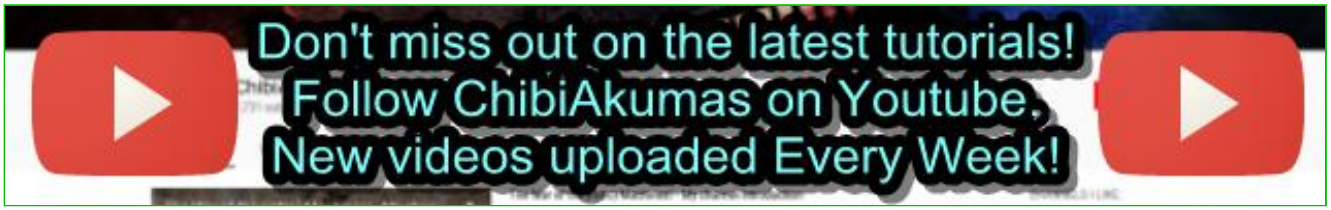

View Options
Default Dark
Simple (Hide this menu)
Print Mode (white background)

Top Menu
Main Menu
Youtube channel
Forum
AkuSprite Editor
Dec/Bin/Hex/Oct/Ascii Table

Z80 Content
Z80 Tutorial List
Learn Z80 Assembly
Hello World
Advanced Series
Multiplatform Series
Platform Specific Series
ChibiAkumas Series
Grime Z80
Z80 Downloads
Z80 Cheatsheet
Sources.7z
DevTools kit

6809 Assembly programming for the Vectrex

The FM-7 is one of a series of Fujitsu computers released in Japan

With a pair of 6809 CPU's it was widely popular in Japan, and is one of the few 6809 computers released.

In these tutorials we'll learn the about the FM7, and write some simple programs for it

	Vectrex
Cpu	Main: 1.5mhz 6809
Ram	1k Ram
Rom	32k
Sound	AY-3-8912
Max Resolution	INFINITE!!!

Vectrex!

OS Calls

Group	Address	Name	AltName	DP (Entry/Exit)	Details	Example
Init	**$F18B**	INITALL	INTALL	$D0	Full Vectrex initialization	
Init	**$F164**	INITMSC	INTMSC	$C8	Initialize misc. parameters	
Init	**$F14C**	INITPIA	INTPIA	$D0	Initialize PIA	
Init	**$F272**	INITPSG	INTPSG	$D0	Initialize the ◆REQx◆ area (sound mirror).	
Init	**$F533**	IREQ	INTREQ	-	Initialize the 'REQZ' area	
Init	**$F000**	POWER	PWRUP	-	Power-up handler	
Drawing	**$F2E6**	DEFLOK	--- C1:C1048576B1:B1048576	$D0	Overcome scan collapse circuitry	
Drawing	**$F192**	FRAM20	FRWAIT	$D0	Wait for frame boundary (Pause for next frame)	
Drawing	**$F29D**	INT1Q	-	$D0	Set beam intensity 1/4 ($1F)	
Drawing	**$F2A5**	INT3Q	-	$D0	Set beam intensity 3/4 ($5F)	
Drawing	**$F2AB**	INTENS	-	$D0	Set beam intensity A (0-127)	
Drawing	**$F2A9**	INTMAX	-	$D0	Set beam intensity 4/4 (7F)	
Drawing	**$F2A1**	INTMID	INT2Q	$D0	Set beam intensity 2/4 ($3F)	
Drawing	**$F30C**	POSIT1	-	$D0	Position relative vector List X (1x scale)	db Y,X

Z80 Platforms
Amstrad CPC
Elan Enterprise
Gameboy & Gameboy Color
Master System & GameGear
MSX & MSX2
Sam Coupe
TI-83
ZX Spectrum
Spectrum NEXT
Camputers Lynx

