Executive ROM

The Executive ROM is 5KB of ROM arranged as 4096 10-bit values, referred to as "decles", mapped into the Intellivision address range $1000-$1FFF. On the Intellivision II only, the Executive ROM includes an additional 256 decles at addresses $0400-$04FF. On the Intellivision I, two chips hold the EXEC ROM: an RO-3-9502 and an RO-3-9504. On the Intellivision II, these two ROMs were combined into a single RO-3-9506.

The Executive ROM provides the "operating system" for the Intellivision. When the Intellivision is turned on or reset, the CP1610 begins executing instructions from the Executive ROM starting at location $1000. Additionally, when the STIC generates its VBlank Interrupt, the CP1610 jumps to the interrupt-handling subroutine located in the Executive ROM at location $1004. The RO-3-9502 (or RO-3-9506) chip asserts these addresses to the CPU during the IAB Bus Phase. Cartridges can theoretically override these addresses by intercepting the IAB bus phase, asserting the desired address, and remapping IAB to NACT.

Cartridge ROM usually originates at address $5000 and, after completing its startup sequence, the Executive ROM jumps to the start of the program contained in the cartridge at that location. However, prior to entering the startup sequence, the Executive ROM first checks addresses $4800 and $7000 for viable ROM, and if it exists, the Executive ROM skips the normal startup process and begins running the program immediately. This is why some games start their map at $4800, to nearly completely bypass the Executive ROM start-up sequence.

Known Executive ROM Subroutines

```
$1126  The EXEC's Video Blank Routine (VBL)
```

```
$1668  Extend sign from low byte of R0 into full word.
```

```
$169E  Generate a random number between 0 and R0.  Result in R0.
```

```
$1738  Zeros memory.
       R0 = count
       R4 = address
```

```
$1741  Fills memory with value in R1.
       R0 = count
       R1 = value
       R4 = address
```

```
$1777  R1 = start addr.
       Returns:
       R2 = x.pos
       R3 = y.pos
```

```
$17C1  Push registers R1 through R4 onto the stack.
```

```
$1867  Same as $187B, except you can specify an address from where to
       fetch the string data.  The address is incremented to point to
       the character after the null terminator, but is placed in R5.
       R1 = address of string data, which must end with a null
            terminating character ($0).
       R3 = attribute word (same as BACKTAB word format)
       R4 = destination address to write to (i.e. BACKTAB location)
```

```
$187B  Write a string of words to a memory location (such as BACKTAB).
       The string of words must follow the JSR instruction, and must
       end with a null-terminated word (0h).  The R7 register will be
       automatically incremented to start running the instruction
       right after the null-terminated word.
       R3 = attribute word (same as BACKTAB word format)
       R4 = destination address to write to (i.e. BACKTAB location)
```

```
$18AD  Write a numeric string like $18C5, but pad with 0's.
       R0 = numeric value
       R1 = number of digits
       R3 = attribute word (same as BACKTAB word format)
       R4 = destination address to write to (i.e. BACKTAB location)
```

```
$18C5  Write a numeric string, padded with spaces on the left instead of
       0 digits.  R4 is positioned at the begining of where the numeric
       was written to, instead of positioned just after.
       R0 = numeric value
       R1 = number of digits
       R3 = attribute word (same as BACKTAB word format)
       R4 = destination address to write to (i.e. BACKTAB location)
```

```
$1910  Get an R0-digit number from the user, and echo the number colored R2 at address R1.
       Range of R0 is 1-4.
       Note: you need to restore the HANDTAB dispatch table after this call.
```

```
$1A83  Silence the PSG sound channels.
```

```
$1B27  Play a tune starting at the address after the call using the "flute" soundform.
```

```
$1B5D  Play a tune starting at the address after the call using the "organ" soundform.
```

```
$1B95  Possibly triggers a "stock" sound effect.
       Parameter is passed as a string of words after the JSR instruction,
       and is terminated by a null-terminated word. The R7 register will
       be automatically incremented to start running the instruction
       right after the null-terminated word.
```

```
$1BBE  Possibly generates a noise envelope.  The data is passed after the
       JSR instruction, and the R7 register is automatically incremented
       to start running the instruction right after a null-terminated word
       of $02CF (??).
```

```
$1DD8  Square: R0.  Result in R2.
```

```
$1DDC  Multiply: R0 and R1.  Result in R2.
```

```
$1DF8  Divide:  Dividend is R1, divisor is R2.
       Quotient placed in R0, remainder placed in R1.
```

```
$1EC4  Razz for R0 ticks.
```

```
$1ED5  Cheer for R0 ticks.
```

```
$1F1D  Whistle for R0 ticks.
```
