

Color Computer 1/2/3 Hardware Programming

This is a document collecting and detailing hardware programming information for the TRS-80 Color Computer, versions 1, 2, and 3. Although it has some tutorial information in it, it is designed to be a reference.

Compiled and edited by Chris Lomont July 2006, www.lomont.org. Version 0.8. Send comments, corrections, and errors to CoCo3 at the domain above. Please don't repost this on the web, but point to this copy, so eventually all information is corrected and integrated.

This document is compiled from many sources. If you feel this infringes any of your copyrighted material, email me with your material, and I will remove or rewrite from scratch the offending sections.

ALL ADDRESSES AND NUMBERS ARE IN HEX unless in parentheses! Addresses like FFFE(65534) give the decimal in parentheses. 16 bit addresses are CPU address space, 20 bit addresses are in GIME address space. Note that the Memory Mapping Unit maps eight 8K pages from the GIME space into CPU space.

Many sections (marked TODO) need a lot more work, which I will do given time.

DISCLAIMER: All information provided as is etc.

Table of Contents

- Introduction
- CoCo 1/2 topics: PIA, VDG, SAM
- CoCo 3 topics: MMU, GIME
- Notes for various operations
 - Graphics
 - CoCo 1/2 - SAM, PIA
 - Colors, Coding, Storing
 - CoCo 3 GIME
 - Text
 - Memory Mapping
 - CoCo 1/2
 - CoCo 3 GIME MMU
 - Basic Memory Map
 - Colors
 - Keyboard
 - Joystick
 - Mouse
 - Interrupts
 - Sound
 - Casette Info
 - Disk Info
 - Serial I/O Info
 - Cartridge
 - ROM Routines
- Hardware Register Reference
 - PIA FF00-FF3F
 - Disk Controller FF40-FF5F
 - Misc (TODO) FF60-FF8F
 - GIME FF90-FFBF
 - SAM FFC0-FFDF
 - Interrupts FFE0-FFFF
- Detailed Memory Map
- Sources

```

*****
*           Color Computer Hardware Introduction           CoCo 1/2/3   *
*****
This document covers the hardware in the Color Computer, versions 1, 2, and 3,
often called the CoCo1, CoCo2, and CoCo3.

```

The CoCo3 supports the CoCo 1 and 2 hardware in CoCo 1/2 compatibility mode, described elsewhere in this document.

The CoCo runs on a Motorola 6809 chip, details of which are in a different document.

The main hardware interfaces are:

- CoCo 1/2/3:
- PIA - Peripheral Interface Adapter - General hardware Input/Output
 - SAM - Synchronous Address Multiplexer - Determines how data moves
 - VDG - Video Display Generator - Converts RAM to images

CoCo 3 only:

GIME - Graphics Interrupt Memory Enhancement - What it says....

```

*****
*           Color Computer 1/2 Hardware Topics           CoCo 1/2   *
*****
The CoCo 1 and 2 (and 3) have

```

The CoCo 2 has a RAM/ROM mode, and an all RAM mode, selected by SAM control bit TY, accessed from FFDE/FFDF.

32K RAM 0000-7FFF /32K ROM 8000-FFFF or 64K RAM 0000-FFFF (TODO - vectors?)

TODO - overview PIA, SAM, VDG, GIME, MMU, Etc.

PIA - FF00- etc. todo

The SAM performs the following functions:

- Clock generation and synchronization for the 6809E CPU and 6847 VDG
- Up to 64K Dynamic Random Access Memory (DRAM) control and refresh
- Device selection based on CPU memory address to determine if the CPU access is to DRAM, ROM, PIA, etc.
- Duplication of the VDG address counter to "feed" the VDG the data it is expecting
- Divides the internal 4x NTSC freq (14.31818MHz for NTSC) by 4, passes it to the VDG for its own internal timing (3.579545MHz for NTSC).
- Divides the master clock by 16 (or 8 in certain cases) for the two phase CPU clock - in NTSC this is .89MHz (or 1.8MHz if div by 8).

TODO

```

*****
*           Color Computer 3 Hardware Topics           CoCo 3   *
*****
The CoCo 3 supports the hardware of the CoCo 1 and 2, and adds a multifunction
chip, the GIME (TODO).

```

The GIME adds

- Many more graphics and text options.
- New interrupt sources, like timer and keyboard.
- Ability to address more memory (128K in original CoCo 3's, 512K after upgrade. There are other, bigger upgrades available). This is done by paging 8K blocks into the address space used by the CPU, and is handled by the Memory Management Unit (MMU).

TODO

```
*****
*      Color Computer Peripheral Interface Adapters      CoCo 1/2/3      *
*      (PIA) Motorola MC6821 or MC6822                  *
*****
There are two PIA chips, PIA0 and PIA1, each consisting of 4 addresses. Each
PIA has two data registers and two control registers.
```

PIA0 uses addresses FF00-FF03. Data registers FF00 and FF02 are mostly keyboard and printer interfaces, and control registers FF01 and FF03 handle horizontal and vertical sync interrupts and joystick direction.

PIA1 uses addresses FF20-FF23, handling cassette, printer, CoCo 1/2 video modes, audio, and cartridge info.

TODO

```
*****
*      Color Computer Video Display Generator (VDG)      CoCo 1/2/3      *
*      (VDG) Motorola MC6847                            *
*****
The MC6847 VDG is capable of displaying text and graphics contained within a
roughly square display 256 pixels wide by 192 lines high. It is capable of
displaying 9 colors: black, green, yellow, blue, red, buff, cyan, magenta,
and orange. It can generate a few modes: text modes, graphics modes, and
"semigraphics" modes. The semigraphics modes replace each character position
from a text mode with blocks containing pixels.
```

The CoCo is physically wired such that its default alphanumeric display is semigraphics-4 mode.

In alphanumeric mode, each character is a 5 dot wide by 7 dot high character in a box 8 dots wide and 12 lines high. This display mode consumes 512 bytes of memory and is a 32 character wide screen with 16 lines. The internal ROM character generator only holds 64 characters, so no lower case characters are provided. Lower case is instead "simulated" by inverting the color of the character.

Semigraphics is a hybrid display mode where alphanumerics and block graphics can be mixed together on the same screen. See other sections for details.

TODO - add semigraphics 8 12 and 24 modes info?

By setting the SAM such that it believes it is displaying a full graphics mode, but leaving the VDG in Alphanumeric/Semigraphics 4 mode, it is possible to subdivide the character box into smaller pieces. This creates the "virtual" modes Semigraphics 8, 12, and 24. These modes were not implemented on the CoCo 3.

There were several full graphics display modes, -C (for "color") modes and -R (for "resolution") modes. See elsewhere in this document for details.

The 256x192 two-colormode allows "artifact colors" on a NTSC TV, due to limitations of the phase relationship between the VDG clock and colorburst signal. In the white and black colorset, alternating dots bleed together to give red or blue, in effect giving a 128x192 four color mode with red, black, white, and blue. Reversing dot order reverses artifact colors. However, the color formed is somewhat random on RESET, so many games have the player press RESET until the colors are correct for the game. The CoCo 3 fixed this problem, always starting the same, and holding F1 during reset would reverse the colors. Artifacting does not work on the RGB monitors.

TODO

```
*****
*      Color Computer Synchronous Address Multiplexer    CoCo 1/2/3      *
*      (SAM) Motorola MC6883 or SN74LS785              *
*****
The SAM's 16-bit configuration register is spread across 32 memory addresses
(FFC0-FFDF). Writing even bytes sets that register bit to 0, Writing to odd
bytes sets it to 1.
```

The SAM contains a duplicate of the VDG's 12-bit address counter, and usually is programmed to be in sync. Mixing modes between the two results in other possible modes.

TODO

```
*****
*      Color Computer Graphics Interrupt Memory Enhancement CoCo 3      *
*      (GIME) Custom ASIC                                  *
*****
The GIME is a custom ASIC chip designed to replace and extend many parts in
the original CoCo 1 and 2. The main features added are support for more than
64K of memory (128K was the standard, and a 512K upgrade was common), advanced
graphics modes, and more interrupt options. A mode bit in TODO switched
between CoCo and 2 mode and CoCo 3 mode.
```

TODO

```
*****
*      Color Computer Graphics Modes                    CoCo 1/2/3      *
*****
See throughout this document
TODO - table?
```

```
*****
*      Color Computer Text Modes                      CoCo 1/2/3      *
*****
```

The character set available for CoCo 1/2 or CoCo 3 in CoCo 1/2 compatible text mode (WIDTH 32).

On CoCo 3(?) the character set assumes that bit 4 of \$FF22 is set. If that bit is clear, then the characters in the range of 0-\$1F must be replaced by the corresponding characters in the range \$40-\$5F in inverse video.

CoCo 1 and 2, and CoCo3 WIDTH 32 character set:
each entry is hex for Inverted, NonInverted, Text

00 40 @	10 50 P	20 60	30 70 0
01 41 A	11 51 Q	21 61 !	31 71 1
02 42 B	12 52 R	22 62 "	32 72 2
03 43 C	13 53 S	23 63 #	33 73 3
04 44 D	14 54 T	24 64 \$	34 74 4
05 45 E	15 55 U	25 65 %	35 75 5
06 46 F	16 56 V	26 66 &	36 76 6
07 47 G	17 57 W	27 67 '	37 77 7
08 48 H	18 58 X	28 68 (38 78 8
09 49 I	19 59 Y	29 69)	39 79 9
0A 4A J	1A 5A Z	2A 6A *	3A 7A :
0B 4B K	1B 5B [2B 6B +	3B 7B ;
0C 4C L	1C 5C \	2C 6C ,	3C 7C <
0D 4D M	1D 5D]	2D 6D -	3D 7D =
0E 4E N	1E 5E up	2E 6E .	3E 7E >
0F 4F O	1F 5F left	2F 6F /	3F 7F ?

The characters defined by 20-3F are inverse video. Graphics blocks are printed for character values 80-FF.

CoCo 3 high resolution text modes (WIDTH 40,80).
 The character set is repeated for character values 80-FF.

```

00 Ç 10 ó 20 30 0 40 @ 50 P 60 ^ 70 p
01 ü 11 æ 21 ! 31 1 41 A 51 Q 61 a 71 q
02 é 12 Æ 22 " 32 2 42 B 52 R 62 b 72 r
03 â 13 ô 23 # 33 3 43 C 53 S 63 c 73 s
04 à 14 ô 24 $ 34 4 44 D 54 T 64 d 74 t
05 å 15 ø 25 % 35 5 45 E 55 U 65 e 75 u
06 ä 16 ù 26 & 36 6 46 F 56 V 66 f 76 v
07 ç 17 ù 27 ' 37 7 47 G 57 W 67 g 77 w
08 ê 18 ø 28 ( 38 8 48 H 58 X 68 h 78 x
09 ë 19 ö 29 ) 39 9 49 I 59 Y 69 i 79 y
0A è 1A Ū 2A * 3A : 4A J 5A Z 6A j 7A z
0B ì 1B § 2B + 3B ; 4B K 5B [ 6B k 7B {
0C î 1C £ 2C , 3C < 4C L 5C \ 6C l 7C |
0D ß 1D ± 2D - 3D = 4D M 5D ] 6D m 7D }
0E Å 1E ° 2E . 3D > 4E N 5E up 6E n 7E ~
0F Å 1F f 2F / 3F ? 4F O 5F lft 6F o 7F _
  
```

For the hi-res screen modes, each character is 2 bytes, in the format char, attrib, char, attrib, etc... where char is an ASCII character code and attrib is an attribute byte. The attribute byte looks like this:

```

bit
7 flash (l=flash,0 = don't)
6 underline (l=underline, 0 = don't)
5 \
4 - three foreground color bits TODO - see palette later?
3 / (palettes 8-15) FFB8-FFBF
2 \
1 - three background color bits
0 / (palettes 0-7) FFB0-FFB7
  
```

```

*****
* Color Computer Memory Mapping CoCo 1/2/3 *
*****
CoCo 1/2:
32/64K maps see TODO
TODO
  
```

CoCo3:
 The GIME chip can access 512K of memory, yet the 6809 can only access 64K. The barrier is broken by a MMU (Memory Management Unit) which splits the access into 8 blocks of 8K each.

There are two possible memory maps, Map 0 and Map 1, selected by TODO

A memory page is an 8K block. A 128K system has 128/8=16 blocks, numbered hex 30-3F. A 512K system has 64 blocks, numbered hex 0-3F. To place a page in CPU memory for access, write the page number in the appropriate memory select register.

