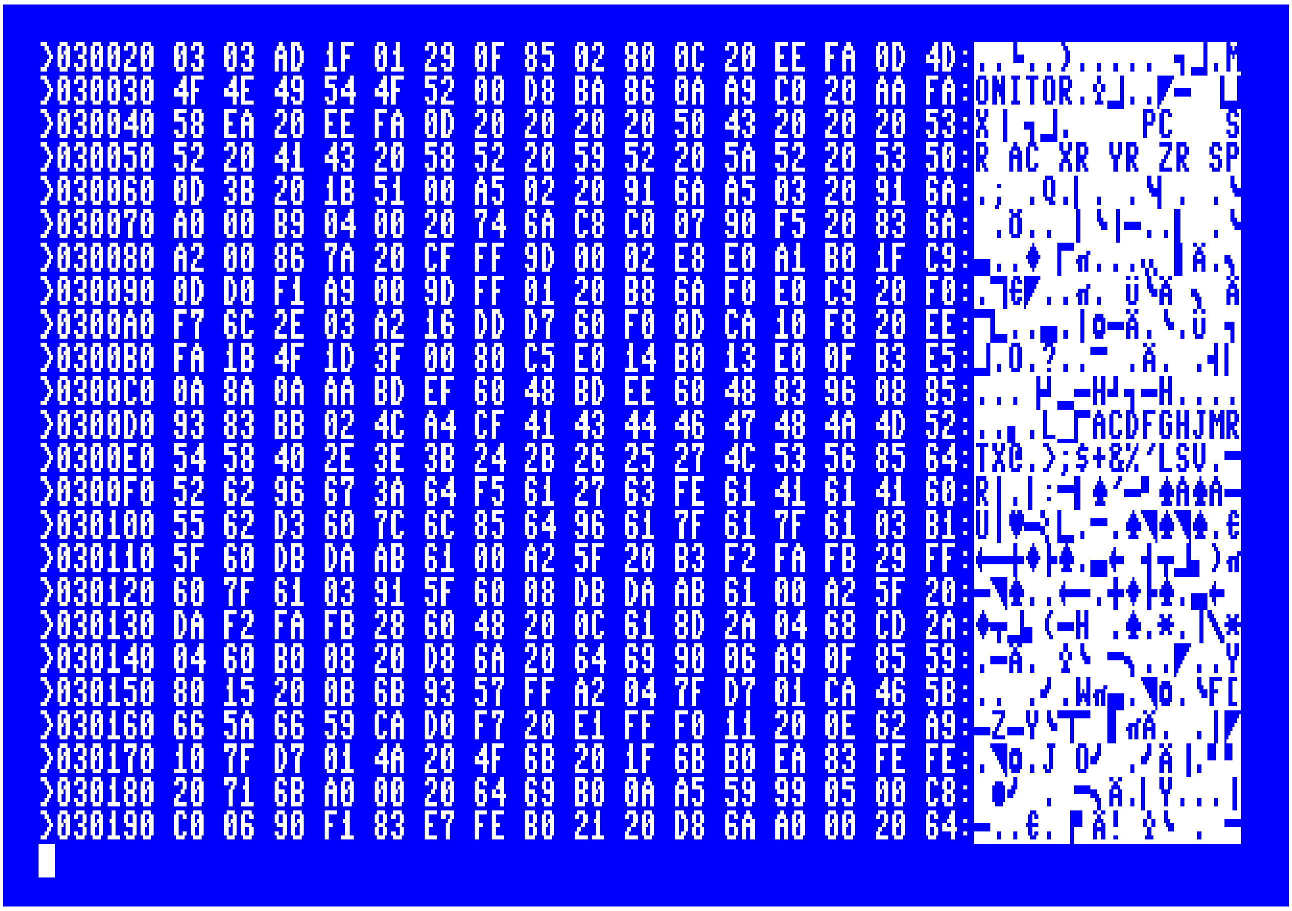
The (still uncomplete) C65 Memory Map

for the 911001 ROM



```
>030020 03 03 AD 1F 01 29 0F 85 02 80 0C 20 EE FA 0D 4D:
>030030 4F 4E 49 54 4F 52 00 D8 BA 86 0A A9 C0 20 AA FA:
>030040 58 EA 20 EE FA 0D 20 20 20 20 50 43 20 20 20 53:
>030050 52 20 41 43 20 58 52 20 59 52 20 5A 52 20 53 50:
>030060 0D 3B 20 1B 51 00 A5 02 20 91 6A A5 03 20 91 6A:
>030070 A0 00 B9 04 00 20 74 6A C8 C0 07 90 F5 20 83 6A:
>030080 A2 00 86 7A 20 CF FF 9D 00 02 E8 E0 A1 B0 1F C9:
>030090 0D D0 F1 A9 00 9D FF 01 20 B8 6A F0 E0 C9 20 F0:
>0300A0 F7 6C 2E 03 A2 16 DD D7 60 F0 0D CA 10 F8 20 EE:
>0300B0 FA 1B 4F 1D 3F 00 80 C5 E0 14 B0 13 E0 0F B3 E5:
>0300C0 0A 8A 0A AA BD EF 60 48 BD EE 60 48 83 96 08 85:
>0300D0 93 83 BB 02 4C A4 CF 41 43 44 46 47 48 4A 4D 52:
>0300E0 54 58 40 2E 3E 3B 24 2B 26 25 27 4C 53 56 85 64:
>0300F0 52 62 96 67 3A 64 F5 61 27 63 FE 61 41 61 41 60:
>030100 55 62 D3 60 7C 6C 85 64 96 61 7F 61 7F 61 03 B1:
>030110 5F 60 DB DA AB 61 00 A2 5F 20 B3 F2 FA FB 29 FF:
>030120 60 7F 61 03 91 5F 60 08 DB DA AB 61 00 A2 5F 20:
>030130 DA F2 FA FB 28 60 48 20 0C 61 8D 2A 04 68 CD 2A:
>030140 04 60 B0 08 20 D8 6A 20 64 69 90 06 A9 0F 85 59:
>030150 80 15 20 0B 6B 93 57 FF A2 04 7F D7 01 CA 46 5B:
>030160 66 5A 66 59 CA D0 F7 20 E1 FF F0 11 20 0E 62 A9:
>030170 10 7F D7 01 4A 20 4F 6B 20 1F 6B B0 EA 83 FE FE:
>030180 20 71 6B A0 00 20 64 69 B0 0A A5 59 99 05 00 C8:
>030190 C0 06 90 F1 83 E7 FE B0 21 20 D8 6A A0 00 20 64:
```

Version: 2020/11/09

This work mainly based on my playing around with the C65 ROM (911001).

Whenever possible, I have used the labels, names and comments that were used in the original unfinished C65 sourcecodes and appropriate expressions in public ROM listings for the C64 and C128 ROMs.

This memory map comes without any warranty of correctness.

This document is still under development and will be gradually expanded and corrected.

You can always get the latest version of this document at: 65site.de

Günther Reiter
65software@gmx.de
65site.de

Date: 2020/11/09

The C65 has 4 operation modes as described in the "C64DX SYSTEM SPECIFICATION":

------------ Quote from "C65 System Memory Layout" -----------------------------

```
C64 mode:      $E000-$FFFF     Kernel, Editor, Basic overflow area
---------      $D000-$DFFF     I/O and Color Nybbles, Character ROM
               $C000-$CFFF     Application RAM
               $A000-$BFFF     BASIC 2.2
               $0002-$9FFF     RAMLO.  VIC screen at $0400-$07FF
                               BASIC program & vars from $0800-$9FFF

C65 mode:      $E000-$FFFF     Kernel, Editor ROM code
---------      $D000-$DFFF     I/O and Color Bytes (CHRROM at $29000)
               $C000-$CFFF     Kernel Interface, DOS ROM overflow area
               $8000-$BFFF     BASIC 10.0 Graphics & Sprite ROM code
               $2000-$7FFF     BASIC 10.0 ROM code
               $0002-$1FFF     RAMLO.  VIC screen at $0800-$0FFF
                               BASIC prgs mapped from $02000-$0FF00
                               BASIC vars mapped from $12000-$1F7FF

C65 DOS mode:  $E000-$FFFF     Kernel, Editor ROM code
-------------  $D000-$DFFF     I/O (CIA's mapped out), Color Bytes
               $C800-$CFFF     Kernel Interface
               $8000-$C3FF     DOS ROM code
               $2000-$7FFF      (don't care)
               $0000-$1FFF     DOS RAMHI

C65 Monitor:   $E000-$FFFF     Kernel, Editor ROM code
------------   $D000-$DFFF     I/O and Color Bytes
               $C000-$CFFF     Kernel Interface
               $8000-$BFFF      (don't care)
               $6000-$7FFF     Monitor ROM code
               $0002-$5FFF     RAMLO
```

It's done this way for a reason. The CPU MAPPER restricts the programmer to one offset for each 32Kbyte half of a 64Kbyte segment. For one chunk of ROM to MAP in another chunk with a different offset, it must do so into the other half of memory from which it is executing. The OS does this by never mapping the chunk of ROM at $C000-$DFFF, which allows this chunk to contain the Interface/MAP code and I/O (having I/O in context is usually desirable, and you can't map I/O anyhow). The Interface/MAP ROM can be turned on and off via VIC register $30, bit 5 (ROM @ $C000), and therefore does not need to be mapped itself. Generally, OS functions (such as the Kernel, Editor, and DOS) live in the upper 32K half of memory, and applications such as BASIC or the Monitor) live in the lower 32K half. For example, when Monitor mode is entered, the OS maps out BASIC and maps in the Monitor. Each has ready access to the OS, but no built-in access to each other. When a DOS call is made, the OS overlays itself with the DOS (except for the magical $C000 code) in the upper 32K half of memory, and overlays the application area with DOS RAM in the lower 32K half of memory.

(Source: "C64DX SYSTEM SPECIFICATION" (C) 1991 by Commodore Business Machines)

Content:
==========

Addr.	dec	Name	Comment
			; #### PAGE ZERO ####
$0000	0	d6510	; 6510 data direction register
$0001	1	r6510	; 6510 data register
$0002	2	bank	($02-$09) used by JSR_FAR, JMP_FAR for ...
$0003	3	pc high	;
$0004	4	pc low	;
$0005	5	s	;
$0006	6	a	;
$0007	7	x	;
$0008	8	y	;
$0009	9	z	;
			; ==== BASIC 10
$000A	10	charac	used by math routines
$000B	11	endchr	used by math routines
$000C	12	verchk	flag LOAD or VERIFY
$000D	13	count	temp
$000E	14	dimflg	DIM flag used by variable search
$000F	15	valtyp	$0 = numeric / $FF = string
$0010	16	intflg	b7: (0=float, 1=integer); b6: (1=get flag)
$0011	17	dores	b7: P1LINE quote flag
$0012	18	subflg	b7: subscript flag (set to disallow subscripts() and integers%)
$0013	19	input_flag	READ($98), GET($40), INPUT($00)
$0014	20	tansgn	
$0015	21	channl	active I/O channel
$0016	22	linnum low	line number
$0017	23	linnum high	;
$0018	24	temppt	pointer to next temp. descriptor in tempst
$0019	25	lastpt low	pointer to last used temporary string
$001A	26	lastpt high	;
$001B	27	tempst	($1B-$23) temp. descriptor pointers (3 at 3 bytes each)
$001C	28	;	;
$001D	29	;	;
$001E	30	;	;
$001F	31	;	;
$0020	32	;	;
$0021	33	;	;
$0022	34	;	;
$0023	35	tempst	($1B-$23) temp. descriptor pointers (3 at 3 bytes each)
$0024	36	index1 low	index1
$0025	37	index1 high	;
$0026	38	index2 low	index2
$0027	39	index2 high	;
$0028	40	resho	multiplicand; 2 bytes for unsigned int. multiply
$0029	41	resmoh	;
$002A	42	resmo	product; addend; 3 bytes for unsigned int multiply
$002B	43	reslo low	;
$002C	44	reslo high	;