6502 Content
6502 Tutorial List
Learn 6502 Assembly
Advanced Series
Platform Specific Series
Hello World Series
Grime 6502
6502 Downloads
6502 Cheatsheet
Sources.7z
DevTools kit
6502 Platforms
Apple IIe
Atari 800 and 5200
Atari Lynx
BBC Micro
Commodore 64
Commander x16
Super Nintendo (SNES)
Nintendo NES / Famicom
PC Engine (Turbografx-16)
Vic 20

68000 Content

Drawing	**$F308**	POSIT2	-	$D0	Position relative vector (2x scale)	
Drawing	**$F30E**	POSITB	-	$D0	Position relative vector list X Scale B	
Drawing	**$F2FC**	POSITD	-	$D0	Position relative vector (X,Y)=(B,A) scale 1x	
Drawing	**$F312**	POSITN	-	$D0	Position relative vector (X,Y)=(B,A) scale T1LOLC	
Drawing	**$F310**	POSITX	-	$D0	Release integrators and position beam, lfrom X	db Y,X
Drawing	**$F2F2**	POSWID	-	$D0	Release integrators and position beam using 16-bit (Y,X) values from address X	dw Y,X
Drawing	**$F34F**	CZERO	ZEGO	$D0	Depending ZSKIP, zero integrators and set the sample / hold for active ground.	
Drawing	**$F35B**	ZEREF	ZEREF	$D0	Set active ground sample / hold to zero volts.	
Drawing	**$F34A**	ZERO.DP	ZERO.DP	$D0	DP=$D0... Zero integrators and set active ground	
Drawing	**$F36B**	ZERO.	ZERO	$D0	Zero the integrators only	
Drawing	**$F354**	ZEROIT	ZEROIT	$D0	Zero integrators and set active ground	
Diffy	**$F610**	DANROT	DROT	-	Rotate ◆Diffy◆ style list A=Angle B=no of Vectors X=Source list U=Dest	
Diffy	**$F433**	DASHE	DSHDF1	$D0	Draw dashed lines from 'DIFFY' list	
Diffy	**$F434**	DASHEL	DSHDF	$D0	Draw dashed lines from 'DIFFY' list A=Vectors-1 X=List	
Diffy	**$F437**	DASHY	DASHDF	$D0	Draw dashed lines from 'DIFFY' list X	
Diffy	**$F3DF**	DIFFAB	-	$D0	Draw from 'DIFFY' style list (B,A)=(X,Y)	
Diffy	**$F3CE**	DIFFAX	-	$D0	Draw from 'DIFFY' style list X=List	

68000 Tutorial List
Learn 68000 Assembly
Hello World Series
Platform Specific Series
Grime 68000
68000 Downloads
68000 Cheatsheet
Sources.7z
DevTools kit
68000 Platforms
Amiga 500
Atari ST
Neo Geo
Sega Genesis / Mega Drive
Sinclair QL
X68000 (Sharp x68k)

8086 Content
Learn 8086 Assembly
Platform Specific Series
Hello World Series
8086 Downloads
8086 Cheatsheet
Sources.7z
DevTools kit
8086 Platforms
Wonderswan
MsDos

ARM Content
Learn ARM Assembly
Platform Specific Series
ARM Downloads
ARM Cheatsheet
Sources.7z
DevTools kit