These registers are registers FFA0-FFAF. A write of a page value to the address on the left makes the page visible at the CPU address on the right.

```

Map 0          Map 1
FFA0 -> 0000-1FFF  FFA8 -> 0000-1FFF
FFA1 -> 2000-2FFF  FFA9 -> 2000-2FFF
FFA2 -> 4000-5FFF  FFAA -> 4000-5FFF
FFA3 -> 6000-7FFF  FFAB -> 6000-7FFF
FFA4 -> 8000-9FFF  FFAC -> 8000-9FFF
FFA5 -> A000-BFFF  FFAD -> A000-BFFF
FFA6 -> C000-DFFF  FFAE -> C000-DFFF
FFA7 -> E000-FFFF  FFAF -> E000-FFFF
  
```

A page address is a 6 bit value. When reading these registers, be sure to mask off the top two bits, since they can contain garbage.

Page values for GIME address, default page values on a power up, and default CPU addresses:

Page	GIME Address	CPU Address*	Standard Page Contents
00-2F	00000-5FFFF		512K upgrade RAM, not in 128K
30	60000-61FFF		Hi-Res page #1
31	62000-63FFF		Hi-Res page #2
32	64000-65FFF		Hi-Res page #3
33	66000-67FFF		Hi-Res page #4
34	68000-69FFF		HGET/HPUT buffer
35	6A000-6BFFF		Secondary Stack
36	6C000-6DFFF		Hi-Res text screen RAM
37	6E000-6FFFF		unused
38	70000-71FFF	0000-1FFF	Basic memory
39	72000-73FFF	2000-3FFF	Basic memory
3A	74000-75FFF	4000-5FFF	Basic memory
3B	76000-77FFF	6000-7FFF	Basic memory
3C	78000-79FFF	8000-9FFF	Extended Basic Interpreter
3D	7A000-7BFFF	A000-BFFF	Color Basic Interpreter
3E	7C000-7DFFF	C000-DFFF	Disk Basic Interpreter
3F	7E000-7FFFF	E000-FFFF	Super Basic, GIME regs, I/O, Interrupts

Example: to set GIME memory location 60000 to value 0, you could:

```

ORCC  #$50      SHUT OFF INTERRUPTS - SAVE FOR RESTORE LATER?
LDA   $FFA1    GET THE PAGE FOR THE RESTORE
ANDA  #63      STRIP OFF TOP BITS
PSHS  A        SAVE THE PAGE FOR LATER
LDA   #$30     ACCESS TO PAGE $30 = GIME $60000
STA   $FFA1    MAP PAGE $60000-$61FFF TO LOCATIONS $2000-$3FFF
LDA   #$00     THE BYTE IS 0
STA   $2000    SET THE PROPER BYTE IN CPU SPACE
PULS  A        RESTORE THE PAGE VALUE THAT WAS THERE
STA   $FFA1    MAP ORIGINAL PAGE BACK INTO CPU SPACE
  
```

Notes:

- (1) Unless you know what you are doing, shut off interrupts when changing pages! If you change a page that has an interrupt handler in it, and an interrupt occurs, you will likely crash the computer!
- (2) If you are using the stack, be careful if you page out the stack. Return addresses will crash, and stack values will not likely be the same. There, KNOW WHERE THE STACK IS! In basic, it starts in the 6000-7FFF page.

Here are some simple memory maps. Detailed versions are elsewhere in this document.

```

SYSTEM MEMORY MAP IN GIME ADDRESSES:
RAM   00000 - 7FFFF (512K, 128K CoCo3 is 60000-7FFFF)
ROM   78000 - 7FEFF when enabled TODO?
I/O   FF000 - FFFFF I/O space and GIME regs TODO - unless paged out?
  
```

```

64K PROCESS MAP CPU ADDRESSES:
RAM   0000 - 7FFF (possible vector page FEXX)
I/O   FF00 - FFFF (appears in all pages)   TODO - ROM?
  
```

Note: the Vector Page RAM at 7FE00 - 7FEFF (when enabled), will appear instead of the RAM or ROM at XFE00 - XFEFF. (see FF90 Bit 3) TODO

The 256 top bytes in CPU space contain byte-mapped hardware interfaces, covered elsewhere in this doc.

```

FF00-03  PIA0
FF04-1F  reserved  copies of PIA0
FF20-23  PIA1
FF24-3F  reserved  copies of PIA1
FF40-5F  SCS      see note below - TODO
FF60-7F  reserved  (for current peripherals)
FF80-8F  reserved
FF90-BF  GIME     CoCo3 only TODO
TODO - more

```

A little more detail for the default power on situation:

```

*GIME Address*      *Contents*
00000 - 5FFFF      Unused by Basic; not preset in 128K or smaller systems
60000 - 67FFF      Hires graphics screen
68000 - 69FFF      Hires GET/PUT buffer
6A000 - 6BFFF      Secondary stack area
6C000 - 6DFFF      Hires text screen
6E000 - 6FFFF      Unused by Basic
70000 - 703FF      System RAM
70400 - 705FF      Lowres text screen
*Non Disk System*
70600 - 70BFF      Page 1 - lowres graphics
70C00 - 711FF      Page 2
71200 - 717FF      Page 3
71800 - 71DFF      Page 4
71E00 - 723FF      Page 5
72400 - 729FF      Page 6
72A00 - 72FFF      Page 7
73000 - 735FF      Page 8
*Disk System*
70600 - 70DFF      Disk System RAM
70E00 - Page 1
*Either System*    1 - 8 graphic pages reserved, Basic program start varies.
71200 - 77FFF      or
                    Basic programs, variables, and user ml programs
71400 - 77FFF
78000 - 79FFF      Extended Color Basic
7A000 - 7BFFF      Color Basic
7C000 - 7DFFF      Cartridge or Disk Controller
7E000 - 7FDFE      Super Extended Basic
7FE00 - 7FEFF      Secondary vector table
7FF00 - 7FF3F      PIAs
7FF90 - 7FBFF      GIME in CoCo 3
7FFC0 - 7FFDF      video control, clock, and map type
7FFE0 - 7FFF1      Unused
7FFF2 - 7FFFF      Interrupt vectors

```

TODO - make sure all this in detailed maps

For a very detailed memory map, see elsewhere in this document. TODO
 For details about the hardware interface, see elsewhere in this document.TODO

```

*****
*      Color Computer Colors      CoCo 1/2/3      *
*****

```

Coco 1/2:
 TODO

CoCo 3:
 Palette colors are defined in registers FFB0-FFBF. The format differs depending on if you are in RGB or Composite monitor mode. Mode is selected by setting TODO

The format is explained in the FFB0-FFBF register section.

The table of (hex) colors given below is the conversion used in OS-9 Level II

Monitor Color		Monitor Color	
RGB	CMP	RGB	CMP
00	00	Black	32 23 Medium intensity red
01	12	Low intensity blue	33 08 Blue tint red
02	02	Low intensity green	34 21 Light Orange
03	14	Low intensity cyan	35 06 Cyan tint red
04	07	Low intensity red	36 39 Full intensity red
05	09	Low intensity magenta	37 24 Magenta tint red
06	05	Low intensity brown	38 38 Brown tint red
07	16	Low intensity white	39 54 Faded red
08	28	Medium intensity blue	40 25 Medium intensity magenta
09	44	Full intensity blue	41 42 Blue tint magenta
10	13	Green tint blue	42 26 Green tint magenta
11	29	Cyan tint blue	43 58 Cyan tint magenta
12	11	Red tint blue	44 24 Red tint magenta
13	27	Magenta tint blue	45 41 Full intensity magenta
14	10	Brown tint blue	46 40 Brown tint magenta
15	43	Faded blue	47 56 Faded magenta
16	34	Medium intensity green	48 20 Medium intensity yellow
17	17	Blue tint green	49 04 Blue tint yellow
18	18	Full intensity green	50 35 Green tint yellow
19	33	Cyan tint green	51 51 Cyan tint yellow
20	03	Red tint green	52 37 Red tint yellow
21	01	Magenta tint green	53 53 Magenta tint yellow
22	19	Brown tint green	54 36 Full intensity yellow
23	50	Faded green	55 52 Faded yellow
24	30	Medium intensity cyan	56 32 Medium intensity white
25	45	Blue tint cyan	57 59 Light blue
26	31	Green tint cyan	58 49 Light green
27	46	Full intensity cyan	59 62 Light cyan
28	15	Red tint cyan	60 55 Light red
29	60	Magenta tint cyan	61 57 Light magenta
30	47	Brown tint cyan	62 63 Light yellow
31	61	Faded cyan	63 48 White

TODO

```

*****
*      Color Computer Keyboard      CoCo 1/2/3      *
*****
The keyboard is accessed through PIA0, addresses FF00-FF03. Access is done by
setting (for example) FF00 for input, FF02 for output, sending a signal down
the required bit(s) in FF02, and reading the inputs from FF00. FF00 and FF02
can be reversed if desired.

```

Note bit values are 0 for on, and 1 for off, in reading to keyboard.

Example code: Needs work on how to do keyboard: - clean this up and correct it

```

CLR      $FF03      set FF00 for direction
CLR      $FF00      set for input
CLR      $FF01      set FF02 for direction
lda      #$FF
sta      $FF02      set for output

lda      #%11101111  check only a single column number 4
sta      $FF02      signal columns (in diagram below)
lda      $FF00      read rows (in diagram below)
coma
anda     #$7F       strip bit
cmpa    #%011111011  check single bit 2 - we tested for T key

```

Here is the keyboard matrix. Some entries have multiple keys separated by a /. For example, es/br is the Esc/Break key.

LSB		FF02						MSB	
0	1	2	3	4	5	6	7		
@	A	B	C	D	E	F	G	0	LSB
H	I	J	K	L	M	N	O	1	
P	Q	R	S	T	U	V	W	2	F
X	Y	Z	up	dn	lf	rt	space	3	F
0	!/1	"/2	#/3	\$/4	%/5	&/6	'/7	4	0
(/8)/9	*:/	+;/	</	=/-	>/	? /	5	
enter	clr	es/br	alt	ctrl	F1	F2	shifts	6	MSB

TODO

```
*****
* Color Computer Mouse CoCo 1/2/3 *
*****
Same as joystick programming?
```

TODO

```
*****
* Color Computer Joystick CoCo 1/2/3 *
*****
PIA control registers at FF01 and FF03 set control registers CA2 and CB2,
which in turn select which joystick to read and which axis to read.
```

The 6 most significant bits of FF20 are the digital to analog converter, and any value here is compared to a joystick reading. The high bit of FF00 will be 1 whenever the joystick value exceeds the D/A value. So set FF20 to FC (the highest possible), check the bit, and decrease the value until the bit changes, giving the joystick value.

Since these bits also affect sound, you should mute the CoCo first.

Example:
see http://www.coco25.com/wiki/index.php/Sampling_TODO

The ROM Routine (TODO - name) in the Color BASIC ROM at address \$A00A leaves the joystick values in the four bytes at addresses \$015A through \$015D.

TODO

```
*****
* Color Computer Interrupts CoCo 1/2/3 *
*****
An interrupt is an external event which alters the normal flow of the
microprocessor. There are many possible ways to generate interrupts.
```

The 6809 has 4 hardware interrupts and 3 software interrupts. They are:

Interrupt	Expanded notation	Default use
SWI	Software Interrupt 1	unused in Basic, used in EDTASM
SWI2	Software Interrupt 2	not used? some use in OS9?
SWI3	Software Interrupt 3	not used?
IRQ	Interrupt Request	sound and TIMER functions
FIRQ	Fast Interrupt Request	disk drive access
NMI	Non-Maskable Interrupt	not supported
RESET	Initial power up and RESET button	resetting the machine

When an interrupt fires, the microprocessor first sees if the corresponding bit in the Condition Code (CC) register in the 6809 microprocessor is 0. If it is, the "exception processing" is performed. The microprocessor gets the address to go to from the interrupt vectors, and jumps to the address stored there.

In the CoCo, each of the vectors is in ROM, and cannot be changed. However, the vectors each point to a RAM location that can be changed.

There is a priority to the interrupts, here listed from lowest to highest

Interrupt	registers pushed	Vector	points to:	code at location
SWI3	A,B,X,Y,U,PC,DP,CC	FFF2	???	??
SWI2	A,B,X,Y,U,PC,DP,CC	FFF4	???	??
FIRQ	PC,CC	FFF6	FEF4	LBRA \$010F
IRQ	A,B,X,Y,U,PC,DP,CC	FFF8	FEF7	LBRA \$010C
SWI	A,B,X,Y,U,PC,DP,CC	FFFA	FEFA	LBRA \$0106
NMI	A,B,X,Y,U,PC,DP,CC	FFFC	8C1B	reset code
RESET	none	FFFE		

If there are multiple interrupts, only the highest priority one will be taken.