Addr.	dec	Name	Comment
$002D	45	txttab low	start address of BASIC program
$002E	46	txttab high	;
$002F	47	vartab low	start address of variable descriptors
$0030	48	vartab high	;
$0031	49	arytab low	start address of array table
$0032	50	arytab high	;
$0033	51	strend low	end of array table
$0034	52	strend high	;
$0035	53	fretop low	bottom of string storage
$0036	54	fretop high	;
$0037	55	frespc low	start address of strings
$0038	56	frespc high	;
$0039	57	max_mem_1 low	highest address available to BASIC in RAM 1
$003A	58	max_mem_1 high	;
$003B	59	curlin low	current BASIC line number
$003C	60	curlin high	;
$003D	61	txtptr low	pointer to BASIC text used by CHRGET, etc.
$003E	62	txtptr high	;
$003F	63	fndpnt low	pointer to item found by search
$0040	64	fndpnt high	;
$0041	65	datlin low	;
$0042	66	datlin high	;
$0043	67	datptr low	;
$0044	68	datptr high	;
$0045	69	inpptr low	;
$0046	70	inpptr high	;
$0047	71	varnam low	;
$0048	72	varnam high	;
$0049	73	varpnt low	;
$004A	74	varpnt high	;
$004B	75	forpnt low	;
$004C	76	forpnt high	;
$004D	77	opptr low	;
$004E	78	opptr high	;
$004F	79	opmask	;
$0050	80	defpnt low	;
$0051	81	defpnt high	;
$0052	82	dscpnt low	;
$0053	83	dscpnt high	;
$0054	84	trmpos	temp used by SPC(), TAB()
$0055	85	helper	P1LINE flag: ; bit7: HELP vs. LIST ; bit6: memory vs. file ; bit5: FIND/CHANGE ; bit4: highlight tokens ; bit3: highlight REM ; bit1: LINGET flag for AUTOSCROLL ; bit0: token in progress
$0056	86	jmper low	3 locations used by Function handler
$0057	87	jmper high	;
$0058	88	oldov	;
$0059	89	tempf1	used by math routines
$005A	90	highds low	;
$005B	91	highds high	;

Addr.	dec	Name	Comment
$005C	92	hightr low	;
$005D	93	hightr high	;
$005E	94	tempf2	used by math routines
$005F	95	deccnt low	;
$0060	96	deccnt high	;
$0061	97	lowtr	;
$0062	98	expsgn	;
$0063	99	facexp	floating point accumulator (primary) FAC1
$0064	100	facho	;
$0065	101	facmoh	;
$0066	102	facmoh	;
$0067	103	faclo	;
$0068	104	facsgn	;
$0069	105	sgnflg	
$006A	106	argexp	floating point accumulator (secondary) FAC2
$006B	107	argho	;
$006C	108	argmoh	;
$006D	109	argmo	;
$006E	110	arglo	;
$006F	111	argsgn	;
$0070	112	arisgn	;
$0071	113	facov	;
$0072	114	fbufpt low	;
$0073	115	fbufpt high	;
$0074	116	autinc low	incremental value for AUTO (0 = off)
$0075	117	autinc high	;
$0076	118	z_p_temp_1	leading zero counter for USING, temporary used by GET, RENUMBER, KEY, MOVSPR, SPRITE, PLAY, VOL,MID$
$0077	119	keysiz	;
$0078	120	syntmp	temporary used all over the place
$0079	121	dsdesc1	3 bytes descriptor for DS$
$007A	122	dsdesc2	;
$007B	123	dsdesc3	;
$007C	124	tos low	top of run time stack
$007D	125	tos high	;
$007E	126	runmod	flags: run/direct (b7), load (b6), trace (b5), edit (b4)
$007F	127	parsts	DOS parser status word
$0080	128	parstx	DOS parser status extensions
$0081	129	oldstk	BASIC saves uP stack pointer here
$0082	130	text_top low	top of BASIC text pointer
$0083	131	text_top high	;
$0084	132	text_bank	bank of BASIC text (default 0)
$0085	133	var_bank	bank of BASIC variables (default 1)
$0086	134	sid_speed_flag	saves system speed during SID ops (used during IRQ)
$0087	135	column	temp for FIND/CHANGE,[L]INPUT,[L]READ, CURSOR
$0088	136	fstr1 a	;
$0089	137	fstr1 b	;
$008A	138	fstr1 c	;
$008B	139	fstr2 a	;
$008C	140	fstr2 b	;
$008D	141	fstr2 c	;

Addr.	dec	Name	Comment
$008E	142		???
$008F	143		???
			; ==== kernel/editor base page allocations
$0090	144	status	serial bus status byte
$0091	145	stkey	stop key flag
$0092	146	partial	signals partial close for close
$0093	147	verify	signals verify mode for load
$0094	148	buf_flag	serial buffered char flag
$0095	149	bsour	char buffer for serial
$0096	150	serial	fast serial internal/external flag
$0097	151	count	temp used by serial routines
$0098	152	ldtnd	index to end of logical file table
$0099	153	dfltn	current (default) input device number
$009A	154	dflto	current (default) output device number
$009B	155	keyboard_lock	locks keyboard during scan
$009C	156	cmp_byte	used temp by verify routines
$009D	157	msgflg	os message flag enable
$009E	158	t1	temporary 1, used by load, save and editor keyset
$009F	159	t2	temporary 2, used by load, save and editor keyset
$00A0	160	vicIRQ	VIC IRQ flags at time of IRQ
$00A1	161	vic48	VIC reg 48 at time of IRQ
$00A2	162	DOS_flag	internal/external DOS device
$00A3	163	EOI_flag	temp used by serial routine
$00A4	164	bsour1	temp used by serial routine
$00A5	165	rs232_temp	temp used by rs232 OPEN
$00A6	166	rsstat	traditional c64/128 (6551) status byte
$00A7	167	rs232_status	rs232 status byte
$00A8	168	rs232_flags	rs232 open flag, xon/xoff status
$00A9	169	rs232_jam	rs232 system character
$00AA	170	rs232_xon_char	rs232 XON character (null=disabled)
$00AB	171	rs232_xoff_char	rs232 XOFF character (null=disabled)
$00AC	172	sa low	start of BASIC program
$00AD	173	sa high	;
$00AE	174	ea low	end of BASIC program
$00AF	175	ea high	;
$00B0	176	rs232_xmit_empty	rs232 xmit buffer empty flag (for close)
$00B1	177	rs232_rcvr_buffer_lo	rs232 lowest page of input buffer
$00B2	178	rs232_rcvr_buffer_hi	rs232 highest page of input buffer
$00B3	179	rs232_rmit_buffer_lo	rs232 lowest page of output buffer
$00B4	180	rs232_rmit_buffer_hi	rs232 highest page of output buffer
$00B5	181	rs232_high_water	rs232 point at which receiver XOFFs
$00B6	182	rs232_high_water	rs232 point at which receiver XONs
$00B7	183	fnlen	length current file n str
$00B8	184	la	current file logical addr
$00B9	185	sa	current file second addr
$00BA	186	fa	current file primary addr
$00BB	187	fnadr low	address of current file name str
$00BC	188	fnadr high	;
$00BD	189	ba	bank of current load/save/verify operation
$00BE	190	fnbank	bank where current filename is found (at 'fnadr')

Addr.	dec	Name	Comment
$00BF	191	vicbank	bank for VIC ; (0=internal, else external expansion RAM)
$00C0	192	sta low	save (start address)
$00C1	193	sta high	;
$00C2	194	memuss low	alternate load address, also used by vector routine
$00C3	195	memuss high	;
$00C4	196	rs232_rcvr_head low	rs232 pointer to end of buffer RCVR
$00C5	197	rs232_rcvr_head high	;
$00C6	198	rs232_rcvr_tail low	rs232 pointer to start of buffer RCVR
$00C7	199	rs232_rcvr_tail high	;
$00C8	200	rs232_xmit_head low	rs232 pointer to end of buffer XMIT
$00C9	201	rs232_xmit_head high	;
$00CA	202	rs232_xmit_tail low	rs232 pointer to start of buffer XMIT
$00CB	203	rs232_xmit_tail high	;
			; ==== screen editor declarations
$00CC	204	keytab low	keyscan table pointer
$00CD	205	keytab high	;
$00CE	206	imparm low	PRIMM utility string pointer
$00CF	207	imparm high	;
$00D0	208	ndx	index to keyboard queue
$00D1	209	kyndx	pending function key flag
$00D2	210	keyidx	index into pending function key string
$00D3	211	shflag	keyscan shift key status
$00D4	212	sfdx	keyscan current key index
$00D5	213	lstx	keyscan last key index
$00D6	214	crsw	<cr> input flag
$00D7	215	mode	40/80 column mode flag: 0 = 80 column mode (def) / other = 40 column mode
$00D8	216	graphm	text/graphic mode flag
$00D9	217	charen	RAM/ROM VIC character fetch flag (bit-2)
			; following locations are shared by several editor routines
$00DA	218	bitmsk	temp. for TAB and line wrap routines
$00DA	218	sedsal low	pointers for MOVLIN
$00DB	219	sedsal high	;
$00DC	220	sedeal low	;
$00DD	221	sedeal high	;
$00DE	222	sedt1	SAVPOS
$00DF	223	sedt2	;
$00DA	218	keysiz	programmable key variables
$00DB	219	keylen	;
$00DC	220	keynum	;
$00DD	221	keynxt	;
$00DE	222	keyadr	;
$00DF	223	keybnk	;

Addr.	dec	Name	Comment
			; ==== local editor variables
$00E0	224	pnt low	pointer to current line (text)
$00E1	225	pnt high	;
$00E2	226	user low	pointer to current line (attribute)
$00E3	227	user high	;
$00E4	228	scbot	window lower limit
$00E5	229	sctop	window upper limit
$00E6	230	sclf	window left margin
$00E7	231	scrt	window right margin
$00E8	232	lsxp	current input column start
$00E9	233	lstp	current input line start
$00EA	234	indx	current input line end
$00EB	235	tblx	current cursor line
$00EC	236	pntr	current cursor column
$00ED	237	lines	maximum number of screen lines
$00EE	238	columns	maximum number of screen columns
$00EF	239	datax	current character to print
$00F0	240	lstchr	previous charcter printed (for <esc> test)
$00F1	241	color	current attribute to print (default = foreground color)
$00F2	242	tcolor	saved attribute to print ('insert' and 'delete')
$00F3	243	rvs	reverse mode flag (bit 7)
$00F4	244	qtsw	quote mode flag (bit 7)
$00F5	245	insrt	insert mode flag # chars to insert, 0=not insert mode
$00F6	246	insflg	autoinsert mode flag
$00F7	247	locks	disables <c=><shift>, <ctrl>-S, function keys
$00F8	248	scroll	disables screen scroll, line linker, autoscroll
$00F9	249	beeper	disables <ctrl>-G
$00FA	250	lintmp	temp. pointer to last line for LOOP4
$00FB	251	app_zp_1	can be used for application software
$00FC	252	app_zp_2	can be used for application software
$00FD	253	app_zp_3	can be used for application software
$00FE	254	app_zp_4	can be used for application software
$00FF	255	basic_reserved_zp	reserved for BASIC
			; #### PAGE ONE ####
$0100	256	bad_1	($0100-$010F) 16 bytes were used by BASIC 10
$0101	257	bad_2	;
$0102	258	bad_3	;
$0103	259	bad_4	;
$0104	260	bad_5	;
$0105	261	bad_6	;
$0106	262	bad_7	;
$0107	263	bad_8	;
$0108	264	bad_9	;
$0109	265	bad_10	;
$010A	266	bad_11	;
$010B	267	bad_12	;
$010C	268	bad_13	;
$010D	269	bad_14	;
$010E	270	bad_15	;

