TO RESET BREAK KEY AFTER WARM X BASIC SOFT
POKE 48630, 1 : POKE 48631, 57

System Monitor 1.0 RAM Allocation

The Aamber Pegasus System Monitor reserves 64 bytes of RAM starting at address $BDC0, of which some may be of interest to the user. A detailed summary of system defined locations is given below:

$BDC0 INBUFF

Single character input buffer for input coming from the keyboard. This location has a character in it when the RXREADY byte contains a non-zero number. (Bytes which are defined as flags or Booleans are deemed to be FALSE when they contain zero, and TRUE when non-zero.) As an example of its use, an input routine may be written thus:

```
INPUT   LDA     RXREADY     . Check the flag
        BEQ     INPUT       . Loop if FALSE
        LDA     INBUFF      . Else get character
        CLR     RXREADY     . Clear flag for next time
        RTS                 . End of subroutine
```

$BDC1 TICKS

This byte is incremented every 20 milliseconds, and varies from 0 to 49 to provide a one-second clock.

$BDC2 TOCKS

This is similar to TICKS, but range from 0 to 255 in 20ms intervals.

$BDC3 SECONDS

Incremented every second, range from 0 to 59. This is part of the time that may be set using the monitor's T command.

$BDC4 MINUTES

Incremented every minute, range 0..59.

$BDC5 HOURS

Incremented every hour, range 0..11.

Whenever a SWI or NMI occurs, the machine state, which consists of the 6809 registers, will be saved in RAM. This feature is designed for use with the DEBUG Eprom, and is useful when testing and debugging machine language programs. These pseudo-registers are defined below:

$BDC6	CC-REG	Condition code
$BDC7	A-REG	A accumulator
$BDC8	B-REG	B accumulator
$BDC9	DP-REG	Direct Page
$BDCA	X-REG	X index
$BDCC	Y-REG	Y index
$BDCE	PC-REG	Program Counter
$BDD0	S-REG	System Stack Pointer
$BDD2	U-REG	User Stack Pointer

$BDD4 INVERT

This byte defines the invert state of characters as they are output to the video screen. When this byte is zero, characters will appear dark on a light background; when this byte contains $80, they will appear light on a dark background. Note that the action of the invert byte is modified by MSBINV, which is controlled by the blank key on the lower right side of the keyboard. INVERT is cleared by the control-A code, and is set by control-B. Note that its normal state is with INVERT set.

$BDD5 CLOCK

This Boolean defines whether the clock is displayed or not. The clock will appear at the upper right of the display. The clock is turned on with control-P, and off with control-Q.

$BDD6 CURSOFF

When TRUE, the cursor is turned off. It is preferred that the cursor be turned off using control-O, and on with the control-F code.

$BDD7 Reserved

$BDD8 PAUSING
Suspends output when non-zero: this function is actuated with the ESC (escape) key, and is used for things like stopping listings.

$BDD9 RAWMODE
Prints a special character for control codes instead of executing the appropriate control function. Control-Y enables, control-Z disables this flag.

$BDDA RXREADY
Flag that indicates when a character is ready for input from the keyboard. See INBUFF description for example of its use.

$BDDB to $BDDD Reserved

$BDDE KEYLOCK
Lock out keyboard - when this flag is TRUE, then the full ASCII keyboard will be disabled.

$BDDF CAPSLOCK
Inverts the sense of the shift key for the letters A..Z on the keyboard. This flag is controlled by the CAPS LOCK key on the keyboard.

$BDE0 Reserved

$BDE1 MSBINV
Inverts video for characters printed: this flag has priority over the INVERT flag, and is controlled by the blank key on the lower right of the keyboard.

$BDE2 START
Two bytes are reserved for storing the start address of files saved on cassette. When files are loaded from cassette, the starting address is placed here.

$BDE4 FINISH
Two bytes for cassette finish address.

$BDE6 RELOC

Flag that specifies, when non-zero, that a file loading from cassette is to be relocated to the address stored in START.

$BDE7 to $BDE9 Reserved

$BDEA IRQ_VEC Interrupts are vectored to address stored here.
$BDEC SWI1_VEC SWI1 vectored here.
$BDEE SWI2_VEC SWI2 vectored here.
$BDF0 SWI3_VEC SWI3 vectored here.

$BDF2 MEMBEG

Address of start of user memory stored here. This address is determined by a non-destructive memory test after a power-on RESET.

$BDF4 MEMEND

Address of end of available user memory is stored here. This address is also where the stack starts.

$BDF6 ABORT

When the BREAK key on the keyboard is depressed, the routine specified by the adress stored here is called as a subroutine.

$BDF8 LINES

This byte is very crucial - it controls the number of lines on the screen, and must always be in the range of 0..16. Due to the software scanned video, a reduction in the number of lines displayed will significantly reduce the processing time.

$BDF9 XPOSIT

Horizontal cursor position. Always in the range of 0 to 31. It is preferred that any changes in this location be effected by executing the appropriate control codes, and not by writing directly into it.