The FIRQ interrupt is fast in the sense that it does not push many registers on the stack.

For example, if an IRQ occurred, the proper CC bit is 0, and location FFF8-FFF9 was A101, the microprocessor would then start executing code at A101. Interrupts save the listed registers before the interrupt handler is called, and these registers are restored when the Return From Interrupt (RTI) instruction is called at the end of the interrupt routine.

RTI is similar to RTS except that it, in conjunction with the E bit in CC, determines how many registers to pull from the stack.

To disable the interrupts (useful before many changes in the system), use

ORCC	#\$10	disables IRQ
ORCC	#\$40	disables FIRQ
ORCC	#\$50	disables them both

To enable the interrupts, use

ANDCC	#\$EF	enables IRQ
ANDCC	#\$BF	enables FIRQ
ANDCC	#\$AF	enables them both

The GIME chip has the capability of sending interrupts to either the IRQ or FIRQ line. If you are running a 100% ML program once they are set, fine. If you are running a combination program, Basic sets the GIME interrupt registers back to Vertical Border only.

Interrupt sources: TODO

New in CoCo3: Programmable timer, HSYNC, VSYNC, Serial Input Data, Keyboard/Joystick buttons, Cartridge, may be set to IRQ or FIRQ

TODO

```
*****
* Color Computer Sound CoCo 1/2/3 *
*****
```

Mute sound:

```
BEGIN LDA $FF23 Get current Control Register B value of PIA 2
ORA $30 Set CB2 to be an output. (Set bits 4 and 5.)
```

Now the status of bit 3 of Control Register B will control the CB2 line. If bit 3 is low the line will be low. If bit 3 is high the line will be high. Setting CB2 low will mute the CoCo.

```
ANDA #$F7   Clear bit 3 - Mute CoCo
STA  $FF23  Write value back to Control Register B
```

In general programming sound uses the 6 bit D/A.

Also, there was a magazine article early on about 4 channel sound, but I have been unable to find it and analyze it for this section. Perhaps Rainbow or Hot-CoCo?

Another source is the single bit sound:
FF23 bit 2 to 0, (changes FF22 to data dir register)
FF22 to output?,
FF23 bit back to 1 (change FF22 back)
Store sound bits into FF22 (top bit?)

FF03 bit 3	FF01 Bit 3	Sound Source
0	0	DAC
0	1	Cassette
1	0	Cartridge
1	1	No Sound

Another source is the casetterecrder

Anoter source is the cartridge slot?

TODO

```
*****
*          Color Computer Cassette Storage          CoCo 1/2/3      *
*****
Color basic saves a file as a series of blocks, each with 0-255 bytes of data.
Some blocks need preceeded by a leader to establish timing.
```

Each bit is recorded as a single cycle of a sine-wave. A "1" is a single cycle at 2400 Hz, and a "0" is a single cycle at 1200 Hz. Bytes are stored least significant bit first. Bits are recognized when the sine wave crosses from positive to negative, so loudness is not as important as one might expect.

A file consists of:

1. a leader
2. a filename block
3. a 1/2 second gap
4. another leader
5. some number of data blocks
6. an end-of-file block

A leader is just hex 80(128 dec) bytes of hex 55 (binary 01010101).

A block contains:

1. two "magic" bytes (55 and 3C)
2. one byte - block type (00=filename, 01=data, FF=EOF)
3. one byte - data length (00 to FF)
4. 0 to 255 bytes - data
5. one byte - checksum (sum of data, type, and length bytes)
6. another magic byte (55)

Filename blocks have F(15) bytes of data; EOF blocks have zero bytes of data; data blocks have 0-FF bytes of data indicated by length byte.

A filename block contains:

1. eight bytes - the filename
2. one byte - file type (00=BASIC, 01=data, 02=machine code)
3. one byte - ASCII flag (00=binary, FF=ASCII)
4. one byte - gap flag (00=no gaps, FF=gaps)
(The tech manual incorrectly (?) shows 01 as the code for "no gaps")
5. two bytes - machine code starting address
6. two bytes - machine code loading address

There should be no gaps, except preceeding the file, and in case the filename blocks requests gaps, in which case there is a 1/2 second gap and leader before each data block and EOF block.

TODO

```
*****
*          Color Computer Disk Storage          CoCo 1/2/3      *
*****
The disk controller chip is a Western Digital 1793 (or 1773?), and has four
registers at addresses FF48 through FF4B, and one control register at FF40.
The control register enables the drive motors, select lines, and so on.
```

In short:

```
FF40 Control register
FF48 Command/Status register
FF49 Track register
FF4A Sector register
FF4B Data register
```

Write a command into the command register, and read the status in the status register. For reads and writes you need to read/write data to/from the data register. You must do this at the proper speed or the command will fail.

Writing a 0 into the control register turns off the drive motor.

The control register is write only, so Disk Basic keeps a copy of what is written there. If you modify it, you should keep this in mind.

The Track and Sector registers hold current track and sector numbers, reflecting register the current position of the head. Use the Seek command to position the head to the Track you want. Then write the Sector register to tell the controller which sector you want.

Command/Status

Writing into register FF48 gives a command to the disk controller chip. Reading from it tells you the status of the command's execution.

There are four types of disk commands.

- Type I - Restore, Seek, Step, Step In, and Step Out.
- Type II - Read Sector and Write Sector.
- Type III - Read Track, Write Track, and Read Address.
- Type IV - Force Interrupt.

Status bits in the error code are defined as follows, from the 1793 data sheet, and have meaning dependent on the command type. Type IV status codes depend on what command was interrupted.

TODO - clean up, unify with hardware reference

Bit 7 - Not Ready

0 - drive ready

1 - drive not ready.

Type II and III will not execute unless the drive is ready.

Bit 6 - Write Protect

Type I : 0 - not write protected, 1 - write protected;
 Type II/III : Not used on Read Sector or Track. On Write, same as Type I.
 This bit is reset when updated.

Bit 5 - Head Loaded/Record Type/Write Fault
 Type I commands: Head Loaded 1 - head loaded and engaged
 Type II/III commands: Record Type/Write Fault
 Read : indicates the record-type code from the data-field address
 mark. 0 = Data Mark, 1 = Deleted Data Mark.
 Write: indicates a write fault. This bit is reset when updated.

Bit 4 Seek Error/Record Not Found
 Type I : Seek Error - 0 = verified, 1 = track not verified. Reset to 0
 when updated.
 Type II/III : Record Not Found - 0 - ok, 1 - track, sector, or side not
 found. Reset when updated

Bit 3 CRC error (Cyclic Redundancy Check)
 Type I commands: 0 - CRC ok, 1 - CRC failed
 Type II/III commands: If bit 4 set, indicates an error in 1+ ID fields,
 else error in Data field This bit is reset when updated.

Bit 2 Track 00/Lost Data
 Type I commands: Track 00 - 0 = ?, 1 = Read/Write head positioned at
 Track 0.
 Type II/III commands: Lost Data 1 - Computer did not respond to DRQ
 (Data Rrequest) in time and lost data. Bit reset to 0 on update.

Bit 1 Index/Data Request
 Type I commands: Index - 0 - ?, 1 - index mark detected from drive.
 Type II/III commands: Data Request., copy of DRQ output. 1 - DR(Data
 Register) is full on a read or empty on write, reset to 0 when
 updated.

Bit 0 Busy
 0 - not busy
 1 - Command being processed

Color BASIC Disk Format:
 23(35 decimal) tracks, numbered 0-22(34).
 12(18) sectors, numbered 1-12(18).
 Each sector has 100(256 decimal) bytes.
 Total size then 35*18*256 = 161280 decimal bytes.

High level format
 Track 11(17) contains the directory and File Allocation Table (FAT). Other
 tracks split the eighteen sectors into two granules: sectors 1-9 make one
 granule, A-12 make the other. The granules are then numbered 0-43, each
 containing 900 (2304) bytes each. Files are allocated at the granule level,
 so a one byte file still reserves 900(2304) bytes. Track 17 is the middle of
 the disk, so is in a good position for disk activity.

Track 11(17) contains the FAT in sector 2, and the directory on sectors 3
 though B(11). Other sectors are unused.

The bytes in the FAT contain linked lists of file locations on the disk.

The FAT is 44(68) bytes long - one byte for each granule on the disk.
 Values 0-43(67) denote the NEXT granule used by the file. Values between
 C0(192) and C9(201) denote the last granule for the file, and the least four
 significnat bits tell how many sectors of the granule are used. Value FF is
 an unused granule.

A directory sector contains eight entries of 20(32) bytes, making room for
 seventy-two files. A directory is:

- eight bytes for the space-padded filename
 - three bytes for the space padded filename extension
 - one file-type byte
 (0=BASIC program, 1=BASIC data, 2=machine code, or 3=ASCII text)
 - one format byte (0=binary or FF=ASCII)
 - one byte containing the file's first granule
 - two bytes containing the number of bytes used in the last
 sector of the last granule,
 - sixteen unused bytes

TODO

 * Color Computer Serial I/O Info CoCo 1/2/3 *

 The 4-pin DIN connector on the CoCo back is a serial port. This must be
 operated from software; a loop reads and write bits to this port as needed.

Set baud rate (values in decimal):

POKE 150,180 [300 bps]
 POKE 150,88 [600 bps]
 POKE 150,41 [1200 bps]
 POKE 150,18 [2400 bps]
 POKE 150,7 [4800 bps]
 POKE 150,1 [9600 bps]

Others have used assembly routines to support much faster rates.

TODO

 * Color Computer Cartridge Info CoCo 1/2/3 *

By covering pin 8 on the cartridge, ROM-packs could be inserted without them
 starting up. It is EXTREMELY DANGEROUS to insert a ROM-Pack with the CoCo
 switched on. You might cook your CoCo.

Color Computer 1, 2, & 3 Cartridge Connector Definitions
 (* are LOW (0 volts) to activate)

Pin	Signal Name	Description
1	N.C.	(-12 VDC on CoCo 1 and 2)
2	N.C.	(+12 VDC on CoCo 1 and 2)
3	HALT*	Halt input to the CPU
4	NMI*	Non-Maskable Interrupt to the CPU
5	RESET*	Main Reset and Power-up Clear
6	E CLOCK	Main CPU Clock
7	Q CLOCK	Clock which leads E by 90 degrees
8	CART*	Rom-Pak Detection Interrupt
9	+5 VDC	+5 Volts DC (300 mA)
10	DATA 0	CPU Data Bus - Bit 0
11	DATA 1	CPU Data Bus - Bit 1
12	DATA 2	CPU Data Bus - Bit 2
13	DATA 3	CPU Data Bus - Bit 3
14	DATA 4	CPU Data Bus - Bit 4
15	DATA 5	CPU Data Bus - Bit 5
16	DATA 6	CPU Data Bus - Bit 6
17	DATA 7	CPU Data Bus - Bit 7
18	R/W*	CPU Read/Write Signal
19	ADDR 0	CPU Address Bus - Bit 0

20	ADDR 1	CPU Address Bus - Bit 1
21	ADDR 2	CPU Address Bus - Bit 2
22	ADDR 3	CPU Address Bus - Bit 3
23	ADDR 4	CPU Address Bus - Bit 4
24	ADDR 5	CPU Address Bus - Bit 5
25	ADDR 6	CPU Address Bus - Bit 6
26	ADDR 7	CPU Address Bus - Bit 7
27	ADDR 8	CPU Address Bus - Bit 8
28	ADDR 9	CPU Address Bus - Bit 9
29	ADDR 10	CPU Address Bus - Bit 10
30	ADDR 11	CPU Address Bus - Bit 11
31	ADDR 12	CPU Address Bus - Bit 12
32	CTS*	Cartridge (ROM) Select Signal
33	GROUND	Signal Ground
34	GROUND	Signal Ground
35	SND	Cartridge Sound Input
36	SCS*	Spare Cartridge (DISK) Select Signal
37	ADDR 13	CPU Address Bus - Bit 13
38	ADDR 14	CPU Address Bus - Bit 14
39	ADDR 15	CPU Address Bus - Bit 15
40	SLENB*	Input to Disable Internal Devices

TODO

* Color Computer ROM Routines CoCo 1/2/3 *

TODO

* Color Computer Hardware Register Reference CoCo 1/2/3 *

* Color Computer PIA Reference CoCo 1/2/3 *

FF00 (65280) PIA 0 side A data register - PIA0AD CoCo 1/2/3	
Bit 7	JOYSTICK COMPARISON INPUT
Bit 6	KEYBOARD ROW 7
Bit 5	ROW 6
Bit 4	ROW 5
Bit 3	ROW 4 & LEFT JOYSTICK SWITCH 2
Bit 2	ROW 3 & RIGHT JOYSTICK SWITCH 2
Bit 1	ROW 2 & LEFT JOYSTICK SWITCH 1
Bit 0	ROW 1 & RIGHT JOYSTICK SWITCH 1