Diffy	**$F3D8**	DIFFX	TDIFFY	$D0	Draw from 'DIFFY' style list A=Vectors-1 B=Scale X=List
Diffy	**$F3DD**	DIFFY	-	$D0	Draw from 'DIFFY' style list X=List
Diffy	**$F3D6**	DIFLST	-	$D0	Draw from 'DIFFY' style list
Diffy	**$F616**	DIFROT	ADROT	-	Rotate ⍰DIFFY⍰ style list X=Source U=Dest ANGLE
Diffy	**$F3D2**	DIFTIM	-	$D0	Draw from 'DIFFY' style list B=scale X=List
Diffy	**$F3DA**	DIFTLS	LDIFFY	$D0	Draw from 'DIFFY' style list A=Vectors-1, X=List
Diffy	**$F613**	DISROT	BDROT	-	'DIFFY' style rotate B=vector count X=source list U=Dest
Duffy	**$F3BE**	DUFFAB	-	$D0	Move a single vector from the current beam position using the relative vector values given in ⍰D⍰ (X,Y)=(B,A)
Duffy	**$F3AD**	DUFFAX	-	$D0	Draw from 'DUFFY' style list X=list
Duffy	**$F3BC**	DUFFY	-	$D0	Draw from 'DUFFY' style list X=list
Duffy	**$F3BC**	DUFLTLS	TDUFFY	$D0	Draw from 'DUFFY' style list B=length X=List
Duffy	**$F3B5**	DUFLST	-	$D0	Draw from 'DUFFY' style list X=list
Duffy	**$F3B1**	DUFTIM	-	$D0	Draw from 'DUFFY' style list B=length X=List
Diffy	**$F3B9**	DUFLST		$D0	A=Vectors-1 X=List
Dot	**$F2D5**	DIFDOT	-	$D0	Draw dots according to 'DIFFY' format X=List pointer
Dot	**$F2C5**	DOT	-	$D0	Turn on beam for dot
Dot	**$F2C3**	DOTAB	-	$D0	Draw Dot at relative (X,Y) pos (B,A)
Dot	**$F2DE**	DOTPAK	DOTPCK	$D0	Draw dots according to

ARM Platforms
Gameboy Advance
Nintendo DS
Risc Os

Risc-V Content
Learn Risc-V Assembly
Risc-V Downloads
Risc-V Cheatsheet
Sources.7z
DevTools kit

PDP-11 Content
Learn PDP-11 Assembly
PDP-11 Downloads
PDP-11 Cheatsheet
Sources.7z
DevTools kit

TMS9900 Content
Learn TMS9900 Assembly
TMS9900 Downloads
TMS9900 Cheatsheet
Sources.7z
DevTools kit
TMS9900 Platforms
Ti 99

6809 Content
Learn 6809 Assembly
6809 Downloads
6809/6309 Cheatsheet
Sources.7z
DevTools kit
6809 Platforms

						'PACKET' format X=list
Dot	**$F2BE**	DOTTIM	-	$D0		Draw one dot from 'DIFFY' style list B=on time X=List
Dot	**$F2C1**	DOTX	-	$D0		Draw one dot from 'DIFFY' style list X
Packet	**$F46E**	DASHY3	DASHPK	$D0		Draw dashed lines from 'PACKET' list X
Packet	**$F408**	PAC1X	PACK1X	$D0		Draw according to ⧫Packet⧫ style list X scale 1x
Packet	**$F404**	PAC2X	PACK2X	$D0		Draw according to ⧫Packet⧫ style list X scale 2x
Packet	**$F40E**	PACB	TPACK	$D0		Draw from 'PACKET' list X B=Scale
Packet	**$F410**	PACKET	-	$D0		Draw according to ⧫Packet⧫ style list X
Packet	**$F40C**	PACXX	LPACK	$D0		Draw from 'PACKET' style list X=list
Packet	**$F61F**	POTATA	PROT	-		'PACKET' style rotate A=Angle X=Source List U=Dest List
Packet	**$F622**	POTATE	APROT	-		'PACKET' style rotate
Raster	**$F37A**	POSDRAS	MSSPOS	$D0		Print $80 terminated String U at (X,Y) relative pos (B,A)
Raster	**$F378**	POSNRAS	RSTPOS	$D0		Display raster message from U
Raster	**$F498**	RASTER	MRASTR	$D0		Display raster string from MESSAGE
Raster	**$F495**	RASTUR	RASTER	$D0		Display raster string from U
Raster	**$F87C**	SADD	SCRADD	-		Add contents of score D to BCD at address X
Raster	**$F880**	SADD2	STKADD	-		Add stack to indicated score
Raster	**$F85E**	SHADD	BYTADD	-		Add contents of 'A' to indicated score
Raster	**$F391**	SHIPSAT	SHIPX	$D0		Display markers (count