Addr.	dec	Name	Comment
$010F	271	bad_16	;
$0110	272	system_map_1	($0110-$0113) 4 bytes kernel/editor,I/O,BASIC, ram 0
$0111	273	system_map_2	;
$0112	274	system_map_3	;
$0113	275	system_map_4	;
$0114	276	monitor_map_1	($0114-$0117) 4 bytes kernel/editor,I/O,monitor, ram 0
$0115	277	monitor_map_2	;
$0116	278	monitor_map_3	;
$0117	279	monitor_map_4	;
$0118	280	dos_map_1	($0118-$011B) 4 bytes kernel/editor,I/O,DOS,ram 1
$0119	281	dos_map_2	;
$011A	282	dos_map_3	;
$011B	283	dos_map_4	;
$011C	284	enviroment_1	($011C-$011F) 4 bytes current memory configuration
$011D	285	enviroment_2	;
$011E	286	enviroment_3	;
$011F	287	enviroment_4	;
$0120	288	dma_lda_list_1	($0120-$012B) 12 bytes; list used by LDA_FAR routine
$0121	289	dma_lda_list_2	;
$0122	290	dma_lda_list_3	;
$0123	291	dma_lda_list_4	;
$0124	292	dma_lda_list_5	;
$0125	293	dma_lda_list_6	;
$0126	294	dma_lda_list_7	;
$0127	295	dma_lda_list_8	;
$0128	296	dma_lda_list_9	;
$0129	297	dma_lda_list_10	;
$012A	298	dma_lda_list_11	;
$012B	299	dma_lda_list_12	;
$012C	300	dma_sta_list_1	($012C-$0137) 12 bytes; list used by STA_FAR routine
$012D	301	dma_sta_list_2	;
$012E	302	dma_sta_list_3	;
$012F	303	dma_sta_list_4	;
$0130	304	dma_sta_list_5	;
$0131	305	dma_sta_list_6	;
$0132	306	dma_sta_list_7	;
$0133	307	dma_sta_list_8	;
$0134	308	dma_sta_list_9	;
$0135	309	dma_sta_list_10	;
$0136	310	dma_sta_list_11	;
$0137	311	dma_sta_list_12	;
$0138	312	dma_list_1	($0138-$0143) 12 bytes; list used by editor and monitor
$0139	313	dma_list_2	;
$013A	314	dma_list_3	;
$013B	315	dma_list_4	;
$013C	316	dma_list_5	;
$013D	317	dma_list_6	;
$013E	318	dma_list_7	;
$013F	319	dma_list_8	;

Addr.	dec	Name	Comment
$0140	320	dma_list_9	;
$0141	321	dma_list_10	;
$0142	322	dma_list_11	;
$0143	323	dma_list_12	;
$0144	324	basic_list_a_1	($0144-$014F) 12 bytes; list 1 reserved for BASIC
$0145	325	basic_list_a_2	;
$0146	326	basic_list_a_3	;
$0147	327	basic_list_a_4	;
$0148	328	basic_list_a_5	;
$0149	329	basic_list_a_6	;
$014A	330	basic_list_a_7	;
$014B	331	basic_list_a_8	;
$014C	332	basic_list_a_9	;
$014D	333	basic_list_a_10	;
$014E	334	basic_list_a_11	;
$014F	335	basic_list_a_12	;
$0150	336	basic_list_b_1	($0150-$015B) 12 bytes; list 2 reserved for BASIC
$0151	337	basic_list_b_2	;
$0152	338	basic_list_b_3	;
$0153	339	basic_list_b_4	;
$0154	340	basic_list_b_5	;
$0155	341	basic_list_b_6	;
$0156	342	basic_list_b_7	;
$0157	343	basic_list_b_8	;
$0158	344	basic_list_b_9	;
$0159	345	basic_list_b_10	;
$015A	346	basic_list_b_11	;
$015B	347	basic_list_b_12	;
$015C	348	dma_byte	source/destination for LDA/STA far DMA routines
$015D	349	stack_bottom	($015D-$01FF) 162 bytes; system stack
...	...	...	; BASIC inits stack pointer to $01FB, kernel to $01FF
$01FF	511	stack_top	;
			; ##### PAGE TWO #####
$0200	512	buf_1	($0200-$02A1) 161 bytes; line input buffer
...	...	...	; used by BASIC and monitor
$02A1	673	buf_161	;
$02B0	688	keyd_1	($02B0-$02BA) 10 bytes; IRQ keyboard buffer
...	...	...	;
$02BA	698	keyd_10	;
$02C0	704	bas_var_pnt_1	($02C0-$02F9) 57 bytes;
...	...	...	; used for BASIC vars & pointers
$02F9	761	bas_var_pnt_57	;
$02FA	762	iautoscroll low	;autoscroll vector passed through by editor
$02FB	763	iautoscroll high	;

Addr.	dec	Name	Comment
			; #### PAGE THREE ####
			; kernel indirect vectors
$0314	788	iirq low	IRQ
$0315	789	iirq high	;
$0316	790	ibrk low	BRK
$0317	791	ibrk high	;
$0318	792	inmi low	NMI
$0319	793	inmi high	;
$031A	794	iopen low	;
$031B	795	iopen high	;
$031C	796	iclose low	;
$031D	797	iclose high	;
$031E	798	ichkin low	;
$031F	799	ichkin high	;
$0320	800	ickout low	;
$0321	801	ickout high	;
$0322	802	iclrch low	;
$0323	803	iclrch high	;
$0324	804	ibasin low	;
$0325	805	ibasin high	;
$0326	806	ibsout low	;
$0327	807	ibsout high	;
$0328	808	istop low	;
$0329	809	istop high	;
$032A	810	igetin low	;
$032B	811	igetin high	;
$032C	812	iclall low	;
$032D	813	iclall high	;
$032E	814	exmon low	monitor command indirect
$032F	815	exmon high	;
$0330	816	iload low	;
$0331	817	iload high	;
$0332	818	isave low	;
$0333	819	isave high	;
$0334	820	italk low	IEEE indirects for DOS
$0335	821	italk high	;
$0336	822	ilisten low	;
$0337	823	ilisten high	;
$0338	824	italksa low	;
$0339	825	italksa high	;
$033A	826	isecond low	;
$033B	827	isecond high	;
$033C	828	iacptr low	;
$033D	829	iacptr high	;
$033E	830	iciout low	;
$033F	831	iciout high	;
$0340	832	iuntalk low	;
$0341	833	iuntalk high	;
$0342	834	iunlisten low	;
$0343	835	iunlisten high	;

Addr.	dec	Name	Comment
			; editor indirect vectors to routines and tables
$0344	836	ctlvec low	'contrl' characters
$0345	837	ctlvec high	;
$0346	838	shfvec low	'shiftd' characters
$0347	839	shfvec high	;
$0348	840	escvec low	'escape' characters
$0349	841	escvec high	;
$034A	842	keyvec low	post keyscan, pre-evaluation of keys
$034B	843	keyvec high	;
$034C	844	keychk low	post-evaluation, pre-buffering of keys
$034D	845	keychk high	;
$034E	846	decode_1	($034E-$0359) 12 bytes; vectors to
$034F	847	decode_2	; keyboard matrix decode tables
$0350	848	decode_3	;
$0351	849	decode_4	;
$0352	850	decode_5	;
$0353	851	decode_6	;
$0354	852	decode_7	;
$0355	853	decode_8	;
$0356	854	decode_9	;
$0357	855	decode_10	;
$0358	856	decode_11	;
$0359	857	decode_12	;
			; kernel I/O channel tables:
$035A	858	lat_1	($035A-$0363) 10 bytes; logical file numbers
$035B	859	lat_2	;
$035C	860	lat_3	;
$035D	861	lat_4	;
$035E	862	lat_5	;
$035F	863	lat_6	;
$0360	864	lat_7	;
$0361	865	lat_8	;
$0362	866	lat_9	;
$0363	867	lat_10	;
$0364	868	fat_1	($0364-$036D) 10 bytes; primary device numbers
$0365	869	fat_2	;
$0366	870	fat_3	;
$0367	871	fat_4	;
$0368	872	fat_5	;
$0369	873	fat_6	;
$036A	874	fat_7	;
$036B	875	fat_8	;
$036C	876	fat_9	;
$036D	877	fat_10	;
$036E	878	sat_1	;
$036F	879	sat_2	;
$0370	880	sat_3	;
$0371	881	sat_4	;
$0372	882	sat_5	;
$0373	883	sat_6	;
$0374	884	sat_7	;

Addr.	dec	Name	Comment
$0375	885	sat_8	;
$0376	886	sat_9	;
$0377	887	sat_10	;
			;kernel ‘common’ RAM code area
$0380	896	ram_code_1	($0380-$03FB) 123 bytes;
...	...	...	; downloaded RestoreSystem, JMP/JSR_FAR routines
$03FB	1019	ram_code_123	;
$03FC	1020	config_1	($03FC-$03FF) 4 bytes; memory configuration at BRK
$03FD	1021	config_2	;
$03FE	1022	config_3	;
$03FF	1023	config_4	;
			; ##### PAGE FOUR AND HIGHER #####
			; MONITOR absolute declarations
$0400	1024	xcnt_1	($0400-$041F) 32 bytes; compare buffer
...	...	...	;
$041F	1055	xcnt_32	;
$0420	1056	hulp_1	
...	...	...	
$0429	1065	hulp_10	
$042A	1066	mtemp	
$042B	1067	format	asm/dis
$042C	1068	length	
$042D	1069	msal_1	for assembler
$042E	1070	msal_2	;
$042F	1071	msal_3	;
$0430	1072	sxreg	temp used all over
$0431	1073	syreg	temp used all over
$0432	1074	wrap	temp for assembler
$0433	1075	number	parse number conversion
$0434	1076	shift	parse number conversion
$0435	1077	numbers	
$0436	1078	opcode	
$0437	1079	hash_pointer	
$0438	1080	operand_mask	
$0439	1081	operand_size	
$043A	1082	temps	($043A-$05FF) 453 bytes temp
...	...	...	;
$05FF	1535	temps	;
$0600	1536	sprite_data_1	($0600-$07FF) 512 bytes; sprite definition area
...	...	...	;
$07FF	2047	sprite_data_512	;
$0800	2048	screen_1	($0800-$0FCF) 2000 bytes;
		...	; VIC-III 80 column text screen (80x25)
$0FCF	4047	screen_2000	;