(1) Todo - keyboard matrix - note

FF01 (65281) PIA 0 side A control reg - PIA0AC CoCo 1/2/3	
Bit 7	HSYNC Flag
Bit 6	Unused
Bit 5	1
Bit 4	1
Bit 3	Select Line LSB of MUX
Bit 2	DATA DIRECTION TOGGLE 0 = FF00 sets data direction 1 = normal
Bit 1	IRQ POLARITY 0 = flag set on falling edge 1=set on rising edge
Bit 0	HSYNC IRQ 0 = disabled 1 = enabled

FF02 (65282) PIA 0 side B data register - PIA0BD CoCo 1/2/3	
Bit 7	KEYBOARD COLUMN 8
Bit 6	7 / RAM SIZE OUTPUT
Bit 5	6
Bit 4	5
Bit 3	4
Bit 2	3
Bit 1	2
Bit 0	KEYBOARD COLUMN 1

(1) Todo - keyboard matrix - note

FF03 (65283) PIA 0 side B control reg - PIA0BC CoCo 1/2/3	
Bit 7	VSYNC FLAG
Bit 6	N/A
Bit 5	1
Bit 4	1
Bit 3	SELECT LINE MSB of MUX
Bit 2	DATA DIRECTION TOGGLE 0 = FF02 sets data direction 1=normal
Bit 1	IRQ POLARITY 0=flag set on falling edge 1=set on rising edge
Bit 0	VSYNC IRQ 0=disabled 1=enabled

Note: FF00-FF03 are repeated through addresses FF04 to FF1F. Thus FF1E is an alias for FF02. Similarly, FF20-FF23 are repeated through FF24-FF3F.

FF20 (65312) PIA 1 side A data register - PIA1AD CoCo 1/2/3	
Bit 7	6 BIT DAC MSB
Bit 6	
Bit 5	
Bit 4	
Bit 3	
Bit 2	6 BIT DAC LSB
Bit 1	RS-232C DATA OUTPUT
Bit 0	CASSETTE DATA INPUT

FF21 (65313) PIA 1 side A control reg - PIA1AC CoCo 1/2/3	
Bit 7	CD FIRQ FLAG
Bit 6	N/A
Bit 5	1
Bit 4	1
Bit 3	CASSETTE MOTOR CONTROL 0=OFF 1=ON
Bit 2	DATA DIRECTION CONTROL 0=\$FF20 data direction 1=normal
Bit 1	FIRQ POLARITY 0=falling 1=rising
Bit 0	CD FIRQ (RS-232C) 0=FIRQ disabled 1=endabled

FF22 (65314) PIA 1 side B data register - PIA1BD CoCo 1/2/3	
Bit 7	VDG CONTROL A/G : Alphanum = 0, graphics = 1
Bit 6	" GM2
Bit 5	" GM1 & invert
Bit 4	VDG CONTROL GM0 & shift toggle
Bit 3	RGB Monitor sensing (INPUT) CSS - Color Set Select 0,1
Bit 2	RAM SIZE INPUT
Bit 1	SINGLE BIT SOUND OUTPUT
Bit 0	RS-232C DATA INPUT

(1) VDG sets graphics modes for CoCo 1/2 and CoCo 3 in compatibility mode. To set a mode, use these bits and the registers FFC0-FFC5. See the Section under FFC0-FFC5 for details and text/graphics mode settings.

FF23 (65315) PIA 1 side B control reg - PIA1BC CoCo 1/2/3	
Bit 7	CART FIRQ FLAG
Bit 6	N/A
Bit 5	1
Bit 4	1
Bit 3	SOUND ENABLE
Bit 2	DATA DIRECTION CONTROL 0 = FF22 data direction 1 = normal
Bit 1	FIRQ POLARITY 0 = falling 1 = rising
Bit 0	CART FIRQ 0 = FIRQ disabled 1 = enabled

Note: FF00-FF03 are repeated through addresses FF04 to FF1F. Thus FF1E is an alias for FF02. Similarly, FF20-FF23 are repeated through FF24-FF3F.

 * Color Computer Disk Controller Reference CoCo 1/2/3 *
 * Chip is WD2797 *

FF40 (65344) Disk Controller DSKREG CoCo 1/2/3	
Bit 7	halt flag 0 = disabled 1 = enabled
Bit 6	drive select 3
Bit 5	density flag 0 = single 1 = double
Bit 4	write precompensation 0 = no precomp 1 = precomp
Bit 3	drive motor enable 0 = motors off 1 = motors on
Bit 2	drive select 2
Bit 1	drive select 1
Bit 0	drive select 0

- (1) This is a write only register
 - (2) Write precomp should be on for tracks over 22.
 - (3) Disk communication is done through FF48-FF4B as follows
- | Reg | Read operation | Write operation |
|------|----------------|-----------------|
| FF48 | Status | Command |
| FF49 | Track | Track |
| FF4A | Sector | Sector |
| FF4B | Data | Data |

(4) See FF48 for the list of commands.

FF41-7(65345-65351) CoCo 1/2/3	
FF41-7	DSKREG IMAGES

(1) Copies of disk registers?

FF48 (65352) Floppy Disk Controller STATUS/COMMAND REGISTER FDCREG CoCo 1/2/3	
FF48	Status/Command register for disk controller

(1) Write sends a command, then read to get status

COMMANDS	TYPE	COMMAND CODE
RESTORE	I	\$03
SEEK	I	\$17
STEP	I	\$23
STEP IN	I	\$43
STEP OUT	I	\$53
READ SECTOR	II	\$80
WRITE SECTOR	II	\$A0
READ ADDRESS	III	\$C0
READ TRACK	III	\$E4
WRITE TRACK	III	\$F4
FORCE INTERRUPT	IV	\$D0

(2) Read obtains status resulting from a command. See Status explained elsewhere

(3) Commands

Bit	Command
7 6 5 4 3 2 1 0	Command
0 0 0 0 x x x x	Restore to track 0
0 0 0 1 x x x x	Seek

0 0 1 x x x x Step
 0 1 0 x x x x Step in
 0 1 1 x x x x Step out

Bits:

4: 0:No update of track reg
 1:Update track register
 3: 0:Unload head at start
 1:Load head at start
 2: 0:No verify of track no
 1:Verify track no. on disc
 1-0:Read as 2-bit stepping rate:
 00 = 6ms
 01 = 12ms
 10 = 20ms
 11 = 30ms

1 0 0 x x x x 0 Read sector
 1 0 1 x x x x x Write sector
 1 1 0 0 0 x x 0 Read address
 1 1 1 0 0 x x 0 Read track
 1 1 1 1 0 x x 0 Write track

Bits:

4: 0:Read/write 1 sector
 1:Read all sectors till the end of a track.
 3: Interpretation of 2 bit sector length field in sector header
 0: Field is interpreted as
 00 = 256 bytes/sector
 01 = 512 bytes/sector
 10 = 1024 bytes/sector
 11 = 128 bytes/sector
 1: Field is interpreted as
 00 = 128 bytes/sector
 01 = 256 bytes/sector
 10 = 512 bytes/sector
 11 = 1024 bytes/sector (set to 1 on Dragon)
 2: 0:No head loading delay
 1:Head loading delay of 30ms prior to read/writes.
 1: 0:Set side select o/p to 0
 1:Set side select o/p to 1
 0: 0:Write Data Address Mark
 1:Write Deleted Data

Address mark

1 1 0 1 x x x x Force Interrupt
 Generate an interrupt & terminate the current operation on:

Bits set:

0 - Drive status transition Not-Ready to Ready
 1 - Drive status transition Ready to Not-Ready
 2 - Index pulse
 3 - Immediate interrupt

Bits clear:

No interrupt occurs, all operations terminated. (\$D0)

Status (read), when set:

Status bits may have different meanings depending on
 the command being performed.

0 - Drive busy
 1 - Data Request (Data Read/Data Written) OR Index Pulse
 2 - Lost Data/Track 00
 3 - CRC error

4 - Record Not Found/Seek Err
 5 - Data Address Mark
 0:Data Address Mark read
 1:Deleted Data Address Mark read OR Head Loaded
 6 - Write Protect
 7 - Not Ready

FF49(65353) FDC Track Register		CoCo 1/2/3
FF49	Disk Controller Track Register	
(1) Track is 0-34 decimal		
(2) Do not write directly, but use SEEK command		

FF4A(65354) FDC Sector Register		CoCo 1/2/3
FF4A	Disk Controller Sector Register	
(1) Sector is 1-18 decimal		
(2) Can write directly		

FF4B(65355) FDC Data Register		CoCo 1/2/3
FF4B	Disk Controller Data Register	
(1) Read or write data bytes from/to the disk controller		
(2) Must do so at the exact needed rate or there will be errors		

FF50(65360)-FF5F(65375) Unused		CoCo 1/2/3
--------------------------------	--	------------

 * Color Computer Miscellaneous Hardware CoCo 1/2/3 *

FF60(65376)-FF62(65378) X-Pad interface?		CoCo 1/2/3
FF60	X COORDINATE FOR X-PAD	
FF61	Y COORDINATE FOR X-PAD	
FF62	STATUS REGISTER FOR X-PAD	
(1) No more info known		

FF63(65379)-FF67(65383) Unused		CoCo 1/2/3
--------------------------------	--	------------

FF68(65384)-FF6B(65387) RS-232 PROGRAM PAK Interface		CoCo 1/2/3
FF68	READ/WRITE DATA REGISTER	
FF69	STATUS REGISTER	

FF6A	COMMAND REGISTER
FF6B	CONTROL REGISTER

(1) No more info known - todo

FF80(65408)-FF8F(65424) Unused in CoCo 3	CoCo 3
--	--------

(1) FF90-FFBF are used in CoCo3 for the GIME chip, elsewhere in this doc

 * Color Computer 3 GIME Hardware Reference *
 * TODO - Chip info? *

FF6C(65388)-FF6F(65391) Direct Connect Modem Pak	CoCo 1/2/3
--	------------

FF6C	READ/WRITE DATA REGISTER
FF6D	STATUS REGISTER
FF6E	COMMAND REGISTER
FF6F	CONTROL REGISTER

(1) No more info known - todo

FF90 (65424) Initialization Register 0 - INIT0	CoCo 3
--	--------

Bit 7	CoCo Bit	1 = Color Computer 1/2 Compatible, 0 = CoCo3
Bit 6	M/P	1 = MMU enabled
Bit 5	IEN	1 = GIME IRQ output enabled to CPU, 0 = disabled
Bit 4	FEN	1 = GIME FIRQ output enabled to CPU, 0 = disabled
Bit 3	MC3	1 = Vector RAM at FEXX enabled, 0 = disabled
Bit 2	MC2	1 = Standard SCS (DISK) (0=expand 1=normal)
Bit 1	MC1	ROM Map - see note (1)
Bit 0	MC0	" "

FF70(65392)-FF79(65401) Unused	CoCo 1/2/3
--------------------------------	------------

FF7A(65392)-FF7B(65404) Orchestra-90	CoCo 1/2/3
--------------------------------------	------------

FF7A	left channel
FF7B	right channel

(1) MC1 Bit MC0 Bit ROM MAP (vectors excluded)
 0 x 16K Internal, 16K External
 1 0 32K Internal
 1 1 32K External (except interrupt vectors)

(2) SCS is Spare Chip Select

(3) To get CoCo 1/2: CoCo bit set, MMU disabled, Video address from SAM, RGB/Comp Palettes => CC2.

(4) To use CoCo 3 graphics, the COCO bit must be set to zero. When using CoCo 1/2 resolutions, the bit is set to 1. RSDOS typically sets the INIT0 register to 196 in CoCo 2 resolutions and 68 when using CoCo 3 graphics modes.