A = 2-digit BCD number
X = Score field pointer

Dragon 32/Tandy Coco
Fujitsu FM7
TRS-80 Coco 3
Vectrex

My Game projects
Chibi Aliens
Chibi Akumas

Work in Progress
Learn 65816 Assembly
Learn eZ80 Assembly

Misc bits
Ruby programming

						remaining)
Raster	**$F393**	SHIPSHO	DSHIP	$C8		Display markers (count remaining) A=ascii B=remaining X=Pos
Raster	**$F373**	SIZPRAS	RSTSIZ	$D0		Display raster message from U
Raster	**$F38C**	TEXPOS	TXTPOS	$D0		Draw Strings from U (0 terminated)
Raster	**$F385**	TEXSIZ	TXTSIZ	$D0		Display raster message from U
Controller	**$F1B4**	ENPUT	DBNCE	$D0		Read controller switches and debounce switch status. (A=Response mask)
Controller	**$F8D8**	HIGHSCR	HISCR	-		Calculate high score and save for logo C=Score Field U=Highscore
Controller	**$F1BA**	INPUT	-	$D0		Read controller buttons
Controller	**$F7A9**	OPTION	SELOPT	-		Fetch number of players and options from player A + B = No of players (0-9)
Controller	**$F1F8**	PANG	JOYBIT	$D0		Read joystick UDLR
Controller	**$F1F5**	PBANG4	JOYSTK	$D0		Read the absolute position of the controller joysticks.
Controller	**$F84F**	SCLR	-	-		Clear indicated score X
Controller	**$F8C7**	WINNER	-	-		Determine highest score X or U
Rotate	**$F610**	DANROT	DROT	-		'DIFFY' style rotate
Rotate	**$F616**	DIFROT	ADROT	-		'DIFFY' style rotate
Rotate	**$F613**	DISROT	BDROT	-		'DIFFY' style rotate
Rotate	**$F61F**	POTATA	PROT	-		'PACKET' style rotate
Rotate	**$F622**	POTATE	APROT	-		'PACKET' style rotate X=Soource U=Dest (ANGLE)
Rotate	**$F5FF**	RATOT	LROT90	$C8		Rotate a single line A=Initial Y B=Angle

```
dc.b
Y,X,"TXT",$80
dc.b
Y,X,"TXT",$80
dc.b 0
```

Rotate	**$F603**	ROTAR	ALNROT	$C8	Rotate a single line A=Initial Y (ANGLE)
Rotate	**$F601**	ROTOR	LNROT	$C8	Rotate a single line A=Initial Y B=Angle
Maths	**$F584**	ABSVAL	ABSAB	-	Form absolute value for 'A' & 'B' registers
Maths	**$F58B**	AOK	ABSB	-	Form absolute value for 'B' register
Maths	**$F57E**	BITE	DECBIT	-	Decode bit position A=Bit number (0-7)
Maths	**$F593**	COMPAS	CMPASS	$C8	Return angle for given delta 'Y:X'
Maths	**$F5D9**	COSGET	COSINE	-	Calculate the cosine of 'A'
Maths	**$F511**	RAND3	-	-	Calculate new random number from SEED
Maths	**$F517**	RANDOM	-	-	Calculate new random number from SEED
Maths	**$F663**	RCOS	LCSINE	$C8	Multiply 'LEG' by previous cosine value
Maths	**$F661**	RCOSA	MCSINE	$C8	Multiply 'A' by previous cosine value
Maths	**$F65D**	RSIN	LSINE	$C8	Multiply 'LEG' by previous sine value
Maths	**$F65B**	RSINA	MSINE	$C8	Multiply 'A' by previous sine value WSINE
Maths	**$F5EF**	SINCOS	-	$C8	Calculate the sine and cosine of 'ANGLE'
Maths	**$F5DB**	SINGET	SINE	$C8	Calculate the sine of 'A'
Misc	**$F67F**	BAGAUX	BLKMV1	-	Transfer A+1 bytes (127 max) from source U destination X
Misc	**$F545**	CLR256		-	Clear 256 bytes at address X
Misc	**$F542**	CLRMEM	CLREX	-	Clear executive area of memory ($C800 - $C8FF)
Misc	**$F53F**	CLRSON	BCLR	-	Clear 'B' bytes from X
Misc	**$F55E**	DEKR	DECTMR	-	Decrement interval timers (XTMR0 ⮕ XTMR5)