Addr.	dec	Name	Comment
$0FD0	4048	vicsprite_ptr_1	($0FD0-$0FFF) 48 bytes;
...	...	...	; reserved for VIC spritepointers
$0FFF	4095	vicsprite_ptr_48	;
			;programmable function key definitions
$1000	4096	pkybuf_1	($1000-$100F) 16 bytes;
...	...	...	; programmable function key lengths table
$100F	4111	pkybuf_16	;
$1010	4112	pkydef_1	($1010-$10FF) 239 bytes;
...	...	...	; programmable function key strings
$10FF	4351	pkydef_239	;
			; absolute kernel variables
$1100	4352	system_vector low	vector to restart system (usually BASIC warm start)
$1101	4353	system_vector high	;
$1102	4354	dejavu	kernel warm/cold init status byte
$1103	4355	palnts	PAL/NTSC system flag
$1104	4356	init_status	flags reset vs. NMI status for init routines
$1105	4357	save_status	saves BASICs init status during monitor call
$1106	4358	default_drive	default device number
$1107	4359	expansion	flags presence (amount?) of expansion RAM
$1108	4360	memstr low	pointer to bottom of available memory in system bank
$1109	4361	memstr high	;
$110A	4362	memsiz low	pointer to top of available memory in system bank
$110B	4363	memsiz high	;
$110C	4364	timer_1	decrementing binary frame counter
$110D	4365	timer_2	;
$110E	4366	timer_3	;
$110F	4367	palcnt	counter for PAL (jiffie adjustment) (unused?)
$1110	4368	speed	save system speed during serial bus ops
$1111	4369	vudeja	flags disk_sei call
$1112	4370	rs232_baud	rs232 baud rate index (0-15)
$1113	4371	rs232_word	rs232 word length (0-3: 8,7,6,5)
$1114	4372	rs232_parity	rs232 parity mode (b1: enable, b0: even/odd)
$1115	4373	rs232_stop	rs232 stop bits (unused for 6511 type UART)
$1116	4374	rs232_duplex	rs232 duplex mode
$1117	4375	rs232_xline	rs232 xline mode (0=3-wire, 1=xline)
			; global absolute editor declarations
$1118	4376	xmax	keyboard queue maximum size
$1119	4377	pause	<ctrl>-S flag
$111A	4378	rptflg	enable key repeats
$111B	4379	kount	delay between key repeats
$111C	4380	delay	delay before a key starts repeating
$111D	4381	lstshf	delay between <C=><shft> toggles
$111E	4382	dead_keys	national deadkey enable/pending key flag
$111F	4383	blnon	VIC cursor mode (blink, solid)
$1120	4384	blnsw	VIC cursor disabled
$1121	4385	blnct	VIC cursor color before blink
$1122	4386	gdbln	VIC cursor character before blink
$1123	4387	gdcol	VIC cursor color before blink

Addr.	dec	Name	Comment
$1124	4388	vm1	VIC text screen/character base pointer
$1125	4389	ldtb1_sa	high byte of sa of VIC screen (use with vm1 to move screen)
$1126	4390	mono	monochrome flag ($80 = Color disabled)
$1127	4391	tabmap_1	($1127-$1130) 10 bytes; bitmap of TAB stops
...	...		;
$1130	4400	tabmap_10	;
$1131	4401	bitabl_1	($1131-$1134) 4 bytes; bitmap of line wraps
$1132	4402	bitabl_2	;
$1133	4403	bitabl_3	;
$1134	4404	bitabl_4	;
$1135	4405	mouse_enable	mouse driver vars
$1136	4406	mouse_pointer	sprite pointer used by mouse driver
$1137	4407	opotx	;
$1138	4408	opoty	;
$1139	4409	newvalue	;
$113A	4410	oldvalue	;
$113B	4411	mouse_top	;margins for mouse pointer, assuming hot spot at sprite 0,0
$113C	4413	mouse_bottom	;
$113D	4414	mouse_left	;
$113E	4415	mouse_right	;
$113F	4416	autoscrollupchr	; normally crsrdn
$1140	4417	autoscrolldnchr	; normally crsrup
$1141	4418	save_cursor_column	esc^ saves x-position here
$1142	4419	save_cursor_line	esc^ saves y-position here
$1143	4420	save_input_column	esc^ where input began here
$1144	4421	save_input_line	;
			; temps for SOUND
$1160	4448	temp_time_lo	
$1161	4449	temp_time_hi	
$1162	4450	temp_max_lo	
$1163	4451	temp_max_hi	
$1164	4452	temp_min_lo	
$1165	4453	temp_min_hi	
$1166	4454	temp_direction	
$1167	4455	temp_step_lo	
$1168	4456	temp_step_hi	
$1169	4457	temp_freq_lo	
$116A	4458	temp_freq_hi	
$116B	4459	temp_pulse_lo	
$116C	4460	temp_pulse_hi	
$116D	4461	temp_waveform	
$116E	4462	pot_temp_1	temporaries for 'POT' function
$116F	4463	pot_temp_2	;
			($1170-$11FF) 143 bytes; used by BASIC
$1170	4464	oldlin low	BASIC storage
$1171	4465	oldlin high	;
$1172	4466	oldtxt low	BASIC storage
$1173	4467	oldtxt high	;

Addr.	dec	Name	Comment
$1174	4468	rndx_1	($1174-$1178) 5 bytes;
$1175	4469	rndx_2	; floating point representation of last random number
$1176	4470	rndx_3	;
$1177	4471	rndx_4	;
$1178	4472	rndx_5	;
			($1179-$117C) 4 bytes; shared temp. by various routines
$1179	4473	window_temp_1	window
$117A	4474	window_temp_2	;
$117B	4475	window_temp_3	;
$117C	4476	window_temp_4	;
$1179	4473	t3	dcat
$1179	4473	renum_tmp_1	renumber
$1179	4473	tmptxt low	do/loop
$117A	4474	tmptxt high	;
$117B	4475	t4	dcat
$117B	4475	renum_tmp_2	renumber
$117B	4475	tmplin low	do/loop
$117C	4476	tmplin high	;
			; BASIC/DOS interface vars
$117D	4477	dosofl low	BLOAD/BSAVE starting addr
$117E	4478	dosofl high	;
$117F	4479	dosofh low	BSAVE ending addr
$1180	4480	dosofh high	;
$1181	4481	dosla	DOS logical addr
$1182	4482	dosfa	DOS physical addr
$1183	4483	dossa	DOS secondary addr
			($1184-$1190) 13 bytes; set to zero each DOS call
$1184	4484	xcnt	DOS loop counter
$1185	4485	dosf1l	DOS filename 1 len
$1186	4486	dosds1	DOS disk drive 1
$1187	4487	dosf2l	DOS filename 2 len
$1188	4488	dosds2	DOS disk drive 2
$1189	4489	dosf2a low	DOS filename 2 addr
$118A	4490	dosf2a high	;
$118B	4491	dosrcl	DOS record length
$118C	4492	dosbnk	DOS load/save bank
$118D	4493	dosdid low	DOS ID identifier
$118E	4494	dosdid high	;
$118F	4495	dosflags	DOS flags 7:ID, 6:recover
$1190	4496	dossa_temp	temp storage for file's sa during RECORD command

Addr.	dec	Name	Comment
			($1191-) 67 bytes; buffer used by MOVSPR, SPRDEF, SAVSPR, and DOS parser
$1191	4497	xabs low	movspr_line calculations
$1192	4498	xabs high	;
$1193	4499	yabs low	;
$1194	4500	yabs high	;
$1195	4501	xsgn low	;
$1196	4502	xsgn high	;
$1197	4503	ysgn low	;
$1198	4504	ysgn high	;
$1199	4505	fct_1	;
$119A	4506	fct_2	;
$119B	4507	fct_3	;
$119C	4508	fct_4	;
$119D	4509	errval_1	;
...	...	...	;
$11D3	4563	errval_55	;
			($11D4-$11EB) 24 bytes; PRINT USING definitions & storage
$11D4	4564	pufill	print using fill symbol
$11D5	4565	pucoma	print using comma symbol
$11D6	4566	pudot	print using decimal point symbol
$11D7	4567	pumony	print using monetary symbol
$11D8	4568	bnr	pointer to begin #
$11D9	4569	enr	pointer to end #
$11DA	4570	dolr	dollar flag
$11DB	4571	flag	comma flag (also used by PLAY ?)
$11DC	4572	swe	counter
$11DD	4573	usgn	sign exponent
$11DE	4574	uexp	pointer to exponent
$11DF	4575	vn	number of digits before decimal point
$11E0	4576	chsn	justify flag
$11E1	4577	vf	number of positions before decimal point (field)
$11E2	4578	nf	number of positions after decimal point (field)
$11E3	4579	posp	+/- flag (field)
$11E4	4580	fesp	exponent flag (field)
$11E5	4581	etof	switch
$11E6	4582	cform	char counter (field)
$11E7	4583	sno	sign no
$11E8	4584	blfd	blank/star flag
$11E9	4585	begfd	pointer to begin of field
$11EA	4586	lfor	length of format
$11EB	4587	endfd	pointer to end of field
			; graphics & sprite vars
$11EC	4588	xpos low	current x position
$11ED	4589	xpos high	;
$11EE	4590	ypos low	current y position
$11EF	4591	ypos high	;
$11F0	4592	xdest low	x-coordinate destination
$11F1	4593	xdest high	;
$11F2	4594	ydest low	y-coordinate destination

Addr.	dec	Name	Comment
$11F3	4595	ydest high	;
$11F4	4596	numcnt	temp, usually coordinate type
$11F5	4597	vtemp1	used by sprite math stuff
$11F6	4598	vtemp2	;
$11F7	4599	vtemp3	misc. graphic temp storage
$11F8	4600	vtemp4	;
$11F9	4601	vtemp5	;
			; angle stuff (used by sprites)
$11FA	4602	angsgn	sign of angle
$11FB	4603	sinval low	sine of value of angle
$11FC	4604	sinval high	;
$11FD	4607	cosval low	cosine of value of angle
$11FE	4608	cosval high	;
$11FF	4609	savsiz_1	temp work locations for SSHAPE, SPRSAV, MOVSPR_TO
$1200	4610	savsiz_2	;
$1201	4611	savsiz_3	;
$1202	4612	savsiz_4	;
$1203	4613	sprtmp_1	temp for SPRSAV
$1204	4614	sprtmp_2	;
$1205	4615	sprite_data_1	($1205-$125C) 88 bytes; speed/direction tabl. for 8 sprites, 11 bytes each
...	...	...	; move ang/dist move line ; offset=0 b7=0+speed b7=1+speed ; 1 counter counter lo ; 2 angle sign hi ; 3,4 delta-X dir+min/max ; 5,6 delta-Y fct1 ; 7,8 total-X fct2 ; 9,10 total-Y error
$125C	4700	sprite_data_88	;
			; music stuff driving stereo SIDs, 3 voices each
$125D	4701	voices_1	($125D-$1268) 12 bytes;
...	...	...	; voice counters (activity flags)
$1268	4712	voices_12	;
$1269	4713	waveform_1	($1269-$) 6 bytes;
...	...	...	; waveforms for each voice
$126E	4718	waveform_6	;
$126F	4719	voice	; play note parameters
$1270	4720	octave	;
$1271	4721	sharp	;
$1272	4722	dnote	;
$1273	4723	tempo_rate	; duration of whole note 4/4 time = 24/rate
$1274	4724	pitch low	;
$1275	4725	pitch high	;
$1276	4726	ntime low	;
$1277	4727	ntime high	;
$1278	4728	filters1_1	; volume and filter parameters
$1279	4729	filters1_2	;
$127A	4730	filters1_3	;