FF7C(65404) Unused	CoCo 1/2/3
--------------------	------------

FF7D(65405)-FF7E(65406) SOUND/SPEECH CARTRIDGE	CoCo 1/2/3
--	------------

FF7D	SOUND/SPEECH CARTRIDGE RESET
FF7E	SOUND/SPEECH CARTRIDGE READ/WRITE

(1) No more info known - todo

FF91 (65425) Initialization Register 1 - INIT1	CoCo 3
--	--------

Bit 7	Unused
Bit 6	Memory type 1=256K, 0=64K chips
Bit 5	TINS Timer INput clock source 1=279.365 nsec, 0=63.695 usec
Bits 4-1	Unused
Bit 0	MMU Task Register select 0=enable FFA0-FFA7 1=enable FFA8-FFAF

FF7F (65407) MULTI-PAK PROGRAMMING REGISTER	CoCo 1/2/3
---	------------

FF7F	Multi-Pak programming register
Bit 7	(1)
Bit 6	(1)
Bits 5-4	Number of active CTS slot (ROM)
Bit 3	(1)
Bit 2	(1)
Bits 1-0	Number of active SCS slot (FDC)

(1) all set means value given is select switch setting

(1) TINS=1 is a 279.365 ns clock, not a 70ns clock as published some places. TINS = 0 is default

(2) The TINS bit selects the clock speed of the countdown timer. The 279 ns clock is useful for interrupt driven sound routines while the 63 us clock is used for a slower timer.

(3) The task register select which set of MMU bank registers to assign to the CPU's 64K workspace. The task bit is generally set to zero in DECB.

FF80(65408)-FFBF(65471) Unused in CoCo 1/2	CoCo 1/2
--	----------

(1) FF90-FFBF are used in CoCo3 for the GIME chip, elsewhere in this doc

FF92 (65426) Interrupt request enable register - IRQENR	CoCo 3
---	--------

Bits 7-6	Unused
Bit 5	TMR 1=Enable timer IRQ, 0 = disable
Bit 4	HBORD 1=Enable Horizontal border Sync IRQ, 0 = disable
Bit 3	VBORD 1=Enable Vertical border Sync IRQ, 0 = disable
Bit 2	EI2 1=Enable RS232 Serial data IRQ, 0 = disable
Bit 1	EI1 1=Enable Keyboard IRQ, 0 = disable

Bit 0 | EIO 1=Enable Cartridge IRQ, 0 = disable

(1) This register works the same as FIRQENR except that it generates IRQ interrupts.

(2) See notes following FF93 FIRQENR for more interrupt information.

FF93 (65427) Fast interrupt request enable reg - FIRQENR CoCo 3

Bits 7-6	Unused
Bit 5	TMR 1=Enable timer FIRQ, 0 = disable
Bit 4	HBORD 1=Enable Horizontal border Sync FIRQ, 0 = disable
Bit 3	VBORD 1=Enable Vertical border Sync FIRQ, 0 = disable
Bit 2	EI2 1=Enable RS232 Serial data FIRQ, 0 = disable
Bit 1	EI1 1=Enable Keyboard FIRQ, 0 = disable
Bit 0	EIO 1=Enable Cartridge FIRQ, 0 = disable

(1) TMR: FIRQ interrupt generated whenever 12 bit timer counts down to zero.

(2) HBORD: Horiz border FIRQ interrupt generated on falling edge of HSYNC.

(3) VBORD: Vert border FIRQ interrupt generated on falling edge of VSYNC.

(4) EI2: Serial FIRQ interrupt generated on falling edge of the signal on PIN 4 of the serial port.

(5) EI1: Keyboard FIRQ interrupt generated whenever a zero appears on any one of PA0-PA6 on the PIA0.

(6) EIO: Cartridge FIRQ interrupt generated on the falling edge of the signal on PIN 8 of the cartridge port.

(7) Reading from the register tells you which interrupts came in and acknowledges and resets the interrupt source.

(8) Here's a table of the interrupt vectors and where they end up going. You can't change the \$FFxx vectors, but you can change the \$FExx and \$0lxx vectors which contain jmps/lbras to the interrupt routine.
Be sure to disable the interrupt you are setting before changing values.

Interrupt ->	CPU reads ->	points to ->	jumps to this routine
SWI3	FFF2	FEEE	0100
SWI2	FFF4	FEF1	0103
FIRQ	FFF6	FEF4	010F
IRQ	FFF8	FEF7	010C
SWI	FFFA	FEFA	0106
NMI	FFFC	FEFD	0109
RESET	FFFE	8C1B	

This is in order of increasing precedence. Thus an IRQ firing while a IRQ is being serviced will interrupt the FIRQ. Conversely, a FIRQ never interrupts an IRQ.

Note that the equivalent interrupt output enable bit must be set in FF90

(9) You can also read these regs to see if there is a LOW on an interrupt input pin. If you have both the IRQ and FIRQ for the same device enabled, you read a 1 bit on both regs if that input is low.
For example, if you set FF02=0 and FF92=2, then as long as a key is held down, you will read back bit 1 as Set.

FF94 (65428) Timer register MSB - TIMERSMB CoCo 3

Bits 7-4	Unused
Bits 3-0	TMRH - Timer Bits 8-11 - write here to start timer

FF95 (65429) Timer register LSB - TIMERLSB CoCo 3

Bit 7-0	TIMRL - Timer Bits 0-7
---------	------------------------

(1) The 12 bit timer can be loaded with any number from 0-4095. The timer

resets and restarts counting down as soon as a number is written to FF94. Writing to FF95 does not restart the timer, but the value does save. Reading from either register does not restart the timer. When the timer reaches zero, it automatically restarts and triggers an interrupt (if enabled). The timer also controls the rate of blinking text. Storing a zero to both registers stops the timer from operating. Lastly, the timer works slightly differently on both 1986 and 1987 versions of the GIME. Neither can actually run a clock count of 1. That is, if you store a 1 into the timer register, the 1986 GIME actually processes this as a '3' and the 1987 GIME processes it as a '2'. All other values stored are affected the same way : nnn+2 for 1986 GIME and nnn+1 for 1987 GIME.

(2) Must turn timer interrupt enable off/on again to reset timer IRQ/FIRQ.

(3) Storing a \$00 at \$FF94 seems to stop the timer. Also, apparently each time it passes thru zero, the \$FF92/93 bit is set without having to re-enable that Int Request.

FF96 (65430) Unused CoCo 3

FF97 (65431) Unused CoCo 3

Bits 7-0 | Both registers unused

FF98 (65432) Video mode register - VMODE CoCo 3

Bit 7	BP 0=alphanumeric (text modes), 1=bit plane (graphics modes)
Bit 6	Unused
Bit 5	DESCEN 1= extra DESCender ENable(text), swap artifact colors
Bit 4	MOCH MOnoCHrome (composite video output) (1=mono, 0 = color
Bit 3	H50 1=50hz vs 0=60hz bit
Bit 2	LPR2 \
Bit 1	LPR1 - Number of lines/char row
Bit 0	LPR0 /

(1) LPR210 is Lines Per Row:

000 - 1 line/row	100 - 9
001 - 2 (CoCol&2)	101 - 10 (Reserved?)
010 - 3 (CoCol&2)	110 - 11 (12?(CoCol&2?))
011 - 8	111 - (12?) Infinite*

(2) Bit 5 is the artifact color shift bit. Change it to flip Pmode 4 colors. A One is what is put there if you hold down the F1 key on reset.
POKE &HFF98,&H13 from Basic if colors artifact the wrong way for you.

*Mostly useless, but it does generate a graphics mode where the whole screen is filled with the same line of graphics - like a 320x1 resolution. This can be used for a very fast oscilloscope type display where the program only updates data in one scan line over time and as the screen refreshes, you get a screen full of samples. Sockmaster used it in his Boink bouncing ball demo to take manual control of the vertical resolution of the screen to make the ball appear that it's going up and down (without actually scrolling the whole screen up and down).

FF99 (65433) Video resolution register - VRES CoCo 3

Bit 7	Unused (?)
Bit 6	LPF1 - Lines Per Field - bit 1 00= 192 lines 10= 210 lines
Bit 5	LPF0 - Lines Per Field - bit 0 01= 200 lines 11= 225 lines

Bit 4	HR2 Horizontal res, bit 2	see below
Bit 3	HR1 Horizontal res, bit 1	
Bit 2	HR0 Horizontal res, bit 0	
Bit 1	CO1 Color bit 1	
Bit 0	CO0 Color bit 0	

- (1) Bits 6-5: Lines Per Field LPF:
 00 -> 192 scan lines on screen
 01 -> 200 scan lines on screen
 10 -> *zero/infinite lines on screen (undefined)
 11 -> 225 scan lines on screen

- (2) Bits 4-2: Horizontal resolution HR
 Graphics modes:

000=16 bytes per row
 001=20 bytes per row
 010=32 bytes per row
 011=40 bytes per row
 100=64 bytes per row
 101=80 bytes per row
 110=128 bytes per row
 111=160 bytes per row

Text modes (HR1 - don't care for text):

0x0=32 characters per row
 0x1=40 characters per row
 1x0=64 characters per row
 1x1=80 characters per row

- (3) Bits 1-0 CRES Color Resolution
 Graphics modes:

00=2 colors (8 pixels per byte)
 01=4 colors (4 pixels per byte)
 10=16 colors (2 pixels per byte)
 11=Undefined (would have been 256 colors!?)

Text modes:

x0=No color attributes
 x1=Color attributes enabled

*The zero/infinite scanlines setting will either set the screen to display nothing but border (zero lines) or graphics going all the way up and down out of the screen, never retriggering. It all depends on when you set the register. If you set it while the video raster was drawing the vertical border you get zero lines, and if you set it while video was drawing graphics you get infinite lines. Mostly useless, but it should be possible to coax a vertical overscan mode using this with some tricky timing.

Old SAM modes work if CC Bit set. HR and CRES are Don't Care in SAM mode. Note the correspondence of HR2 HR0 to the text mode's bytes/line. Also that CRES bits shifted left one = number of colors.

Commonly used graphics modes:

Width	Colors	HR210	CO10
640	4	111	01
640	2	101	00
512	4	110	01
512	2	100	00
320	16	111	10
320	4	101	01
320	2	011	00
256	16	110	10
256	4	100	01
256	2	010	00
160	16	101	10
160	4	011	01 *
160	2	001	00 *

128	16	100	10	*
128	4	010	01	*
128	2	000	00	*

* - not supported. Other combos also possible but not supported.

- (4) HiRes text always two bytes per character; even byte 6 bit character, odd byte attribute. Characters from 128 ASCII, no graphic chars.

Format is

Bit 7	1 = Blink
Bit 6	1 = Underline
Bit 5	MSB Foreground Palette 0-7 from FFB0-FFB7
Bit 4	" " " "
Bit 3	LSB " " " "
Bit 2	MSB Background Palette 0-7 from FFB8-FFBF
Bit 1	" " " "
Bit 0	LSB " " " "

FF9A (65434) Border color register - BRDR CoCo 3

Bits 7-6	Unused
Bits 5-0	Border palette color, same format as FFB0-FFBF

- (1) This controls the color of the border around the screen. The color bits work the same as the palette registers. This register only controls the border color of CoCo 3 video modes and does not affect CoCo 1/2 modes.
 (2) See FFB0-FFBF for color definition
 (3) Format depends on Composite or RGB monitor

FF9B (65435) Reserved CoCo 3

Bits 7-2	Unused
Bit 1-0	VBANK Used by Disto 2 Meg upgrades to switch between 512K banks

FF9C (65436) Vertical scroll register - VSC CoCo 3

Bits 7-4	Unused
Bits 3-0	VSC Vertical smooth scroll 3=MSB <-> LSB=0 vals 0=16

The vertical scroll register is used to allow smooth scrolling in text modes. Consecutive numbers scroll the screen upwards one scan line at a time in video modes where more than one scan line makes up a row of text (typically 8 lines per character row) or graphics (double height+graphics).

FF9D (65437) Vertical offset register MSB CoCo 3

Bits 7-0	Y15-Y8 MSB Start of video in RAM (video location * 2048)
----------	--

FF9E (65438) Vertical offset register LSB CoCo 3

Bit 7	Y7-Y0 LSB Start of video in RAM (video location * 8)
-------	--

FF9D VERTICAL OFFSET V SCROLL MUST BE \$0F

FF9D Screen start address Bits 18-11

FF9E Screen Start Address Register 0 (bits 10-3)
 FF9E V OFFSET #2 WORD = ADDRESS/8 EX. \$E000 = \$60000/8
 BIT 7 WHY 8? BECAUSE 4 BITS(=8) FOR SCROLL
 |
 BIT 0 LSB
 FF9E Screen start address Bits 10-3
 DDDDDDDDEEEEEEE000

FF9E (65438) Vertical offset register LSB

Y15-Y0 is used to set the video mode to start in any memory location in 512K by steps of 8 bytes. On a 128K machine, the memory range is \$60000-\$7FFFF. There is a bug in some versions of the GIME that causes the computer to crash when you set odd numbered values in FF9E in some resolutions, so it's safest to limit positioning to steps of 16 bytes. Fortunately, you can use FF9F to make up for it and get steps as small as 2 bytes.