Misc	**$F55A**	DEKR3	D3TMR	-	Decrement 3 interval timers (XTMR0 ⊡ XTMR2)	
Misc	**$F57A**	DEL	-	-	Delay execution for a minimum of 20 cycles (B=Delay)	
Misc	**$F57D**	DEL13	-	-	Delay execution for 13 cycles	
Misc	**$F579**	DEL20	-	-	Delay execution for 20 cycles	
Misc	**$F575**	DEL28	-	-	Delay execution for 28 cycles	
Misc	**$F571**	DEL33	-	-	Delay execution for 33 cycles	
Misc	**$F56D**	DEL38	-	-	Delay execution for 38 cycles	
Misc	**$F1AA**	DPIO	-	-	Set 6809 ⊡DP⊡ register for I/O accesses ($D0)	
Misc	**$F1AF**	DPRAM	-	-	Set 6809 ⊡DP⊡ register for RAM accesses ($C8)	
Misc	**$F552**	FILL	BLKFIL	-	Set a block of memory A=data B=bytes X=dest	
Misc	**$F8FF**	FINDBOX	BXTEST	-	Symmetric collison test	A = Box ⊡Y⊡ dimension (delta ⊡Y⊡) B = Box ⊡X⊡ dimension (delta ⊡X⊡) X = Y:X coordinates of point to be tested Y = Y:X coordinates of box center
Misc	**$F548**	GILL	CLRBLK	-	Clear D bytes of memory from X	

Recent New Content

Amiga - ASM PSET and POINT for Pixel Plotting

Learn 65816 Assembly: 8 and 16 bit modes on the 65816

SNES - ASM PSET and POINT for Pixel Plotting

ARM Assembly Lesson H3

Lesson P65 - Mouse reading on the Sam Coupe

Mouse Reading in MS-DOS

Risc-V Assembly Lesson 3 - Bit

Misc	**$F550**	NEGSOM	CLR80	-	Set B bytes at memory X to $80
Misc	**$F8E5**	OFF1BOX	OFF1BX	-	Off-center symmetric collision test
Misc	**$F8F3**	OFF2BOX	OFF2BX	-	Off-center symmetric collision text
Misc	**$F683**	STFAUX	BLKMOV	-	Transfer ⍰A⍰ bytes (0-127) from source ⍰U⍰ to destination ⍰X⍰
Sound	**$F92E**	AXE	EXPLOD	$C8	Complex explosion sound effect U=table
Sound	**$F9CA**	LOUDIN	SETAMP	$C8	Set amplitude in ⍰REQx⍰ B=Volume
Sound	**$F259**	PSG	WRPSG	$D0	Write to PSG A=PSG Addr B=PSG data X=mirror
Sound	**$F284**	PSGLPU	PSGMIR	$D0	Send sound string to PSG from U and mirror X
Sound	**$F27D**	PSGLUP	PSGLST	$D0	Send sound string to PSG from U
Sound	**$F256**	PSGX	WRREG	$D0	Write to PSG and Morror A=Psg Addr B=PSG Data
Sound	**$F687**	REPLAY	-	$C8	Set tune Sequence to list U
Sound	**$F289**	REQOUT	-	$D0	Send 'REQX' to PSG and mirror
Sound	**$F690**	SOPLAY	ASPLAY	$C8	Set tune sequence with alternate note set X=user not table U=Tune List
Sound	**$F68D**	SPLAY	-	$C8	Set tune Sequence to list U
Sound	**$F742**	XPLAY	-	$C8	Terminate current tune
Sound	**$F692**	YOPLAY	TPLAY	$C8	Set tune Sequence to list U