Addr.	dec	Name	Comment
$127B	4731	filters1_4	;
$127C	4732	filters2_1	;
$127D	4733	filters2_2	;
$127E	4734	filters2_3	;
$127F	4735	filters2_4	;
$1280	4736	fltsav_1	temps
$1281	4737	fltsav_2	;
$1282	4738	fltsav_3	;
$1283	4739	fltsav_4	;
$1284	4740	fltflg	temp
$1285	4741	tonnum	tune envelope stuff
$1286	4742	tonval_1	;
$1287	4743	tonval_2	;
$1288	4744	tonval_3	;
$1289	4745	atktab_1	($1289-$1292) 10 bytes; tune envelopes (attack)
...	...	...	;
$1292	4754	atktab_10	;
$1293	4755	sustab_1	($1293-$129C) 10 bytes; tune envelopes (sustain)
...	...	...	;
$129C	4764	sustab_10	;
$129D	4765	wavtab_1	($129D-$12A6) 10 bytes; tune envelopes (wave)
...	...	...	;
$12A6	4774	wavtab_10	;
$12A7	4775	pulslw_1	($12A7-$12B0) 10 bytes; tune envelopes (pulse low)
...	...	...	;
$12B0	4784	pulslw_10	;
$12B1	4785	pulshi_1	($12B1-$12BA) 10 bytes; tune envelopes (pulse high)
...	...	...	;
$12BA	4794	pulshi_10	;
$12BB	4795	parcnt	temp: envelope
$12BC	4796	nibble	temp: envelope, filter
			; SOUND command stuff
$12BD	4797	sound_voice	;
$12BE	4798	sound_time_lo_1	;
$12BF	4799	sound_time_lo_2	;
$12C0	4800	sound_time_lo_3	;
$12C1	4801	sound_time_lo_4	;
$12C2	4802	sound_time_lo_5	;
$12C3	4803	sound_time_lo_6	;
$12C4	4804	sound_time_hi_1	;
$12C5	4805	sound_time_hi_2	;
$12C6	4806	sound_time_hi_3	;
$12C7	4807	sound_time_hi_4	;
$12C8	4808	sound_time_hi_5	;
$12C9	4809	sound_time_hi_6	;
$12CA	4810	sound_max_lo_1	;
$12CB	4811	sound_max_lo_2	;
$12CC	4812	sound_max_lo_3	;
$12CD	4813	sound_max_lo_4	;
$12CE	4814	sound_max_lo_5	;

Addr.	dec	Name	Comment
$12CF	4815	sound_max_lo_6	;
$12D0	4816	sound_max_hi_1	;
$12D1	4817	sound_max_hi_2	;
$12D2	4818	sound_max_hi_3	;
$12D3	4819	sound_max_hi_4	;
$12D4	4820	sound_max_hi_5	;
$12D5	4821	sound_max_hi_6	;
$12D6	4822	sound_min_lo_1	;
$12D7	4823	sound_min_lo_2	;
$12D8	4824	sound_min_lo_3	;
$12D9	4825	sound_min_lo_4	;
$12DA	4826	sound_min_lo_5	;
$12DB	4827	sound_min_lo_6	;
$12DC	4828	sound_min_hi_1	;
$12DD	4829	sound_min_hi_2	;
$12DE	4830	sound_min_hi_3	;
$12DF	4831	sound_min_hi_4	;
$12E0	4832	sound_min_hi_5	;
$12E1	4833	sound_min_hi_6	;
$12E2	4834	sound_direction_1	;
$12E3	4835	sound_direction_2	;
$12E4	4836	sound_direction_3	;
$12E5	4837	sound_direction_4	;
$12E6	4838	sound_direction_5	;
$12E7	4839	sound_direction_6	;
$12E8	4840	sound_step_lo_1	;
$12E9	4841	sound_step_lo_2	;
$12EA	4842	sound_step_lo_3	;
$12EB	4843	sound_step_lo_4	;
$12EC	4844	sound_step_lo_5	;
$12ED	4845	sound_step_lo_6	;
$12EE	4846	sound_step_hi_1	;
$12EF	4847	sound_step_hi_2	;
$12F0	4848	sound_step_hi_3	;
$12F1	4849	sound_step_hi_4	;
$12F2	4850	sound_step_hi_5	;
$12F3	4851	sound_step_hi_6	;
$12F4	4852	sound_freq_lo_1	;
$12F5	4853	sound_freq_lo_2	;
$12F6	4854	sound_freq_lo_3	;
$12F7	4855	sound_freq_lo_4	;
$12F8	4856	sound_freq_lo_5	;
$12F9	4857	sound_freq_lo_6	;
$12FA	4858	sound_freq_hi_1	;
$12FB	4859	sound_freq_hi_2	;
$12FC	4860	sound_freq_hi_3	;
$12FD	4861	sound_freq_hi_4	;
$12FE	4862	sound_freq_hi_5	;
$12FF	4863	sound_freq_hi_6	;
$1300	4864	basdos_buf_1	($1300-$13FF) 256 bytes; BASIC DOS buffer
...	...	...	;
$13FF	5119	bas_dos_buf_256	;

Addr.	dec	Name	Comment
$1400	5120	rs232_input_buf_1	($1400-$14FF) 256 bytes; RS-232 input buffer
...	...	...	;
$14FF	5375	rs232_input_buf_256	;
$1500	5376	rs232_output_buf_1	($1400-$14FF) 256 bytes; RS-232 output buffer
...	...	...	;
$15FF	5631	rs232_output_buf_256	;
			; Graphics Kernel Interface
$1F00	7936	GKI.parm1	($1F00-$1F10) 17 bytes; ml interface parm values
...	...	...	;
$1F10	7952	GKI.parm17	;
$1F11	7953	GKI.subparm1	subroutine parm values
$1F12	7954	GKI.subparm2	;
$1F13	7955	GKI.subparm3	;
$1F14	7956	GKI.subparm4	;
$1F15	7957	GKI.subparm5	;
$1F16	7958	GKI.temp1	($1F16-$1F26) 17 bytes; local variables within subroutines
...	...	...	;
$1F26	7974	GKI.temp17	;
			; $2000: beginning of bankable memory & ROM code overlays
$2000	8192	user_memory	; BASIC text starts here (kernel sets 'membot' here)
...	...	...	

Addr.	dec	Name	Comment
			; #### PAGE ZERO - DOS ####
$0000	0	bmpnt low	bit map pointer
$0001	1	bmpnt high	;
$0002	2	track	current track
$0003	3	sector	current sector
$0004	4	lindx	;logical index
$0005	5	result_1	;
$0006	6	result_2	;
$0007	7	result_3	;
$0008	8	result_4	;
$0009	9	accum_1	;
$000A	10	accum_2	;
$000B	11	accum_3	;
$000C	12	accum_4	;
$000D	13	accum_5	;
$000E	14	usrjmp low	vector for the user-definable USER command
$000F	15	usrjmp high	;
$0010	16	bit_flag	; bit_flag: ; ; bit7 - This routine is used by the command parser routine to indicate the status of bit7 of the byte being tested as a command set by PARSXQ, cleared by PARSXQ. ; ; bit6 - This bit informs the FDC's READSEC routine to only read in the 1st two bytes off of a sector and place them directly into TRACK and SECTOR variables. (assumed to be the track and sector links to the next sector. This bit is set by the MARKTS routine and is cleared by the PARSXQ which ensures this flag will be cleared each time a new command is received by the IP routines and by the MARKTS routine. ; ; bit5 - This bit informs the WUSED routine (allocates sectors in the BAM) not to generate an error message when it starts to allocate a sector and finds out that it is already in use. This bitis set by the MARKTS routine and is cleared by the MARKTS routine and by the PARSXQ routine to ensure this bitis cleared each time a new command is received by the DOS. ; ; bit4 - This routine informs the MARKTS routine to call the WUSED routine to allocate sectors when set, and to call FRETS when it is cleared. It is set, and cleared by the same method as bit5 above ; ; bit3 - This bit inform the AUTO INIT. routine not to init the disk that is about to be used by the DOS. This bit is not set by the DOS by any current method. ; ; bit2 - Used by scratch to denote number of files scratched is > 256 ; ; bit1 - Informs the PARSE routine that it has already found the '/' char. ; ; bit0 - This bit informs the PARTITION routine that this is the 1st filename that it has parsed for. See partition routine for more info. This bit is set and cleared by the PARTITION routine. It is also cleared by the PARSXQ routine to prevent problems with other cmds

Addr.	dec	Name	Comment
$0011	17	bit_flag_1	; bit_flag_1: ; ; bit7 - This bit when set inform the DOS that it is an active talker and is set by the HTALK routine. This bit is used by the error message routine, and the HSECOND routine for i/o. ; ; bit6 - This bit when set informs the dos that it is an active listener and is set by the HLISTEN routine. This bit is used by the error message routine, and the HACPTR routine for i/o. ; ; bit5 - Used to inform various routines that an UNSCRATCH command is in progress. It is set by the OPEN, and RESTORE routines. ; ; bit4 - This bit is used by the UNLISTEN communications routines to inform the IP routines when it is to perform an EOI operation with the C65. When set then NO EOI is to be done. ; ; bit3 - This bit when set informs the PUT routine that there is a filename ready to be parsed in the command buffer. ; ; bit2 - This bit when set informs the various drive number checks to allow drive number 1 to be in the command buffer. ; ; bit1 - This bit when set informs the HCIOUT routine that there is a byte in it's one character buffer. This is the same method used by the kernel's CIOUT routines label 'BUF_FLAG' (see bit 7) ; ; bit0 - This bit when set informs the various routines that a DIRECTORY LISTING is in progress.
$0012	18	bit_flag_2	; bit_flag_2: ; ; bit7 - reserved ; bit6 - reserved ; ; bit5 - This bit when set inform the DOS that big relative files are allowed to be used. It will also force the DOS to create a BIG rel file when it is first created. This bit is set from the IOBYT variable stored on the disk's 1st bam sector as well as by a user command. ; ; bit4 - reserved ; bit3 - reserved ; bit2 - reserved ; bit1 - reserved ; ; bit0 - This bit when set informs the FDC routine not to call the error routines upon an error, but to just place the error number in the the variable JOBS.
$0013	19	t0	; temp workspace
$0014	20	t1	;
$0015	21	t2	;
$0016	22	t3	;
$0017	23	t4	;
$0018	24	ip low	indirect ptr variable
$0019	25	ip high	;
$001A	26	r0	; temp workspace
$001B	27	r1	;
$001C	28	r2	;
$001D	29	r3	;
$001E	30	r4	;
$001F	31	dirbuf low	directory buffer pointer
$0020	32	dirbuf high	;
$0021	33	buftab_1	($0021-$004C) 44 bytes; buffer byte pointers
...	...	...	;
$004C	35	buftab_44	;

Addr.	dec	Name	Comment
$004D	77	cmd_buf_ptr low	cb. command buffer pointer
$004E	78	cmd_buf_ptr high	;
$004F	79	err_buf_ptr low	er. error buffer pointer
$0050	80	err_buf_ptr high	;
			; base page array
$0051	81	buf0_1	($0051-$005C) 12 bytes; buf0
..	...	...	; bit7 - Last byte in file reached ; bit6 - File is dirty (modified) ; bit5 - Record overflow has occured ; bit4 thru bit0 - Buffer number in use
$005C	92	buf0_12	;
$005D	93	buf1_1	($005D-$0068) 12 bytes; buf1
...	...	...	; bit7 - Last byte in file reached ; bit6 - File is dirty (modified) ; bit5 - Record overflow has occured ; bit4 thru bit0 - Buffer number in use
$0068	104	buf1_12	;
$0069	105	recl_1	($0069-$0074) 12 bytes; recl
...	...	...	;
$0074	116	recl_12	;
$0075	117	rech_1	($0075-$0080) 12 bytes; rech
...	...	...	;
$0080	128	rech_12	;
$0081	129	next_record_1	($0081-$008C) 12 bytes; next record
...	...	...	;
$008C	140	next_record_12	;
$008D	141	record_size_1	($008D-$0098) 12 bytes; record size
...	...	...	;
$0098	152	rec_size_12	;
$0099	153	side_sector_1	($0099-$00A4) 12 bytes; side sector
...	...	...	;
$00A4	164	side_sector_12	;
$00A5	165	filtyp_1	($00A5-$00B0) 12 bytes; file type
...	...	...	;Channel file type from the directory sector if none from user ; bit7 = Last Record Found (Rel files) ; bit6 = Record ammended flag ; bit5 = OVeR FLow flag ; bit4 = not used ; bit 3 2 1 ; 0 0 0 = DEL ; 0 0 1 = SEQ ; 0 1 0 = PRG ; 0 1 1 = USR ; 1 0 0 = REL ; 1 0 1 = CBM ; 1 1 0 = SPARE FILE TYPE ; 1 1 1 = DIRECT FILE TYPE (U1/2) ; bit0 = drive number
$00B0	176	filtyp_12	;
$00B1	177	chnrdy_1	($00B1-$00BC) 12 bytes; channel status byte
...	...		;Channel status byte ; bit7 - set then ready to talk ; bit3 - clear then send EOI ; bit0 - Direct channel access (U1, U2)
$00BC	188	chnrdy_12	;

