 0, the two segments are played once then the sound shuts off until the next sound command is issued. If this bit is 1, the two segments are played repeatedly until the next sound command is issued.

6.3 Sound Effects

When the command value is between $1 and $D the chip generates sound effects. The data value for these commands has no effect on the sound playback.

7.0 Expansion Connector

The expansion connector is a single sided, 25 pin card edge connector on the side on the main PCB. The pins are as follows:

1	T0
2	~RD
3	~PSEN
4	~WR
5	ALE
6	D0
7	D1
8	D2
9	D3
10	D4
11	D5
12	D6
13	D7
14	P1.7
15	P2.0

16	P1.6
17	P2.1
18	P1.5
19	P2.2
20	P1.4
21	P2.3
22	P1.3
23	PROG
24	+5v
25	GND

It appears that the expansion connector was designed to support the 8243 I/O expander that is a companion chip to the 8048 processor. This explains the PROG pin, which is used when program the 8048's internal ROM, but also is part of the communication bus needed to support the 8243. There are also two routines in the BIOS used to write to I/O ports 4 and 5 which are only available when the 8243 is used.

8.0 BIOS

The 8048 in the AV contains a 1K BIOS that has many useful function. The BIOS is enabled when the P12 output is set to 0, and can be accessed in the address range $000-$7FF.

8.1 Startup

When the AV starts up the BIOS is enabled and the 8048 jumps to location $0 in ROM. The first two instructions in the BIOS will cause a jump to location $800 which is where the cartridge should start. Locations $802 - $80B in the cartridges are used as call back addresses by various routines in the BIOS. If you plan to use these routines, you should put a jump instruction at $800 that jumps to the actual start of your cartridge program.

8.2 Vector Table

Location $03 - $2A contain 20 jump instructions that jump to the start of each of the BIOS routines. This table would have allowed the makers of the system to modify the details of the BIOS without breaking any existing programs as long as they only used the jump table and did not jump directly into the BIO.

8.3 Routines

This section provides information on a number of the BIOS routines. I have not figured out all the routines yet, so I have only documented the ones that I understand the function of.

Vector: $03
BIOS address: $36
Function: Copy data from RAM to video display

Description:
This routine is used to copy the data in RAM banks 1, 2, and 3 to the LED display. It handles synchronization and the proper timing of the data writes to create 150x40 pixel display. The sequence of events in this routine is:

1. Increment RB0($3F)
2. Wait for the video sync pulse
3. Move bytes $6 - $A from RAM page 1 to the first column of the display
4. Continues moving 5 bytes at a time from RAM page 1 to the display.
5. Repeat process for RAM bank 2 and 3 starting at byte $6 in both banks.

Vector: $05
BIOS Address: $71
Function: Moves graphics data to the display RAM

Description:
The purpose of this routine is to move 8 pixel high graphics data from the cartridge ROM to the video display. It can also do collision detection to see if the new graphics data collides with anything already in memory.

The routine is called with a pointer in R1 to a data structure used to control the graphics move. The location of the structure must be aligned on a 8 byte boundary, but R1 can point anywhere in the structure. So if R1 points to $E2, the structure will be read starting at $E0.

The data structure has the following format:

$00 - Unknown
$01 - Initial data source pointer
$02 - Number of pixels to shift image down the screen
$03 - Starting RAM bank of destination (if >3 then routine will exit immediately)
$04 - Starting RAM location of destination

1. Read a byte of graphics data by calling the routine at $02 in the cartridge which should return with the graphics byte in A. This callback routine can use RB1(R2) as a pointer to where to get the next byte of data. If the callback returns $FF, then the move is done and the routine exits.

2. Shift the image down the screen by the number of pixels specified in $02 in the data structure.

3. Check for collision with data already in the video RAM. If a collision is detected, set bit 0 of $3B to 1.

4. Logically OR new graphics data with data already in RAM.

Vector: $07
BIOS Address: $F0
Function: Similar to the routine at $71 with a few differences.

Vector: $09
BIOS Address: $1A0
Function: *Unknown*

Vector: $0B
BIOS Address: $2B9
Function: *Unknown*

Vector: $0D
BIOS Address: $361
Function: *Unknown*

Vector: $0F
BIOS Address: $324
Function: *Unknown*

Vector: $11
BIOS address: $2E2
Function: *See description*
Description:

This routine was apparently meant to fill the video RAM with $00, thus turning on all the display pixels, but it appears to have a bug that would prevent it from working. The end of the routine is missing a RET instruction so it will run directly into the next routine which fill the video RAM with $FF.

Vector: $13
BIOS address: $2EF
Function: Fills video RAM with $FF

Description:

This routine fills the video RAM with $FF which turns off all the pixels.

Vector: $15
BIOS address: $2B
Function: Clears internal user RAM
Description:

Sets the internal RAM locations $20-$3F to 0.

Vector: $17
BIOS address: $39A
Function: Two byte BCD add
Descriptions:

Adds the contents of A to the two byte BCD value stored in internal RAM $33,$34.

Vector: $19
BIOS Address: $2CE
Function: *Unknown*

Vector: $1B
BIOS address: $3A9
Function: Writes sound command
Description:

This function writes the sound command stored in R1 to the sound hardware.

Vector: $1D
BIOS address: $2FF
Function: Move data structure used by routine at $71 back to registers
Description:

This function moves the data from the data structure used by the routines at $71 and $F0 into registers. R1 points to the data structure and the structure is returned as follows:

R5 = Bits to shift data
R6 = RAM bank
R7 = RAM pointer

Vector: $1F
BIOS address: $30E
Function: Moves data from registers to the data structure used by routine at $71.
Description:

This function moves the data from the registers to the data structure used by the routines at $71 and $F0 into registers. R1 points to the destination of the data structure and the registers contain the following values:

R5 = Bits to shift data
R6 = RAM bank
R7 = RAM pointer

This routine in the inverse of the one at vector $1D.

Vector: $21
BIOS Address: $31D
Function: *Unknown*

Vector: $23
BIOS Address: $197
Function: *Unknown*

Vector: $25
BIOS address: $3F1
Function: Sets a specific bit in a byte
Description:
Bit number contained in R5 in A is set high

Vector: $27
BIOS address: $3EF
Function: Writes A to port P5
Description:
This routine was probably intended for use with a device connected to the expansion port.

Vector: $29
BIOS address: $3ED
Function: Writes A to port P4
Description:
This routine was probably intended for use with a device connected to the expansion port.