OS Vars

Address	Name	Details	Notes
$C800	REG0	Channel A: Fine tone period	
$C801	REG1	Channel A: Course tone period	
$C802	REG2	Channel B: Fine tone period	

$C803	REG3	Channel B: Course tone period
$C804	REG4	Channel C: Fine tone period
$C805	REG5	Channel C: Course tone period
$C806	REG6	Noise period
$C807	REG7	Tone / Noise enables
$C808	REG8	Channel A: Amplitude
$C809	REG9	Channel B: Amplitude
$C80A	REGA	Channel C: Amplitude
$C80B	REGB	Fine envelope period
$C80C	REGC	Course envelope period
$C80D	REGD	Envelope shape / cycle
$C80E	REGE	I/O port data register
$C80F	TRIGGR	Collective Switch Settings
$C811	EDGE (HEDGES)	Used by button handlers
$C812	KEY0	Controller #1 - Switch #0 (Leftmost)
$C813	KEY1	Controller #1 - Switch #1
$C814	KEY2	Controller #1 - Switch #2
$C815	KEY3	Controller #1 - Switch #3 (Rightmost)
$C816	KEY4	Controller #2 - Switch #0 (Leftmost)
$C817	KEY5	Controller #2 - Switch #1
$C818	KEY6	Controller #2 - Switch #2
$C819	KEY7	Controller #2 - Switch #3 (Rightmost)
$C81A	POTRES	Joystick resolution limit
$C81B	POT0	Joystick #1 - 'X' Axis
$C81C	POT1	Joystick #1 - 'Y' Axis
$C81D	POT2	Joystick #2 - 'X' Axis
$C81E	POT3	Joystick #2 - 'Y' Axis
$C81F	EPOT0 (DPOT0)	Controller #1: Right / left joystick pot enable (must be $00 or $01)
$C820	EPOT1 (DPOT1	Controller #1: Up / down joystick pot enable (must be $00 or $03)
$C821	EPOT2 (DPOT2)	Controller #2: Right / left joystick #2 enable (must be $00 or $05)
$C822	EPOT3 (DPOT3)	Controller #2: Up / down joystick #2 enable (must be $00 or $07)
$C823	LIST	Number Of Vectors to be drawn

$C824	ZSKIP	Flag controlling whether integrators will be zeroed:	Z=no NZ=Yes
$C825 - $C826	FRAME	Frame Counter	
$C827	TENSTY	Contains the last value used for the intensity setting	
$C828	DWELL	Dot ⍰ON⍰ time	
$C829	DASH	Dash pattern for drawing routines	
$C82A - $C82B	SIZRAS	Raster Message Size ($HHWW)	$F848=lrg $FC38 = sml
$C82C - $C82D	MESAGE	Used by string display functions to hold pointer to message to be displayed.	
$C82E	XTMR0 (X0)	?	
$C82F	XTMR1 (X1)	Countdown timer	
$C830	XTMR2 (X2)	Countdown timer	
$C831	XTMR3 (X3)	Countdown timer	
$C832	XTMR4 (X4)	Countdown timer	
$C833	XTMR5 (X5)	Countdown timer	
$C834	ABSY	Working storage for ⍰CMPASS⍰	
$C835	ABSX	Working storage for ⍰CMPASS⍰	
$C836	ANGLE	Angle for rotation	
$C837 - $C838	WSINE (SINE)	Location for parameter passing. Generally contains the last sine value calculated	
$C839 ⍰ C83A	WCSINE (COSINE)	Location for parameter passing. Generally contains the last cosine value calculated.	
$C83B	LEG	Executive Working Storage	
$C83C	LAG	Used by transformation functions	
$C83D - $C83E	FRMTIM (XMSEC)	Frame rate	
$C83F	REQ0	Envelope shape / cycle	
$C840	REQ1	Course envelope period	
$C841	REQ2	Fine envelope period	
$C842	REQ3	Channel C: Amplitude	
$C843	REQ4	Channel B: Amplitude	
$C844	REQ5	Channel A: Amplitude	
$C845	REQ6	Tone / noise enables	
$C846	REQ7	Noise period	
$C847	REQ8	Channel C: Course tone period	