Addr.	dec	Name	Comment
$00BD	189	lintab_1	($00BD-$00E2) 38 bytes; lintab
...	...	...	; bit 7 6 ; 0 0 then read only chnl ; 0 1 then read/write chnl ; 1 0 then write only chnl ; 1 1 then chnl inactive ; ; bit5 and bit4 are spare ; bit3 thru bit0 contain the LINDX value for this channel ; ; LINDX is allocated from $00 thru $0A, starting at zero
$00E2	226	lintab_38	;
$00E3	227	chndat_1	($00E3-$00EE) 12 bytes; channel data byte
...	...	...	;
$00EE	238	chndat_12	;
$00EF	239	lstchr_1	($00EF-$00EE) 12 bytes; channel last char ptr
...	...	...	;
$00FA	250	lstchr_12	;
$00FB	251	drvnum	current drive number
			; Note: Anything past this point is considered available for user programs.
$00FC	252	half	block (which half of 512-byte physical sector)
$00FD	253	erword low	error word for recovery, each drive
$00FE	254	erword high	;
$00FF	255	prgdrv	;last program drive
			; #### PAGE ONE - DOS ####
$0100	256	prgsec	last program sector
$0101	257	write_lindx	write lindx for copy routines
$0102	258	read_lindx	read lindx
$0103	259	wlindx	write lindx
$0104	260	nbtemp low	number blocks temp
$0105	261	nbtemp high	;
$0106	262	char	char under parser
$0107	263	limit	pointer to end of filename or next comma, =, /
$0108	264	f1cnt	file stream 1 count
$0109	265	f2cnt	file stream 2 count
$010A	266	f2ptr	file stream 2 pointer (maximum = MXFILS)
$010B	267	linuse low	; LINUSE is used to keep track of the logical index values that are currently in use. ; ; bit0 and bit1 of linuse+1 are always in use because they're the command and error channels. When another bit is set then it shows that lindx value to be in use and not available.
$010C	268	linuse high	;
$010D	269	cmdsiz	; command string size. If 'I0:1' is in the command buffer then CMDSIZ would contain a 4
$010E	270	prgtrk	last PRG file's track number
$010F	271	nodrv	NO DRiVe ready flag (BIT 7 drive 0 (internal). BIT 6 drive 1 (external))
$0110	272	curtrk low	current track under read/write heads

Addr.	dec	Name	Comment
$0111	273	curtrk high	;
$0112	274	fiptr	File stream 1 pointer (indexed by MAXFILS)
$0113	275	cbdosaddress low	Listen address
$0114	276	cbdosaddress high	;
$0115	277	tos	Top Of Stack for ERPROC routine
$0116	278	jobnum	current job number
$0117	279	entsec_1	($0117-0120) 10 bytes; sector of directory entry
...	...	...	;
$0120	288	entsec_10	;
$0121	289	entind_1	($0121-012A) 10 bytes; index of directory entry
...	...	...	;
$012A	298	entind_10	;
$012B	299	file_drive_1	($012B-$0134) 10 bytes; file_drive
...	...	...	; this variable contains the drive number located in the command buffer for this file. If no drive number existed then BIT 7 will be set to indicate the default drive number is to be used.
$0134	308	file_drive_10	;
$0135	309	pattyp_1	($0135-$0150) 28 bytes; pattyp
...	...	...	; bit7 = Wild cards in the filename ; bit6 = Save with replace flag ; bit5 = File closed correctly ; bit4 = not used ; bit 3 2 1 ; 0 0 0 = DEL ; 0 0 1 = SEQ ; 0 1 0 = PRG ; 0 1 1 = USR ; 1 0 0 = REL ; 1 0 1 = CBM ; 1 1 0 = SPARE FILE TYPE ; 1 1 1 = SPARE FILE TYPE ; bit0 = drive number
$0150	336	pattyp_28	;
$0151	337	filtbl_1	($0151-$016D) 29 bytes; filtbl
...	...	...	; used by various routines to point to the DRIVE number in the user given filename
$016D	365	filtbl_29	;
			; header
$016E	366	sz	; 2=512 byte sectors
$016F	367	sec	sector
$0170	368	sd	side
$0171	369	tt1	track
$0172	370	ttemp	used by CRC routine
$0173	371	crc_1	($0173-$0187) 21 bytes; CRC, 2 bytes per sector
...	...	...	;
$0187	391	crc_21	;
$0188	392	sa	secondary address
$0189	393	orgsa	original secondary address
$018A	394	data	temp data byte
$018B	395	recptr	record pointer
$018C	396	ssnum	side sector number
$018D	397	ssind	index into side sector
$018E	398	relptr	pointer into record
$018F	399	type	active file type
$0190	400	beginame	new pattern matching

Addr.	dec	Name	Comment
$0191	401	beginpat	;
$0192	402	dirtrk low	directory track
$0193	403	dirtrk high	;
$0194	404	dirst low	directory starting sector
$0195	405	dirst high	;
$0196	406	system_track low	system track number for disk formating check
$0197	407	system_track high	;
$0198	408	dchange	disk changed flag; ; b7:drive 0 (int.) / b6: drive 1 (ext.)
$0199	409	maxtrk low	max track
$019A	410	maxtrk high	;
$019B	411	startrk low	starting track number
$019C	412	startrk high	;
$019D	413	grpnum	group number
$019E	414	sssgrp_1	($019E-$01A9) 12 bytes; resident group
...	...	...	; 255 = no SSS or group resident ; 254 = 222 is resident in ram ; 0 thru 91 = number group is resident
$01A9	425	sssgrp_12	;
$01AA	426	ssssec_1	($01AA-$01B5) 12 bytes; super side sector address
...	...	...	;
$01B5	437	ssssec_12	;
$01B6	438	ssstrk_1	($01B6-$01C1) 12 bytes; super side sector track addr
...	...	...	;
$01C1	449	ssstrk_12	;
$01C2	450	r5	current group for big rel
$01C3	451	secinc	sector allocation
$01C4	452	lo	Used by the partition routines
$01C5	453	high	
$01C6	454	tmp	Used to parse directory entries vs cmd buf or to create a bam
$01C7	455	dirsecinc	directory sector increment value
$01C8	456	datasecinc	data sector increment value
$01C9	457	drivenumber	physical drive number
$01CA	458	dskver	disk dos version
$01CB	459	fmttyp	method disk was formatted under
$01CC	460	image	file stream image; ; is used by the TAGCMD routine to set up the various file stream flags for the command level routines to check for proper syntax. ; bit7 - wild cards present in file stream 1 (fs1) ; bit6 - more than one filename is present in fs1 ; bit5 - drive number was specified in fs1 ; bit4 - a file name exists in fs1 ; bit3 thru bit0 - are the same as bit7 thru bit4 ; but are for file stream 2
$01CD	461	drvcnt	number of drv searches
$01CE	462	drvflg	drive search flag
$01CF	463	control_store low	shadow control register
$01D0	464	control_store high	;
$01D1	465	last_side	last phys. side read by ReadSec
$01D2	466	last_track	last phys. track read by ReadSec
$01D3	467	last_sector	last phys. sector read by ReadSec
$01D4	468	last_drive	last phys. drive read by ReadSec

Addr.	dec	Name	Comment
$01D5	469	found	found flag in dir searches
$01D6	470	?	;
...	...		;
$01FF	511	?	;
			; #### PAGE TWO AND HIGHER - DOS ####
$0200	512	cmdbuf_1	($0200-$) 177 bytes; command input buffer
...	...	...	;
$02B0	688	cmdbuf_177	;
$02B1	689	nambuf_1	($02B1-$02D4) 36 bytes; directory buffer
...	...	...	;
$02D4	724	nambuf_36	;
$02D5	725	errbuf_d0_1	($02D5-$02FF) 43 bytes; error message buffer, first drive
...	...	...	;
$02FF	767	errbuf_d0_43	;
			; I/O buffers
$0300	768	dosbuffer0_1	($0300-$03FF) 256 bytes; I/O buffer number 0
...	...	...	;
$03FF	1023	dosbuffer0_256	;
$0400	1024	dosbuffer1_1	($0400-$04FF) 256 bytes; I/O buffer number 1
...	...	...	;
$04FF	1279	dosbuffer1_256	;
$0500	1280	dosbuffer2_1	($0500-$05FF) 256 bytes; I/O buffer number 2
...	...	...	;
$05FF	1535	dosbuffer2_256	;
$0600	1536	dosbuffer3_1	($0600-$06FF) 256 bytes; I/O buffer number 3
...	...	...	;
$06FF	1791	dosbuffer3_256	;
$0700	1792	dosbuffer4_1	($0700-$07FF) 256 bytes; I/O buffer number 4
...	...	...	;
$07FF	2047	dosbuffer4_256	;
$0800	2048	dosbuffer5_1	($0800-$08FF) 256 bytes; I/O buffer number 5
...	...	...	;
$08FF	2303	dosbuffer5_256	;
$0900	2304	dosbuffer6_1	($0900-$09FF) 256 bytes; I/O buffer number 6
...	...	...	;
$09FF	2559	dosbuffer6_256	;
$0A00	2560	dosbuffer7_1	($0A00-$0AFF) 256 bytes; I/O buffer number 7
...	...	...	;
$0AFF	2815	dosbuffer7_256	;
$0B00	2816	dosbuffer8_1	($0B00-$0BFF) 256 bytes; I/O buffer number 8
...	...	...	;
$0BFF	3071	dosbuffer8_256	;
$0C00	3072	dosbuffer9_1	($0C00-$0CFF) 256 bytes; I/O buffer number 9
...	...	...	;
$0CFF	3327	dosbuffer9_256	;
$0D00	3328	dosbuffer10_1	($0D00-$0DFF) 256 bytes; I/O buffer number 10
...	...	...	;
$0DFF	3583	dosbuffer10_256	;
$0E00	3584	dosbuffer11_1	($0E00-$0EFF) 256 bytes; I/O buffer number 11