FFA3/B page 6000-7FFF
 FFA4/C page 8000-9FFF
 FFA5/D page A000-BFFF
 FFA6/E page C000-DFFF
 FFA7/F page E000-FFFF (or E000-FDFF - see (1))

- (1) The MMU registers select 8K pages from the GIME addressable space 0-7FFFF into CPU addressable space 0-FFFF in 8K blocks.
- (2) The pages are numbered by the top 6 bits of the address, and are 30-3F for a 128K machine, and 0-3F for a 512K machine.
- (3) In a 128K machine pages 0-2F are copies of pages 30-3F.
- (4) The registers to set the various 8K blocks, and power-up contents:

MMU Register:		CPU:			
Task0	Task1	Logical Address	/ Block#	Default	page
FFA0	FFA8	0000 - 1FFF	0	38	
FFA1	FFA9	2000 - 3FFF	1	39	
FFA2	FFAA	4000 - 5FFF	2	3A	
FFA3	FFAB	6000 - 7FFF	3	3B	
FFA4	FFAC	8000 - 9FFF	4	3C	
FFA5	FFAD	A000 - BFFF	5	3D	
FFA6	FFAE	C000 - DFFF	6	3E	
FFA7	FFAF	E000 - FDFF	7	3F	

- (5) Here is the GIME address view and default page usage:

Page	GIME Address	CPU Address*	Standard Page Contents
00-2F	00000-5FFFF		512K upgrade RAM, not in 128K
30	60000-61FFF		Hi-Res page #1
31	62000-63FFF		Hi-Res page #2
32	64000-65FFF		Hi-Res page #3
33	66000-67FFF		Hi-Res page #4
34	68000-69FFF		HGET/HPUT buffer
35	6A000-6BFFF		Secondary Stack
36	6C000-6DFFF		Hi-Res text screen RAM
37	6E000-6FFFF		unused
38	70000-71FFF	0000-1FFF	Basic memory
39	72000-73FFF	2000-3FFF	Basic memory
3A	74000-75FFF	4000-5FFF	Basic memory
3B	76000-77FFF	6000-7FFF	Basic memory
3C	78000-79FFF	8000-9FFF	Extended Basic Interpreter
3D	7A000-7BFFF	A000-BFFF	Color Basic Interpreter
3E	7C000-7DFFF	C000-DFFF	Disk Basic Interpreter
3F	7E000-7FFFF	E000-FFFF	Super Basic, GIME regs, I/O, Interrupts

- (6) FF91 Bit 0 selects task 0 (bit = 0) or task 1 (bit = 1)
 Task 0 uses MMU pages from FFA0-7 and Task 1 uses MMU pages from FFA8-7
- (7) FE00-FFFF can be held constant at 7FExx
- (8) If you don't know it is safe not to, you should turn off interrupts before swapping MMU blocks. Be very careful when swapping out ROM or low system RAM.
- (9) These registers can be read, but the top two bits must be mased out since they might contain garbage.

FF9F (65439) Horizontal offset register - TODO - CoCo 3

Bit 7 HVEN 1=Horizontal virtual screen enable (256 bytes per row)
 Bit 6 0=Normal horizontal display
 Bit 5
 Bit 4 0-127 byte offset from
 Bit 3 - FF9D/FF9E
 Bit 2
 Bit 1
 Bit 0

- (1) If Bit 7 set & in Text mode there are 128 chars (only 80 seen)/line. This allows an offset to be specified into a virtual 128 char/line screen, useful for horizontal hardware scrolling on wide text or spreadsheets.
- (2) If you set Bit 7 and you're in Gfx mode, you can scroll across a 128 byte picture. To use this, of course, you'd have to write your own gfx routines. On my machine, tho, an offset of more than about 5 crashes.

Bit 7
 Bits 6-0 X6-X0 Horizontal offset address (video location *2)

You can combine the horintal and vertical offsets to get a higher definition video position: Y15-Y4,X6-X0 which gives you 19 bit positioning by steps of 2 bytes. Otherwise, you can use this register to do scrolling effects. The virtual screen mode allows you to set up a 256 byte wide graphics or text screen, showing only part of it at a time and allowing you to scroll it vertically.

FFA0-FFA7 (65440-65447) MMU bank registers (task 0) CoCo 3

FFA8-FFAF (65448-65455) MMU bank registers (task 1) CoCo 3

FFA0/8 page 0000-1FFF
 FFA1/9 page 2000-3FFF
 FFA2/A page 4000-5FFF

FFB0-FFBF (65456-65471) Color palette registers -TODO CoCo 3

FFB0 todo RGB Mode: Bits 7-6 Unused
 FFB1 - names Bit 5 = High order Red R1
 FFB2 Bit 4 = High order Green G1
 FFB3 Bit 3 = High order Blue B1
 FFB4 Bit 2 = Low order Red R0
 FFB5 Bit 1 = Low order Green G0

FFB6	Bit 0 = Low order Blue	B0
FFB7	Composite mode:	
FFB8	Bits 5-4 = 4 intensity levels	I1 I0
FFB9	Bits 3-0 = 16 colors	P3 P2 P1 P0
FFBA	Todo - RGB/Composite bit?	
FFBB		
FFBC		
FFBD		
FFBE		
FFBF		

- (1) These 16 registers set the 16 colors used in the system.
- (2) Their format depends on the RGB/Composite bit setting in TODO
- (3) They can be read, but the top two (or three) bits must be masked off for correctness.
- (4) Both reading and writing to the palette registers causes a small 'glitch' on the screen. To avoid them change the palettes while the video retrace is in the vertical or horizontal border.
- (5) The BORDER register uses the same format, and also depends on the RGB/COMPOSITE setting
- (6) FFB0-FFB7 are also used for the text mode character background colors, and FFB8-FFBF TODO
- (7) Here are the default RGB palette values on power up: (TODO composite)

FFB0	GREEN	12	FFB8	BLACK	00
FFB1	YELLOW	36	FFB9	GREEN	12
FFB2	BLUE	09	FFBA	BLACK	00
FFB3	RED	24	FFBB	BUFF	3F
FFB4	BUFF	3F	FFBC	BLACK	00
FFB5	CYAN	10	FFBD	GREEN	12
FFB6	MAGENTA	2D	FFBE	BLACK	00
FFB7	ORANGE	26	FFBF	ORANGE	26

Full graphic 3-C	1	1	0	0	1	0	0	128x96x4	C00(3072)
Full graphic 3-R	1	1	0	1	1	0	1	128x192x2	C00(3072)
Full graphic 6-C	1	1	1	0	1	1	0	128x192x4	1800(6144)
Full graphic 6-R	1	1	1	1	1	1	0	256x192x2	1800(6144)
Direct memory access	X	X	X	X	1	1	1	TODO	

- (6) Notes:
- The graphic modes with -C are 4 color, -R is 2 color.
 - 2 color mode - 8 pixels per byte (each bit denotes on/off)
 - 4 color mode - 4 pixels per byte (each 2 bits denotes color)
 - CSS (in FF22) is the color select bit:

Color set 0:	0 = black,	1 = green	for -R modes
	00 = green,	01 = yellow	for -C modes
	10 = blue,	11 = red	for -C modes
Color set 1:	0 = black,	1 = buff	for -R modes
	00 = buff,	01 = cyan,	for -C modes
	10 = magenta,	11 = orange	for -C modes

In semigraphic-4 mode, each byte is a char or 4 pixels:
 bit 7 = 0 -> text char in following 7 bits
 bit 7 = 1 -> graphic: 3 bit color code, then 4 bits for 4 quads of color
 colors 000-cyan, yellow, blue, red, buff, cyan, magenta, orange=111
 quad bits orientation UL, UR, LL, LR

In semigraphic-6 mode, each byte is 6 pixels:
 bit 7-6 = C1-C0 color from 4 color sets above
 bit 5-0 = 6 pixels in 2x3 block, each on/off
 TODO - orientation

Example: To set 6-C color set 0, lda #\$E0, sta in FF22, FFC3, FFC5
 To return to text mode, clra, sta in FF22, FFC2, FFC4

- (7) In the CoCo 3, The SAM is mostly CoCo 1/2 compatible Write-Only switches

FFC6(65478)-FFD3(65491) SAM Page Select Register-SAM_Fx CoCo 1/2/3

FFC6/7	SAM_F0, or F0CLR/F0SET
FFC8/9	SAM_F1, or F1CLR/F1SET
FFCA/B	SAM_F2, or F2CLR/F2SET
FFCC/D	SAM_F3, or F3CLR/F3SET
FFCE/F	SAM_F4, or F4CLR/F4SET
FFD0/1	SAM_F5, or F5CLR/F5SET
FFD2/3	SAM_F6, or F6CLR/F6SET

- (1) These registers denote the start of the image in RAM to display in CoCo 1 and 2 text and graphics modes. The value in F0-F6 times 512 (decimal) is the start of video RAM
- (2) SAM_Fx are seven pairs of addresses (F0-F6), and poking any value to EVEN addresses sets bit Fx off (0) in Video Display Generator (VDG) circuitry. Poking value to ODD addresses sets bit on (1) in VDG circuit.

FFD4(65492)-FFD5(65493) SAM Page Select Register-SAMPAG CoCo 1/2/3

FFD4	Any write sets page #1 P1 control bit to 0, 0 = normal
FFD5	Any write sets page #1 P1 control bit to 1

- (1) page register MPU addresses 0000-7FFF, apply page #1 if P1 = 1

FFD6(65494)-FFD9(65497) Clock Speed R0/R1 - SAM_R0/1 CoCo 1/2/3

FFD6	SAM_R0 - Any write sets R0 control bit to 0
------	---

 * Color Computer 1/2/3 SAM registers FFC0-FFDF *
 * The SAM chip is a Motorola 6883 chip *

FFC0(65472)-FFC5(65477) SAM Video Display mode - SAM_Vx CoCo 1/2/3

FFC0/1	SAM_V0, or V0CLR/V0SET
FFC2/3	SAM_V1, or V1CLR/V1SET
FFC4/5	SAM_V2, or V2CLR/V1SET

- (1) This allows setting video modes in the CoCo 1 and 2
- (2) SAM_Vx are three pairs of addresses (V0-V2), and poking any value to EVEN addresses sets bit Vx off (0) in Video Display Generator (VDG) circuitry. Poking value to ODD addresses sets bit on (1) in VDG circuit.
- (3) These registers work with FF22 for setting modes, and must match up
- (4) Default screen mode is semigraphic-4
- (5) Mode correspondence between the SAM and the VDG:

Mode	VDG Settings				SAM		Desc.	RAM used in hex(dec)
	A/G	GM2	GM1	GM0	V2/V1/V0			
Internal alphanumeric	0	X	X	0	0	0	0	32x16 (5x7 pixel ch)
External alphanumeric	0	X	X	1	0	0	0	32x16 (8x12 pixel ch)
Semigraphic-4	0	X	X	0	0	0	0	32x16 ch, 64x32 pixels
Semigraphic-6	0	X	X	1	0	0	0	64x48 pixels
Full graphic 1-C	1	0	0	0	0	0	1	64x64x4 400(1024)
Full graphic 1-R	1	0	0	1	0	0	1	128x64x2 400(1024)
Full graphic 2-C	1	0	1	0	0	1	0	128x64x4 800(2048)
Full graphic 2-R	1	0	1	1	0	1	1	128x96x2 600(1536)

FFD7	- Any write sets R0 control bit to 1
FFD8	SAM_R1 - Any write sets R1 control bit to 0
FFD9	- Any write sets R1 control bit to 1

(1) R1-R0: 00-0.89 MHZ only, 01-0.89/1.78 MHZ <== both transparent refresh
10-1.78 MHZ only, 11-1.78 MHZ

(2) May not work on early CoCo (and 2?), but works on all CoCo 3's (true?)

(3) 0.89 Mhz: no address-dependent speed

(4) Speedup only for ROM accesses?

(5) These are commonly used as follows:
 Slow poke: FFD8 write selects 0.89 Mhz CPU clock
 Fast poke: FFD9 write selects 1.79 Mhz CPU clock

(6) Switching the SAM into 1.8MHz operation gives the CPU the time ordinarily used by the VDG and refresh, so the display shows garbage, so this mode is seldom used. The SAM in Address Dependent mode, where ROM reads (since they do not use the DRAM) occur at 1.8MHz but regular RAM access occurs at .89MHz, runs the BASIC interpreter from ROM twice as fast, nearly doubling BASIC program performance.