$C848	REQ9	Channel C: Fine tone period
$C849	REQA	Channel B: Course tone period
$C84A	REQB	Channel B: Fine tone period
$C84B	REQC	Channel A: Course tone period
$C84C	REQD	Channel A: Fine tone period
$C84D - $C84E	DOREMI	Note table pointer
$C84F - $C850	FADE	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C851	VIBE	Working storage for tune player subroutines
$C853 - $C854	TUNE	?
$C855	NEWGEN	Working storage for tune player subroutines
$C856	TSTAT	Working storage for tune player subroutines
$C857	RESTC	Working storage for tune player subroutines
$C858	RATEA	Working storage for ⍰EXPLOD⍰
$C859	VIBA	Working storage for ⍰EXPLOD⍰
$C85A	RATEB	Working storage for ⍰EXPLOD⍰
$C85B	VIBB	Working storage for ⍰EXPLOD⍰
$C85C	RATEC	Working storage for ⍰EXPLOD⍰
$C85D	VIBC	Working storage for ⍰EXPLOD⍰
$C85E	FADEA	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C85F	FADEB	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C860	FADEC	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C861 - $C862	TONEA	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C863 - $C864	TONEB	Working storage for ⍰SELOPT⍰ and tune player subroutines
$C865 - $C866	TONEC	Working storage for tune player subroutines
$C86A	GAP	Allocated but not used by Mine Storm or Executive
$C86F - $C870	F1FREQ	Allocated but not used by Mine Store or Executive
$C80F - $C810	TRIGGR	Current controller button status.
$C867	SATUS	Working storage for ⍰EXPLOD⍰

$C868	LATUS	Allocated but not used by Mine Storm or Executive
$C869	XATUS	Allocated but not used by Mine Storm or Executive
$C86B - $C86C	B1FREQ	Allocated but not used by Mine Storm or Executive
$C86D - $C86E	B2FREQ	Allocated but not used by Mine Store or Executive
$C871	FEAST	Allocated but not used by Mine Storm or Executive
$C872	PEDGE	Allocated but not used by Mine Storm or Executive
$C873	NEDGE	Allocated but not used by Mine Storm or Executive
$C874 ⍰ $C875	K1FREQ	Allocated but not used by Mine Storm or Executive
$C876	BACON	Allocated but not used by Mine Storm or Executive
$C877	XACON	Working storage for ⍰EXPLOD⍰
$C878	SPEKT	Allocated but not used by Mine Store or Executive
$C879	PLAYRS (PLAYRZ)	Number of players ($01 - $09)
$C87A	OPTION (GAMZ)	Number of player options ($01 - $09)
$C87B - $C87C	SEED	Seed used by the random number generator
$C87D ⍰ C87F	RANCID	Working storage for random number generators
$C880	SBTN	First memory location available for use by a game
$C881	SJOY	Joystick mask (used by MineStorm)
$C883	ETMP1	Mine Storm: Temporary working storage
$C884	ETMP2	Mine Storm: Temporary working storage
$C885	ETMP3	Mine Storm: Temporary working storage
$C886	ETMP4	Mine Storm: Temporary working storage
$C887	ETMP5	Mine Storm: Temporary working storage
$C888	ETMP6	Mine Storm: Temporary working storage
$C889	ETMP7	Mine Storm: Temporary working storage
$C88A	ETMP8	Mine Storm: Temporary working storage
$C88B	ETMP9	Mine Storm: Temporary working storage
$C88C	ETMP10	Mine Storm: Temporary working storage
$C88F	TEMP1	Mine Storm: Temporary working storage
$C890	TEMP2	Mine Storm: Temporary working storage
$C891	TEMP3	Mine Storm: Temporary working storage