Addr.	dec	Name	Comment
...	...	...	;
$0EFF	3839	dosbuffer11_256	;
$0F00	3840	dosbuffer12_1	($0F00-$0FFF) 256 bytes; I/O buffer number 12
...	...	...	;
$0FFF	4095	dosbuffer12_256	;
$1000	4096	dosbuffer13_1	($1000-$10FF) 256 bytes; I/O buffer number 13
...	...	...	;
$10FF	4351	dosbuffer13_256	;
$1100	4352	dosbuffer14_1	($1100-$11FF) 256 bytes; I/O buffer number 14
...	...	...	;
$11FF	4607	dosbuffer14_256	;
$1200	4608	dosbuffer15_1	($1200-$12FF) 256 bytes; I/O buffer number 15
...	...	...	;
$12FF	4863	dosbuffer15_256	;
$1300	4864	dosbuffer16_1	($1300-$13FF) 256 bytes; I/O buffer number 16
...	...	...	;
$13FF	5119	dosbuffer16_256	;
$1400	5120	dosbuffer17_1	($1400-$14FF) 256 bytes; I/O buffer number 17
...	...	...	;
$14FF	5375	dosbuffer17_256	;
$1500	5376	dosbuffer18_1	($1500-$15FF) 256 bytes; I/O buffer number 18
...	...	...	;
$15FF	5631	dosbuffer18_256	;
$1600	5632	dosbuffer19_1	($1600-$16FF) 256 bytes; I/O buffer number 19
...	...	...	;
$16FF	5887	dosbuffer19_256	;
$1700	5888	dosbuffer20_1	($1700-$17FF) 256 bytes; I/O buffer number 20
...	...	...	;
$17FF	6143	dosbuffer20_256	;
$1800	6144	dosbuffer21_1	($1800-$18FF) 256 bytes; I/O buffer number 21
...	...	...	;
$18FF	6399	dosbuffer21_256	;
			; Note: The following buffers are NOT available ; for the U1,U2 commands and if you attempt to ; access these buffers you will receive a 70,NO ; CHANNEL,00,00
$1900	6400	bambuf0d0_1	($1900-$19FF) 256 bytes; BAM 0 for drive 0
...	...	...	;
$19FF	6655	bambuf0d0_256	;
$1A00	6656	bambuf1d0_1	($1A00-$1AFF) 256 bytes; BAM 1 for drive 0
...	...	...	;
$1AFF	6911	bambuf1d0_256	;
$1B00	6912	bambuf0d1_1	($1B00-$1BFF) 256 bytes; BAM 0 for drive 1
...	...	...	;
$1BFF	7167	bambuf0d1_256	;
$1C00	7168	bambuf1d1_1	($1C00-$1CFF) 256 bytes; BAM 1 for drive 1
...	...	...	;
$1CFF	7423	bambuf1d1_256	;
$1D00	7424	userbuffer0_1	($1D00-$1DFF) 256 bytes; FDC 1st. sector buffer
...	...	...	;
$1DFF	7679	userbuffer0_256	;

Addr.	dec	Name	Comment
$1E00	7680	userbuffer1_1	($1E00-$1EFF) 256 bytes; FDC 2nd. sector buffer
...	...	...	;
$1EFF	7935	userbuffer1_256	;
			; miscellaneous
$1F00	7936	cmdnum	number of commands in buffer
$1F01	7937	strsiz	size of command string
$1F02	7938	tempsa	temporary sa
$1F03	7939	cmd	temporary job command
$1F04	7940	errbuf_d1_1	($1F04-$1F2E) 43 bytes; error message buffer, 2nd drive
...	...	...	;
$1F2E	7982	errbuf_d1_43	;
$1F2F	7983	bufuse_1	; --- ; bufuse_1 ; --- ; Bits 7 6 5 4 3 2 1 0 ; ; BUFFER# 7 6 5 4 3 2 1 0 ; ^ ^ ^ ^ ^ ^ ^ ^ ; : : : : : : : :---DOS buffer 0 ; : : : : : : :------DOS buffer 1 ; : : : : : :---------DOS buffer 2 ; : : : : :------------DOS buffer 3 ; : : : :---------------DOS buffer 4 ; : : :------------------DOS buffer 5 ; : :---------------------DOS buffer 6 ; :------------------------DOS buffer 7 ; ; --- ; bufuse_2 ; --- ; Bits 7 6 5 4 3 2 1 0 ; ; BUFFER# F E D C B A 9 8 ; ^ ^ ^ ^ ^ ^ ^ ^ ; : : : : : : : :---DOS buffer 8 ; : : : : : : :------DOS buffer 9 ; : : : : : :---------DOS buffer 10 ; : : : : :------------DOS buffer 11 ; : : : :---------------DOS buffer 12 ; : : :------------------DOS buffer 13 ; : :---------------------DOS buffer 14 ; :------------------------DOS buffer 15 ; ; --- ; bufuse_3 ; --- ; Bits 7 6 5 4 3 2 1 0 ; ; BUFFER# 17 16 15 14 13 12 11 10 (hex) ; ^ ^ ^ ^ ^ ^ ^ ^ ; : : : : : : : :---DOS buffer 16 ; : : : : : : :------DOS buffer 17 ; : : : : : :---------DOS buffer 18 ; : : : : :------------DOS buffer 19 ; : : : :---------------DOS buffer 20 ; : : :------------------DOS buffer 21 ; : :---------------------NOT USABLE ; :------------------------NOT USABLE ; ; --- ; ; buffer use flags, if bit is 0 then buffer is free

Addr.	dec	Name	Comment
$1F30	7984	bufuse_2	;
$1F31	7985	bufuse_3	;
$1F32	7986	dskid_1	current disk ID's
$1F33	7987	dskid_2	;
$1F34	7988	dskid_3	;
$1F35	7989	dskid_4	;
$1F36	7990	mdirty	If bit is set then BAM is dirty (b7: drive 0 int/ b6: drive 1 ext)
$1F37	7991	entfnd	DIR entry found flag
$1F38	7992	lbused	last buffer used
$1F39	7993	rec	record length
$1F3A	7994	trkss	side sector track
$1F3B	7995	secss	side sector sector
			; RAM array area
$1F3C	7996	lstjob_1	($1F3C-$1F55) 26 bytes; last job
...	...		;
$1F55	8021	lstjob_26	;
$1F56	8022	dsec_1	($1F56-$1F61) 12 bytes; sector of directory entry
...	...	...	;
$1F61	8033	dsec_12	;
$1F62	8034	dind_1	($1F62-$1) 12 bytes; index of directory entry
...	...	...	;
$1F6D	8045	dinf_12	;
			; parser tables
$1F6E	8046	filtrk_1	file track number
...	...	...	; bit 7: pattern match specified
$1F77	8055	filtrk_10	;
$1F78	8056	filsec_1	file sector number
...	...	...	;
$1F81	8065	filsec_10	;
$1F82	8066	patflg	pattern presence flag ; BIT 7 Comma specified in filename ; BIT 6-0 used as a counter to show the number of ; wildcards in a given filename (eg. '*' and '?'). ; If B7 is set then B6 thru B0 are set to zero.
$1F83	8067	cbm_startrk low	;Used by the NEW command to indicate the parent
$1F84	8068	cbm_startrk high	;maxtrk, startrk, and system_track for CBM file type.
$1F85	8069	cbm_maxtrk low	;They are set by the PARTition and SETDEF routines.
$1F86	8070	cbm_maxtrk high	
$1F87	8071	cbm_system_track low	
$1F88	8072	cbm_system_track high	
$1F89	8073	dirsec	directory sector number
$1F8A	8074	delsec	sector of 1st available entry (could be splat DEL type,A scratched file, or a new entry)
$1F8B	8075	delind	index
$1F8C	8076	lstbuf	; is 0 if last block
$1F8D	8077	index	current index in buffer
$1F8E	8078	filcnt	counter, file entries
$1F8F	8079	typflg	match by type flag

Addr.	dec	Name	Comment
$1F90	8080	mode	active file mode: ; 0 = read, 1 = write, 2 = append, ; 3 = modify, 4 = free
$1F91	8081	ndbl_0	number of blocks used by a file on drive 0, 1
$1F92	8082	ndbl_1	;
$1F93	8083	ndbh_0	;
$1F94	8084	ndbh_1	;
$1F95	8085	jobs_1	($1F95-$1FAE) 26 bytes; BAM buffers (+4)
...	...	...	;
$1FAE	8119	jobs_26	;
$1FAF	8120	hdrs_1	($1FAF-$1FE2) 52 bytes; job headers
...	...	...	;
$1FE2	8162	hdrs_52	;
$1FE3	8163	cdrive	controller drive number
$1FE4	8164	iobyte	; bit7: set VERIFY on, ; bit6: CRC check on (NOT USED), ; bit5: large relative files enabled
$1FE5	8165	file_type	currently active file type
$1FE6	8166	partition	pointer to last '/' encountered
$1FE7	8167	save_err low	contains the lsb pointer to the error channel
$1FE8	8168	save_err high	; of the drive that is not active
$1FE9	8169	save_chn_data	Contains the 1st character to be sent out; over the error channel by the inactive drive
$1FEA	8170	save_lstchr_ptr	contains the pointer to the end of the error message buffer
$1FEB	8171	save_chn_rdy	contains error channel status of inactive drive
			; save rc8 when blanking screen during format
$1FEC	8172	acc	;
$1FED	8173	x	;
$1FEE	8174	y	;
$1FEF	8175	dosver	; Note: 'dosver' MUST be the last variable used; in the page, see ROUNTS2 [DiskInit]

C65 (911001) Memory Map - C64 mode

Addr.	dec	Name	Comment
			; ##### PAGE ZERO #####
$0000	0	---	($0000-$0002) 6510 register area
$0001	1	---	;
$0002	2	---	;
$0003	3	adray1 low	convert float->integer
$0004	4	adray1 high	;
$0005	5	adray2 low	convert integer->float
$0006	6	adray2 high	;
$0007	7	x	;
$0008	8	y	;
$0009	9	z	;
...			; almost the same as at the C64 ...
			; ... to be continued ...
			; C65 system interface
$E4B8	58552	c65_mode	go directly to C65 mode

B	Addr.	dec	Name	Comment
				; KERNEL
2	$C800	51200	start	;C65 system initialization
2	$C84B	51275	boot	;boot loader; 1/ Read disk block at Track 1 Sector 0 into RAM at BOOT_BUFFER.; 2/ Check for auto-boot signature, RTS if not found.
				; serial bus driver
2	$C853	51283	talk	;make a talk address; enter with .a = sa
2	$C856	51286	listen	;make a listen address; enter with .a = sa
				; DOS banking routines
				; DOS passes parameters between itself and the Kernel via unused VIC attribute bytes, because they are accessible in any memory configuration in I/O space.
2	$CCB0	52400	Get_DOS	;
2	$CCF5	52469	Leave_DOS	;
2	$CD1F	52255	Set_DOS_Status	; called with DOS mapped in
2	$CD3B	52539	Leave_DOS_1	; really leave the DOS environment, restore system I/O
2	$CD54	52564	ColdStartDOS	; assumes DOS indirects have already been set up
2	$CD64	52580	WarmStartDOS	;
2	$CD74	52596	DOS_talk	;
2	$CD80	52608	DOS_listen	;
2	$CD8C	52620	DOS_talksa	;
2	$CD9B	52635	DOS_second	;
2	$CDAA	52650	DOS_acptr	;
2	$CDB9	52665	DOS_ciout	;
2	$CDC8	52680	DOS_untalk	;
2	$CDD7	52695	DOS_unlisten	;
2	$CDE6	52710	FastLoad	; load memory from internal DOS
2	$CE63	52835	FastReadByte	; read byte immediately from chip buffer
2	$CE75	52853	FastBreak	;
2	$CE7E	52862	bootsys	; boot an alternate system. Reads the "home" sector of any diskette (physical track 0 sector 1, 512 bytes) into memory at $00400, turns off BASIC, and JMPs to it. Nothing done if disk not present. JMP not made if first byte is not $4C.
2	$CF3A	53050	ByteRead_fdc	; read byte from chip buffer
2	$CF3E	53054	BusyWait_fdc	; wait for fdc op to finish
				; C64 and monitor mode routines
2	$CF44	53060	monitor_call	;
2	$CF9E	53150	nmi_exit	;
2	$CFA4	53156	monitor_exit	;
2	$CFB1	53169	c65mode	; get default C65 memory configuration
	...	...	...	
				; C65 system interface
2	$E4B8	58552	c65_mode	go directly to C65 mode
	...	...	...	