FFDA(65498)-FFDD(65501)	Memory size M0/M1 - SAM_M0/1	CoCo 1/2/3
-------------------------	------------------------------	------------

FFD6	SAM_M0 - Any write sets M0 control bit to 0
FFD7	- Any write sets M0 control bit to 1
FFD8	SAM_M1 - Any write sets M1 control bit to 0
FFD9	- Any write sets M1 control bit to 1

(1) M1-M0: 00 - 4K, 01 - 16K
10 - 64K (all 3 dynamic), 11 = 64K static

(2) Todo - is this right? Or Dragon only?

FFDE/FFDF (65502/65503)	ROM/RAM map type - SAM_TYP	CoCo 1/2/3
-------------------------	----------------------------	------------

FFDE	Any write switches system ROMs into memory map (ROM mode)
FFDF	Any write selects all-RAM mode (RAM mode)

(1) RAM accesses use MMU translations in CoCo 3

(2) Default mode 0 - ROM Mode CoCo 1/2, Default mode 1 - RAM Mode CoCo 3

(3) These registers are often called TY=0 and TY=1

FFDE/FFDF (65502/65503)	ROM/RAM map type - TODO	CoCo 1/2/3
-------------------------	-------------------------	------------

FFDE	Any write switches system ROMs into memory map (ROM mode)
FFDF	Any write selects all-RAM mode (RAM mode)

(RAM accesses use MMU translations)

 * Color Computer Interrupt Vectors *

FFE0-FFF1 (65504/65522)	Reserved	CoCo 1/2/3
-------------------------	----------	------------

Unused

(1) Reserved for future enhancements :)

FFF2-FFFF (65523/65535)	Interrupt vectors	CoCo 1/2/3
-------------------------	-------------------	------------

FFF2/3	SWI3	points to FEEE
FFF4/5	SWI2	points to FEF1
FFF6/7	FIRQ	points to FEF4
FFF8/9	IRQ	points to FEF7
FFFA/B	SWI	points to FEFA
FFFC/D	NMI	points to FEFD
FFFE/F	RESET	points to 8C1B

(1) When an interrupt of the given type occurs, the vector is loaded into the Program Counter, which points to the address given above. You can set your own interrupt routines by replacing the FExx values with your own lbra XXXX values (TODO - hex?).

(2) Turn off interrupts before setting a new value.

(3) Restore what was there to restore the system

(4) See also the section on interrupts in this document

 * Color Computer 3 Detailed Memory Map *

 This section also contains some information on CoCo clones: Dragon 32 & 64.

Format conventions:
 xxxx references a CPU memory address
 0xab or 0xabcd are C style hexadecimal constants
 %TITLE% shows a 'standard' assembler reference
 UPPERCASE words typically refer to Basic keywords or Assembler mnemonics
 (0x1234) Numbers in brackets refer to the default value at power-up

Abbreviations:
 CoCo refers to the Tandy CoCo only
 D32 only applicable to Dragon 32
 D64 only applicable to Dragon 64
 DOS refers to a generic DragonDos compatible unless stated otherwise
 lsb least significant byte
 msb most significant byte
 ptr pointer (or address of)
 w/o without

0000	BREAK message flag - if negative print BREAK
0001	String delimiting char (0x22 '"')
0002	Another delimiting char (0x22 '"')
0003	General counter byte
0004	Count of IFs looking for ELSE
0005	DIM flag
0006	%VALTYP% Variable type flag (0x00 numeric, Non-0x00 string)
0007	Garbage collection flag
0008	Subscript allowed flag
0009	INPUT/READ flag
000a	Arithmetic use
000b:000c	String ptr first free temporary
000d:000e	String ptr last free temporary
000f-0018	Temporary results
0019:001a	Start address of BASIC program (\$1e01, \$2401 with DOS)
001b:001c	Start address of simple variables
001d:001e	Start address of array variables
001f:0020	End of storage, Start of unused mem after BASIC program
0021:0022	Top of stack, growing down (\$7e36)
0023:0024	Top of free string space (\$7ffe)
0025:0026	Temp Ptr to string in string space
0027:0028	Top of Ram available to BASIC - returned by DOS HIMEM (\$7ffe)

0029:002a	Last/CONT line number	0082	Cassette I/O - Pulse width counter
002b:002c	Temp/Input line number store	0083	Cassette I/O - Sync bits counter
002d:002e	Ptr to next statement to be executed	0084	Cassette I/O - Bit phase flag
002f:0030	Direct mode command text pointer	0085	Last sine wave value for output to DAC
0031:0032	Current DATA statement line number	0086	Data for low res SET/RESET, POINT routines
0033:0034	Ptr to next item in current DATA statement	0087	ASCII code of last key pressed (cleared by Break check)
0035:0036	Ptr to keyboard input buffer	0088:0089	Current VDU cursor addr (typ 0x0400-0x05ff)
0037:0038	Ptr to variable last in use	008a:008b	Gen purpose 16bit scratch pad / 16bit zero (0x0000)
0037:0038	ASCII codes of last variable used	008a:008b	CoCo - Motor on delay
0039:003a	VARPTR address of last variable used	008c	Sound pitch frequency
003b-004e	Evaluation variables	008d:008e	Gen purpose countdown (?sound timer)
0041:0042	High end destination addr for block	008f	Cursor flash counter (0x20)
0043:0044	High end origin addr	0090:0091	Cassette leader byte count - number of 0x55 bytes written as sync leader (D32 - 0x0080, D64 - 0x0100)
0045:0046	Low end destination addr for block	0092	Minimum cycle width of 1200Hz (0x12)
0047:0048	Low end origin addr	0092:0093	CoCo - Cassette leader byte count
004f-0054	Floating Point Accumulator Num 1	0093	Minimum pulse width of 1200Hz (0x0a)
004f	Exponent	0094	Maximum pulse width of 1200Hz (0x12)
0050-0053	Mantissa	0095:0096	Motor on delay (0xda5c = approx 0.5s)
0050:0051	16 bit values in FAC stored here	0095:0096	CoCo - Serial Baud rate constant (0x0057 = 600 baud)
0052:0053	VARPTR of variables is stored here	0097:0098	Keyboard scan debounce delay constant (0x045e)
0054	Mantissa Sign (0x00 positive, 0xff negative)	0097:0098	CoCo - Serial Line Printer End of Line delay (0x0001)
0055	Temp sign of FAC	0099	Printer comma field width (0x10 = 16)
0056	String variable length	009a	Printer last comma field (0x74 = 116) (CoCo 0x70 = 112)
0057-005b	String Descriptor temporaries	009b	Printer line width dflt (0x84 = 132)
005c-0061	Floating Point Accumulator Num 2	009c	Printer head column posn == POS(-2), Updated by LPOUT (\$800f) routine
0062	Sign comparison	009d:009e	EXEC default entry address (D32 - \$8b8d = ?FC ERROR; D64 - \$bf49 = Boot 64k mode)
0062-0067	Misc use	009f-00aa	%CHRGET% Self modifying routine to read next char
0063	CoCo - Extended precision byte	009f:00a0	INC <\$A7
0068:0069	Current Line number (0xffff in direct mode)	00a1:00a2	BNE \$00A5
006a-006e	Device Params used in PRINT	00a3:00a4	INC <\$A6
006a	Device Comma field width (VDU - 0x10)	00a5-00a7	LDA >xxxx
006b	Device Last comma field	00a6:00a7	Ptr to next character to read
006c	Device Current column num (VDU - 0x00-0x1f)	00a8-00aa	JMP \$BB26
006d	Device Line width - num chars per line (VDU 0x20)	00ab-00ae	Used by RND
006e	Cassette I/O in progress flag - 0xff on input or output occurring	00af	TRON/TROFF trace flag - non zero for TRON
006f	%DEVNUM% Current device number	00b0:00b1	Ptr to start of USR table (\$0134; DOS - \$0683)
	0x00 VDU screen	00b2	Current foreground colour (0x03)
	0x01-0x04 DOS - DosPlus only - drive number.	00b3	Current background colour (0x00)
	0xfd serial port (Dragon 64 only)	00b4	Temp/active colour in use
	0xfe printer	00b5	Byte value for current colour - ie bit pattern
	0xff tape	00b6	Graphics PMODE number in use (0x00)
0070	Cassette EOF flag - non-zero if EOF - used by EOF(-1)	00b7:00b8	Ptr to last byte+1 of current graphics mode (\$0c00 w/o Dos)
0071	Restart flag - if not 0x55 cold start on reset, see \$0072	00b9	Number of bytes per line in current PMODE (0x10)
0072:0073	Restart vector - Following a reset if \$0072 pts to a NOP opcode & \$0071 is 0x55 then a warm start is performed to this vector else a cold start. (0xb44f) (DOS SuperDosE6 \$c706)	00ba:00bb	Ptr to first byte of current graphics mode (\$0600)
0074:0075	Physical end of Ram minus 1 (0x7ffe)	00bc	Msb of start of graphics pages (0x06 or 0x0c with Dos)
0076:0077	Unused	00bd:00be	Current X cursor position (not user available ?)
0078	Cassette status	00bf:00c0	Current Y cursor position (not user available ?)
	0x00 closed	00c1	Colour set currently in use (0x08 if colorset 1)
	0x01 input	00c2	Plot/Unplot flag: 0x00 reset, non zero set
	0x02 output	00c3:00c4	Current horizontal pixel number
0079	Cassette I/O - Buffer size - bytes in block	00c5:00c6	Current vertical pixel number
007a:007b	Header buffer addr - ptr to filename block	00c7:00c8	Current X cursor coord (0x0080)
007c	%BLKTYP% Cassette block type	00c9:00ca	Current Y cursor coord (0x0060)
	0x00 filename	00cb:00cc	CIRCLE command X coord as if drawn in PMODE 4
	0x01 data	00cd:00ce	CIRCLE command Y coord as if drawn in PMODE 4
	0xff EOF block	00cf:00d0	CIRCLE radius as if drawn in PMODE 4
007d	%DBLEN% Cassette block length, number bytes read/to write	00cf:00d0	RENUM increment value
007e:007f	%DBADR% Cassette I/O Buffer address	00d1:00d2	RENUM start line
	Contains 1 + End address of last program loaded	00d3:00d4	CLOADM 2's complement load offset
0080	Cassette I/O - block checksum used internally	00d5:00d6	RENUM new start line
0081	Cassette I/O - error code	00d7	EDIT line length (not user available)
	0x00 none	00d7	PLAY -
	0x01 CRC (checksum) error	00d8	PLAY - bytes left in string
	0x02 attempt to load into ROM		