$C892	TEMP4	Mine Storm: Temporary working storage
$C893	TEMP5	Mine Storm: Temporary working storage
$C894	TEMP6	Mine Storm: Temporary working storage
$C895	TEMP7	Mine Storm: Temporary working storage
$C896	TEMP8	Mine Storm: Temporary working storage
$C897	TEMP9	Mine Storm: Temporary working storage
$C898 - $C89A	TEMP10	Mine Storm: Temporary working storage
$C89B	ACTPLY	Mine Storm: Currently active player ($00 or $02)
$C89C - $C89E	TMR1	Mine Storm: Timer
$C89F - $C8A1	TMR2	Mine Storm: Timer
$C8A2 - $C8A4	TMR3	Mine Storm: Timer
$C8A5 - $C8A7	TMR4	Mine Storm: Timer
$C8A8 - $C8AE	SCOR1	Holds player 1►s score
$C8AF ► C8B5	SCOR2	Holds player 2►s score
$C8EB - $C8F1	HISCOR (HEIGH)	Contains ASCII high score

Hardware Registers

$D000 8-bit port ►B► (Control bits) [CNTRL]
$D001 8-bit port ►A► (DAC and PSC data) [DAC]
$D002 Port ►B► direction control [DCNTRL]
$D003 Port ►A► direction control [DDAC]
$D004-7 Timer #1 [T1LOLC, T1HOC, T1LOL, T1HOL]
$D008-9 Timer #2 [T2LOLC, T2HOC]
$D00A Shift register [SHIFT]
$D00B Auxiliary control register [ACNTRL]
$D00C Peripheral control register [PCNTRL]
$D00D Interrupt flag register [IFLAG]
$D00E Interrupt enable register [IENABL]

bit modes on the 65816

SNES - ASM PSET and POINT for Pixel Plotting

ARM Assembly Lesson H3

Lesson P65 - Mouse reading on the Sam Coupe

Mouse Reading in MS-DOS

Risc-V Assembly Lesson 3 - Bit ops and more maths!

Mouse reading on the MSX

Hello World on RISC-OS

Atari 800 / 5200 - ASM PSET and POINT for Pixel Plotting

Apple 2 - ASM PSET and POINT for Pixel Plotting

Making a 6502 ASM Tron game... Photon1 - Introduction and Data Structures

Gaming + more:

Emily The Strange (DS) - Live full playthrough

$150 calculator: Unboxing the Ti-84 Plus CE (eZ80 cpu)

$D00F ORA

Character Map

The Vectrex has some character printing routines - they have a built in character map, which has upper case and a few system characters.	

Direct Page

The Vectrex uses two Direct page settings

$C8 points to RAM
$D0 points to Hardware registers

With ASW (Our assembler) use ASSUME dpr:$xx to tell the assembler where the direct page is... this is called SETDP on some assemblers.

AY Registers

The procedure for setting and reading AY registers is not quite direct... the method is shown here	**Select Register:** RegNum -> $D001 #$19 -> $D000 #$01 -> $D000	**Write Selected Register:** New Value -> $D001 #$11 -> $D000 #$01 -> $D000

Recent New Content

POINT for Pixel Plotting

Apple 2 - ASM PSET and POINT for Pixel Plotting

Making a 6502 ASM Tron game... Photon1 - Introduction and Data Structures

Gaming + more:

Emily The Strange (DS) - Live full playthrough

$150 calculator: Unboxing the Ti-84 Plus CE (eZ80 cpu)