B	Addr.	dec	Name	Comment
				;kernel64; C65 DOS Interface
2	$F72C	63276	dos_in	
2	$F77C	63356	dos_out	
2	$F7A6	63398	c65_ioinit_cold	
2	$F7AF	63407	c65_ioinit_exit	
2	$F7C1	63425	c65_ioinit_warm	
2	$F7CC	63436	talk	
2	$F7D8	63448	listn	
2	$F7E4	63460	tksa	
2	$F7F3	63475	secnd	
2	$F802	63490	acptr	
2	$F811	63505	ciout	
2	$F820	63520	untlk	
2	$F82F	63535	unlsn	
2	$F83E	63550	dos_setstatus	
2	$F8BA	63674		; jump to 'device not present error' for removed tape routines: cste1; cste2; faf; fah; rblk; wblk; trd; twrt; jtp20; tapeh; zzz
	...	...		
				; Kernel Subroutines
3	$FA86	64134	setnam	;set up a filename
3	$FA8D	64141	setlfs	;set up la, fa & sa variables
3	$FA94	64148	setbnk	;set up ba variable & filename bank
3	$FA99	64153	readss	;read I/O status for last operation
3	$FAAA	64170	setmsg	;enable/disable Kernel messages
3	$FAAE	64174	memtop	;jump entry to read/set top of user RAM
3	$FAB0	64176	gettop	;read top of memory (.c=1)
3	$FAB6	64182	settop	;set top of memory (.c=0)
3	$FABD	64189	membot	;jump entry to read/set bottom of user RAM
3	$FABF	64191	getbot	;read bottom of memory (.c=1)
3	$FAC5	64197	setbot	;set bottom of memory (.c=0)
3	$FACC	64204	iobase	;return base address of I/O block in x & y
3	$FAD1	64209	lkupsa	;look up secondary address: Enter with sa sought in y. Routine looks for match in tables. Exits with .c=1 if not found, else .c=0 & .a=la, .x=fa, .y=sa
3	$FAE6	64230	lkupla	;look up logical file address: Enter with la sought in a. Routine looks for match in tables. Exits with .c=1 if not found, else .c=0 & .a=la, .x=fa, .y=sa
3	$FAEE	64238	primm	;print immediate; a JSR to this routine is followed by an immediate ASCII string, terminated by a $00. the immediate string must not be longer than 255 characters including the terminator.
3	$FB13	64275	nnmi	;NMI/IRQ Interrupt Handlers
3	$FB4A	64330	nirq	;kernel IRQ handler (assumes I/0 at $DC00)
3	$FB77	64375	nmi	;interrupt dispatch code
3	$FB84	64388	irq_kernel	;hardware IRQ always comes here
3	$FBA5	64421	prend	;entry from other Kernel routines (MAYBE DELETE SUBROUTINE?)
3	$FBC6	64454	print_hex_byte	;convert .a to 2 hex digits and print them

B	Addr.	dec	Name	Comment
				; kernel diagnostic routines
3	$FE00	65024	phoenix	;call every logged cartridge's cold start routine; check default serial disk unit for 'boot' disk; order: external low/high, internal low/high, boot disk.
3	$FE78	65144	scan_devices	;just scan newDOS drives
3	$FEA6	65190	checksum_rom	;calculate and print ROM checksum
	...	...	...	
				; ... to be continued ...

B	Addr.	dec	Name	Comment
				; C65 BASIC 10.0 Initialization
3	$2000	8192		; basic start
3	$200C	8204	soft_reset	; warm start BASIC
3	$2025	8229	hard_reset	; cold start BASIC
3	$2053	8275	go_ready	; enable IRQ and jump to ready
3	$2057	8279	init_storage	
3	$210D	8461	init_sound_sprites	; initialize music stuff
3	$2133	8499	signon_message	; shows start logo and text
3	$22A8	8872	init_vectors	
3	$22CC	8908	chrget	; get next character from text
3	$22CE	8910	chrgot	; re-get current character from text
				; C65 BASIC Indirect Load Subroutines
3	$22E2	8930	inddef	; #defpnt - lda_far_ram1
3	$22E6	8934	indfrm	; #form - lda_far_ram1
3	$22EA	8938	inddpt	; #dscpnt - lda_far_ram1
3	$22EE	8942	indhtr_ram1	; #hightr - lda_far_ram1
3	$22F2	8946	indfmo	; #facmo - lda_far_ram1
3	$22F6	8950	indlow	; #lowtr - lda_far_ram0
3	$22FA	8954	indst1	; #strng1 - lda_far_ram0
3	$22FE	8958	indst1_ram1	; #strng1 - lda_far_ram1
3	$2302	8962	indgrb	; #grbpnt - lda_far_ram1
3	$2306	8966	indlow_ram1	; #lowtr - lda_far_ram1
3	$230A	8970	indin1	; #index1 - lda_far_ram1
3	$230E	8974	indtxt	; #txtptr - lda_far_ram0
3	$2310	8976	lda_far_ram0	; LDA (.x),Y from bank 0
3	$231E	8990	indin1_ram1	; #index1 - lda_far_ram1
3	$2320	8992	lda_far_ram1	; LDA (.x),Y from bank 1
3	$2339	9017	sta_far_ram1	; STA (.x),Y to bank 1
3	$234F	9039	sta_far_in1	; #index1 - sta_far_ram0
3	$2353	9043	sta_far_txt	; #txtptr - sta_far_ram0
3	$2355	9045	sta_far_ram0	; STA (.x),Y to bank 0
3	$2360	9056	indcmp_in1	; #index1 - STA (.x),Y to bank 0

B	Addr.	dec	Name	Comment
3	$2368	9064	crunch	; crunch - tokenization routine ; ; entry: TXTPTR points to start of text to crunch ; exit: TXTPTR points to start of crunched text ; ; calls: CHRGET ; CHRGOT ; RESER ; KLOOP ; REM ; DATA ; ; CRUNCH collapses all reserved words into tokens. It removes all graphic characters (characters with msb set) not in quoted strings, DATA or REM statements. ; ; An escape token is implemented as follows: ; ; As each character on a line of text to be crunched is scanned, an indirect jump is performed. Anyone wishing to scan for their own commands should grab off this vector, saving the return vector. ; On entry, if the carry flag is set, it is still up for grabs. ; The current text pointer is at TXTPTR. If the escape routine recognizes the command, it should: ; ;) put the length of the reserved word in .y ;) put the desired ‘second’ token in .a ;) clear the carry flag ;) put type of token in x: 0==>command, ff==>function ; ; If it is not your command, leave .a and the carry flag intact. ; NOTE: The reserved word must be >= 2 characters long. Exit through the old vector (for daisy chaining). If the carry flag is clear on entry it means someone else before you recognized this command. In this case, just pass control through the old vector.
3	$2427	9255	kloop	; crunch loop; moves offset .y characters from txtptr to end of line ; .x is preserved
3	$243D	9277	reser	; search reserved word list for a match ; ; entry: (txtptr) is first char of word to match ; (y,a) is start of table to check ; ; exit: .y length of word matched ; .c success/fail (set/clear) flag ; count token value
3	$2483	9347	keyword_list 1	($2483-$27DF) 861 bytes ; non-escape keyword list
	...	...		;
	$27DF	10207	keyword_list 861	;
3	$27E0	10208	esc_function_list 1	($27E0-$2821) 66 bytes ; escape function tokens
	...	...		;
	$2821	10273	esc_function_list 66	;
3	$2822	10274	stmdsp 1	($2822-$29AF) 398 bytes ; jump table for dispatch routine
	...	...		;
3	$29AF	10671	stmdsp 398	;

B	Addr.	dec	Name	Comment
3	$29B0	10672	ok_error_message	; error message: ok = O(K+$80) for end
3	$29B2	10673	error_message_list 1	($29b2-$2BEE) 573 bytes ; error messages
	...	...		;
3	$2BEE	11246	error_message_list 573	;
				; error message output ; ; routine to translate error message # in .a into address of string containing message in index2
3	$2BEF	11247	erstup	; error set up ; start with address of first error message
				; execute dispatcher ; ; here for new statement. ; character -> by txtptr is ':' or eol. ; the adr of this loc is left on the stack when a statement is executed so that it can merely do a rts when it is done. ; get char, exit via xeqcm3, and return to newstt.
3	$2C08	11272	xeqcm	; check if there is an interrupt from VIC that needs to be serviced
3	$2C0B	11275	ngone	; get off here if we are in direct mode
	$2C50	11344	xeqdir	;
3	$2C53	11347	newstt	; in run mode: save txtptr for CONTinue command
3	$2C8D	11405	tto	; copy txtptr to oldtxt
3	$2C97	11415	xeqrts	;
				; set up for command processing and set processor address on stack. ; exit via jmp to CHRGET
3	$2C98	11416	xeqcm3	; print '[line-number]'
3	$2CAF	11439	xeqcm2	; special cases: ESC, go to, mid$()=
	$2CCA	11466	xeqcm4	; convert adjusted token into an index into a jump table.
3	$2CD0	11472	xeqcm5	; dispatch table 1 or 2?
3	$2CE8	11496	xeqmid	; handle special case of MID$=
3	$2CEE	11502	xeqchr	; jmp chrget
3	$2CF1	11505	xeqesc	; execute escape token
3	$2D06	11526	nescex	; jmp chrget
3	$2D08	11528	snerr1	
3	$2D0B	11531	morsts	; if ':', continue statement
				; STOP, STOP KEY, and END handlers
3	$2D12	11538	is_stop_key_down	; test stop key
3	$2D17	11543	break_exit	; STOP KEY ; wait for the user to release the key
3	$2D39	11577	bcc_ready	; say 'ready' (if END),
3	$2D3C	11580	break	; say 'break' (if STOP) ; exit via 'in line #'

B	Addr.	dec	Name	Comment
				; function handler ; ; At this point, eval has determined that the token in a has to be a function. It must therefor be in the range SGN...MID$ (old BASIC), or RGR...INSTR (new extensions). We will collapse these two disjoint blocks into one continuous range. ; ; On entry, we can assume the token is >= 'sgn'
3	$2D53	11603	isfun	;is this an escape function?
				; Most functions take a single argument. The return address of these functions is CHKNUM, which ascertains that VALTYP=0 (numeric). Normal functions which return string results (eg. CHR$) must pop off that return address and return directly to FRMEVL. ; ; The so called "funny" functions can take more than one argument, the first of which must be string and the second of which must be a number between 0 and 255. The closed parenthesis must be checked and return is directly to FRMEVL with the text pointer pointing beyond the ")". The pointer to the description of the string argument is stored on the stack underneath the value of the integer argument.
3	$2D97	11671	oknorm	; check for open parens, evaluate argument and restore token
3	$2D9B	11675	fingo	; convert token to index into jump table
3	$2DB0	11696	do_esc_fn	; escape function handler
3	$2DD1	11729	foreign_esc_fn	; flag 'up for grabs'
3	$2DD5	11733	n_esc_fn_vec	; it's unwanted. off to the refuse pile
3	$2DDB	11739	go_foreign_esc_fn	;
3	$2DDE	11742	orop	; must always complement
3	$2DE1	11745	andop	; op (facmo&lo)=int value and check size ; use Demorgan's Law on high and low ; float (a,y) and return to user
3	$2E0E	11790	dorel	; perform a relational operator ; (domask) contains the bits as to which relational operator it was. ; carry bit on = string compare.
3	$2E25	11813	strcmp	;
3	$2E4D	11853	stasgn	;
3	$2E52	11858	nxtcmp	;
3	$2E58	11864	qcomp	;
3	$2E5D	11869	getcmp	;
3	$2E75	11893	docmp	;
				; ... to be continued ...
	...	...		
3	$3B31	15153	SYS	
	...	...		
3	$3C8E	15502	AUTO	
	...	...		
				; ... to be continued ...

This work mainly based on my playing around with the C65 ROM (911001).

Whenever possible, I have used the labels, names and comments that were used in the original unfinished C65 sourcecodes and appropriate expressions in public ROM listings for the C64 and C128 ROMs.

This memory map comes without any warranty of correctness.

This document is still under development and will be gradually expanded and corrected.

You can always get the latest version of this document at: 65site.de

Günther Reiter
65software@gmx.de
65site.de

Date: 2020/11/09