00d9:00da	PLAY - ptr to current char in string	014a	Number of bytes in EOL sequence 1-6 (0x01)
00d8-00dd	Graphics use ?	014b	EOL chr 1 (0x0d CR)
00de	PLAY: Current octave in use (0-4) (0x02)	014c	EOL chr 2 (0x0a LF)
00df:00e0	PLAY: Volume data for volume setting (D32 - 0xba42) (D64 - 0xb844)	014d	EOL chr 3 (D64 - 0x00; D32 - 0x20 ' ')
00e1	PLAY: Current note length (0x04)	014e	EOL chr 4 (D64 - 0x00; D32 - 0x44 'D' Duncan)
00e2	PLAY: Current tempo (0x02)	014f	EOL chr 5 (D64 - 0x00; D32 - 0x4e 'N' N.)
00e3:00e4	PLAY: Music duration count	0150	EOL chr 6 (D64 - 0x00; D32 - 0x4f 'S' Smeed)
00e5	PLAY: Music dotted note flag	0151-0159	Keyboard matrix state table
00e6-00ff	D32 - Unused in Dragon 32 w/o DOS	0152-0159	CoCo - Keyboard roll-over table
00e6	CoCo - baud rate constant	015a-015d	%POTVAL% Joystick values (0-63)
00e7	Coco - Input timeout constant	015a	Right Joystick, x value == JOYSTK(0)
00e8	Current angle used in DRAW (??)	015b	Right Joystick, y value == JOYSTK(1)
00e9	Current scale used in DRAW (??)	015c	Left Joystick, x value == JOYSTK(2)
00ea-00f6	DOS - Used by DragonDos	015d	Left Joystick, y value == JOYSTK(3)
00f8	DOS - sector currently seeking {SuperDos Rom}	015e-01a8	RAM hooks - each is called from ROM with a JSR before carrying out the specified function
0100-0102	SWI3 Secondary vector (Uninitialised)	015e-0160	Device Open (DOS JMP \$d902; SuperDosE6 \$d8f4)
0103-0105	SWI2 Secondary vector (Uninitialised)	0161-0163	Verify Device Number (DOS SuperDosE6 JMP \$d8ec)
0106-0108	SWI Secondary vector (Uninitialised)	0164-0166	Device Init (DOS SuperDosE6 JMP \$c29c)
0109-010b	NMI Secondary vector (Uninitialised)	0167-0169	Output char in A to DEVN (DOS JMP \$d8fa; SuperDosE6 \$d90b)
	(CoCo DOS JMP \$d7ae; SuperDos E6 JMP \$c71e)	0167	Setting to 0xff disables keyboard ???
010c-010e	IRQ Secondary vector - JMP \$9d3d		Setting to 0x39 (RTS) allows use of SCREEN 0,1 etc. ??
	(CoCo JMP \$a9b3 or \$894c (extended); CoCo DOS JMP \$d7bc; SuperDos E6 JMP \$c727)	016a-016c	Input char from DEVN to A (DOS SuperDosE6 JMP \$c29c)
010f-0111	FIRQ Secondary vector - JMP \$b469	016d-016f	Input from DEVN using INPUT (DOS SuperDosE6 JMP \$c29c)
	(CoCo JMP \$a0f6; SuperDos E6 JMP \$c7da)	0170-0172	Output to DEVN using PRINT (DOS SuperDosE6 JMP \$c29c)
0112:0113	TIMER value	0173-0175	Close all files (DOS SuperDosE6 JMP \$c29c)
0114	Unused	0176-0178	Close file(DOS JMP \$d917; SuperDosE6 \$d6f5)
0115-0119	Random number seeds (0x80, 0x4f, 0xc7, 0x52, 0x59)	0179-017b	Command Interpreter - interpret token in A as command (DOS SuperDosE6 JMP \$c29c)
011a-011f	D32 - Unused	017c-017e	Re-request input from keyboard (DOS JMP \$d960; SuperDosE6 \$d954)
011a	D64 - %FLAG64% checked on Reset from 64K mode if 0x55 then checksum at \$011b is checked against current contents of RAM, if the same then a warm start is performed (64 mode) else a cold start (32 mode)	017f-0181	Check keys - scan for BREAK, SHIFT+'@' (DOS SuperDosE6 JMP \$c29c)
011a	CoCo - Caps lock, 0x00 lower, non-0x00 upper	017f	Setting this to 0x9e disables LIST/DIR
011b:011c	D64 - %CSUM64% 16bit sum of words of BASIC Rom-in-ram in 64K mode from \$c000 to \$feff	0182-0184	Line input from DEVN using LINE INPUT (DOS JMP \$d720; SuperDosE6 \$dac5)
011b:011c	CoCo - Keyboard Delay constant	0185-0187	Close BASIC file read in and goto Command mode (DOS SuperDosE6 JMP \$c29c)
011d-011f	CoCo - JMP \$8489 ?	0188-018a	Check EOF on DEVN (DOS JMP \$dd4d; SuperDosE6 \$dd54)
011d	D64 - %LSTKEY% Last key code return by keybd poll routine	018b-018d	Evaluate expression (DOS SuperDosE6 JMP \$c29c)
011e	D64 - %CNTDWN% Auto repeat countdown	018e-0190	User error trap, called from \$8344 (DOS SuperDosE6 JMP \$c29c)
011f	D64 - %REPDLY% Auto repeat inter-repeat delay value (0x05)	0191-0193	System error trap, called from \$8344 (DOS JMP \$c69e; SuperDosE6 \$c6c5)
0120	%STUB0% Stub 0 - Number of reserved words (0x4e)	0194-0196	Run Link - used by DOS to RUN filename (DOS JMP \$d490; SuperDosE6 \$d4b7)
0121:0122	Stub 0 - Ptr to reserved words table (\$8033)	0197-0199	Reset Basic Memory, editing or entering BASIC lines
0123:0124	Stub 0 - Ptr to reserved words dispatch table (\$8154)	019a-019c	Get next command - reading in next command to be executed
0125	Stub 0 - Number of functions (0x22)	019d-019f	Assign string variable
0126:0127	Stub 0 - Ptr to reserved function words table (\$81ca)	01a0-01a2	Screen access - CLS, GET, PUT
0128:0129	Stub 0 - Ptr to function words dispatch table (\$8250)	01a3-01a5	Tokenise line
012a	%STUB1% Stub 1 - Number of reserved words (0x00) (DOS 0x1a)	01a6-01a8	De-Tokenise line
012b:012c	Stub 1 - Ptr to reserved words table (0x0000) (DOS \$ded4; SuperDosE6 \$deda)	01a9-01d0	String buffer area
012d:012e	Stub 1 - Ptr to reserved words token processing routine (\$89b4; DOS \$c64c; SuperDosE6 \$c670)	01d1	Cassette filename length in range 0-8
012f	Stub 1 - Number of functions (0x00) (DOS 0x07)	01d2-01d9	Cassette filename to search for or write out
0130:0131	Stub 1 - Ptr to function table (0x0000) (DOS \$debb; SuperDosE6 \$decl)	01da-02d8	Cassette I/O default data buffer - 255 bytes
0132:0133	Stub 1 - Ptr to function token processing routine (\$89b4; DOS \$c667; SuperDosE6 \$c68b)	01da-0268	D64 - 64K mode bootstrap routine is copied here to run
0134	%STUB2% Stub 2 - acts as a stub terminator under DOS	01da-01e1	Cassette buffer - filename of file read
0134-0147	USR address table, relocated by DOS (10 x 2 bytes) (\$8b8d)	01e2	Cassette buffer - filetype
0148	Auto line feed flag on buffer full - setting this to 0x00 causes a EOL sequence to be sent to printer when buffer reaches length in \$009b (0xff)		0x00 BASIC program
0149	Alpha Lock flag - 0x00 Lower case, 0xff Upper case (0xff)	01e3	0x01 Data
014a-0150	Line Printer End of line termination sequence		0x02 Machine code
		01e3	Cassette buffer - ASCII flag
			0x00 Binary
			0xff ASCII flag
		01e4	Cassette buffer - gap flag
			0x00 Continous

```

01e5:01e6 0xff Gapped file
01e5:01e6 Cassette buffer - Entry (Exec) addr of m/c file
01e7:01e8 Cassette buffer - Load address for ungapped m/c file
02d9-02dc BASIC line input buffer preamble
02dd-03d8 BASIC line input buffer - used for de-/tokenising data
02dd-03dc CoCo - 255 byte keyboard buffer
02e1-033b CoCo - 90 byte screen buffer
03d9-03ea Buffer space
03eb-03fc Unused
03fd-03ff D32 - Unused in Dragon 32
03fd:03fe D64 - Printer end of line delay in milliseconds (0x0000)
03ff D64 - %PRNSEL% selects default printer port
           0x00 Parallel, non-0x00 Serial (0x00)
0400-05ff Default Text screen
0600-1dff Available graphics pages w/o DOS
0600-0bff DOS - workspace area see also $00ea-$00f6
0600-0dff CoCo DOS workspace area (no more info)
0c00-23ff DOS - Available graphics pages
8000-bfff BASIC ROM in 32K mode
8000-9fff CoCo - Extended Color BASIC ROM
a000-bfff CoCo - Color BASIC ROM
bff0-bfff These addresses mapped from ROM to $fff0-$ffff by the SAM
c000-dfff DOS - Dos ROM
c000-feff DOS - Cumana DOS ROM only
c000-feff Available address range to cartridge expansion port 32K mode
c000-feff D64 - 64K mode - copy of BASIC ROM 2 exists in RAM here
ff00 PIA 0 A side Data reg.
ff01 PIA 0 A side Control reg.
ff02 PIA 0 B side Data reg.
ff03 PIA 0 B side Control reg.
ff04 D64 - ACIA serial port read/write data reg.
ff05 D64 - ACIA serial port status (R)/ reset (W) reg.
ff06 D64 - ACIA serial port command reg.
ff07 D64 - ACIA serial port control reg.
ff20 PIA 1 A side Data reg.
ff21 PIA 1 A side Control reg.
ff22 PIA 1 B side Data reg.
ff23 PIA 1 B side Control reg.
ff40 DOS - Disk Controller command/status reg.
ff41 DOS - Disk Controller track reg.
ff42 DOS - Disk Controller sector reg.
ff43 DOS - Disk Controller data reg.
ff48 DOS - Disk Controller hardware control reg.
ffc0-ffdf SAM (Synchronous Address Multiplexer) register bits - use even
           address to clear, odd address to set
ffc0-ffc5 SAM VDG Mode registers V0-V2
ffc0/ffc1 SAM VDG Reg V0
ffc2/ffc3 SAM VDG Reg V1
ffc3/ffc5 SAM VDG Reg V2
ffc6-ffd3 SAM Display offset in 512 byte pages F0-F6
ffc6/ffc7 SAM Display Offset bit F0
ffc8/ffc9 SAM Display Offset bit F1
ffca/ffcb SAM Display Offset bit F2
ffcc/ffcd SAM Display Offset bit F3
ffce/ffcf SAM Display Offset bit F4
ffd0/ffc1 SAM Display Offset bit F5
ffd2/ffc3 SAM Display Offset bit F6
ffd4/ffd5 SAM Page #1 bit - in D64 maps upper 32K Ram to $0000 to $7fff
ffd6-ffd9 SAM MPU Rate R0-R1
ffd6/ffd7 SAM MPU Rate bit R0
ffd8/ffd9 SAM MPU Rate bit R1
ffda-ffdd SAM Memory Size select M0-M1
ffda/ffdb SAM Memory Size select bit M0
ffdc/ffdd SAM Memory Size select bit M1
ffde/ffdf SAM Map Type - in D64 switches in upper 32K RAM $8000-$feff
ffec-ffef PC-Dragon - Used by Burgin's emulator to provide enhanced services

```

```

fff0-ffff 6809 interrupt vectors mapped from $bff0-$bfff by SAM
fff0:fff1 Reserved ($0000; D64 64K mode 0x3634 '64')
fff2:fff3 SWI3 ($0100)
fff4:fff5 SWI2 ($0103)
fff6:fff7 FIRQ ($010f)
fff8:fff9 IRQ ($010c)
fffa:fffb SWI ($0106)
fffc:fffd NMI ($0109)
fffe:ffff RESET ($b3b4; D64 64K mode $c000 - never accessed)

```

Sources

- [1] Self experimentation :
- [2] Sockmaster's webpage (John Kowalski, <http://www.axess.com/twilight/sock/>)
- [3] Notes from Kevin K. Darling, help from Greg Law, Dennis W., and Marsha.
- [4] Notes from Mike Pepe
- [5] Notes from Graham E. Kinns
- [6] PC CoCo Emulator (coco2-13.zip) by Jeff Vavasour.
- [7] "The Dragon Notebook", Ray Smith, NDUG.
- [8] "Inside the Dragon", Duncan Smeed & Ian Sommerville, Addison-Wesley, 1983.
- [9] "TRS-80 Color Computer Tech Ref Manual", Tandy Corp, 1981.
- [10] WD2797 Floppy Disc Driver Controller Data Sheet (RS #6991).
- [11] Dragon Disc Controller Circuit Diagram, ex Dragon Data Ltd, now NDUG.
- [12] Dragon 32/64 Upgrade Manual, R. Hall, NDUG, 1985.
- [13] "Inside the 32", Dave Barnish, p13, Jan 1987.
- [14] "BREAKing the '64", Martyn Armitage, p8-9, Feb 1988.
- [15] "Firmware - Part 1", Brian Cadge, p19, Sep 1985.
- [16] "Dragon Answers", Brian Cadge, p31, Sep 1985.
- [17] Assembly Language Graphics for the TRS-80 Color Computer, Don & Kurt Inman, 1983, Reston Publishing Company, ISBN 0-8359-0318-4.
- [18] TRS-80 Color Computer Assembly Language, William Barden, Jr., Radio Shack, 1983
- [19] "What's Inside Radio Shack's Color Computer?", Byte Magazine, 1981, <http://www.byte.com/art/9603/sec5/art4.htm>
- [20] Notes from <http://www.cs.unc.edu/~yakowenk/coco.html>
- [21] "Assembly Language Programming for the Color Computer", Laurence A Tepolt, Tepco, 1985.
- [22] "Assembly Language Programming for the CoCo3", Laurence A Tepolt, Tepco, 1984?.
- [23] The Unravalled series: "Color Basic Unravalled II", "Extended Basic Unravalled II", "Super Extended Basic Unravalled II", "Disk Basic Unravalled II", Walter K. Zydhek, Spectral Associates, 1999.
- [24] Info from <http://www.trs-80.com/>

```

TODO - scan my asm books for more info
TODO - check for tabs, spacing correct, etc
       - see how prints, make 1, 2, and 4 page versions
       - Need lots of content filled in, verified, corrected.
       - final proof pass

```

END OF FILE