$BDFA YPOSIT

Vertical cursor position, range always 0 to 15.

User Accessible Pegasus System Monitor Subroutines

The Aamber Pegasus System Monitor EPROM contains a number of machine language subroutines that may be found to be of use. Each routine is called via a table of addresses, using the 6809 indirect addressing mode. E.g. ECHO is called with AD 9F F8 00. Unless stated otherwise, assume that no registers are preserved.

ECHO $F800
ASCII character in A reg. is output to video display. All registers are preserved.

INPUT $F802
Waits for key to be typed, returns ASCII value 0..7F in A reg. Character is not echoed.

INKEY $F804
Scans keyboard for keystroke. If key found, it will be returned in A reg. with carry clear - if not found, then carry will be set and A will be undefined. Character not echoed.

PROMPT $F806
Re-entry point into machine language monitor.

CASSOUT $F808
Outputs data to cassette recorder with start and end addresses in locations START and FINISH. (See RAM allocation.) Record button on recorder must be going.

CASSIN $F80A
Reads the next file from cassette tape, assuming that the play button has been pushed.

TRIMCASE $F80C
Character in A reg. is converted from lower case to upper case. Also, '.' becomes ' ' and ',' becomes ' '.

EXOUT $F80E
Takes number in A reg., displays as two hex digits.

HEXDIG $F810
Takes digit 0..F in A reg, converts to ASCII '0'..'F'.

GETBYTE $F812
Reads two hex digits from keyboard, echoing if valid, returns 8 bit number in A reg. Carry set if not valid.

GETLINE $F814
Inputs line into buffer pointed to by X reg. on entry. Buffer size given by A reg. Line terminated with RETURN key (echoed as CR,LF pair). Characters are echoed as they are typed, and may be corrected with the BACK SPACE key. The control-U function will clear the buffer. On return, the A and X regs. are preserved, and B contains a byte count of characters typed. The RETURN will not be in buffer.

TEXTOUT $F816
Output string pointed to by X reg, terminated with nul ($00). The X reg. is left pointing to after the message.

GRAFIX $F818
See discussion of Pegasus graphics capabilities.

GETDIG $F81A
Reads and echoes a decimal digit from keyboard, returning in A reg, carry set if non-numeric digit was typed.

GETNUM $F81C
Reads and echoes decimal number range 00 to 99 from keyboard, returns in A reg. Non-numeric means carry set.

CONVERT $F81E
Takes decimal number, range 00 to 99 in B reg, converts to two ASCII characters in the D reg. (A and B regs.)

GETWORD $F820
Gets four hex digits from keyboard, echoing them. Returns number in X reg. Carry set if non-hex typed.

RETMENU $F822
Entry point back to Menu selection mode in Monitor.

HEXTEST $F824
Tests ASCII char. in A reg. for range '0' to 'F'. If not in range, returns with carry set, else digit returns in A.

NEWLINE $F826
Causes cursor to move to start of next line.

FORMFD $F828
Equivalent to control-L: clears the video screen.

DELINE $F82A
Clears line that cursor is on - same as control-U.

SPACER $F82C
Outputs one space to screen at current cursor position.

HOMEUP $F82E
Homes cursor to top left corner - same as control-T.

NUL $F830
Nul function - has no effect.

LINEOUT $F832
Similar to TEXTOUT, except that a NEWLINE is done first.

Using the Aamber Pegasus System Monitor

When your Pegasus is switched on and running, you will see a reassuring sign-on message and menu on your T.V. set.

```
AAMBER Pegasus 6809
Technosys Research Labs

Assembler 1.1
Disassembler 1.1
Tiny 1.1
Monitor 1.1

Select one of above:
```

The menu items appearing upon the screen will depend upon which EPROM based software you have purchased with your Pegasus. Any of the menu items may be selected by typing the first letter of the appropriate entry. For instance, Tiny BASIC may be selected by tapping the 'T' key for 'Tiny'.

The system monitor may be entered with 'M' for 'Monitor'. The prompt used for monitor commands is the greater-than symbol, '>'. A summary of monitor commands is given below:

G hhhh

GO function - machine language programs may be executed simply by typing the desired starting address. If you wish execution to occur, type the '.' key; any other key typed will abort back to the monitor.

L hhhh

LOAD function - the load operation will read data from the cassette interface into RAM. The data loaded will be a contiguous region of memory, and the load start and end addresses will be displayed, along with the filename that was used when saving the file. A new load start address may be specified to relocate the file on loading.

M hhhh

MODIFY function - a memory address is specified, and the byte found there will be displayed. You may proceed to the next byte with the '.' key, and move to the previous byte with the ',' key. A new address may be specified by typing the 'M' key. At any stage, the byte at a RAM location may be changed, simply by entering a new, two digit hex value. If the location was not changed, then the contents of the location are redisplayed.

T hh mm ss

The time may be set using the TIME function. The time is measured in hours, minutes, and seconds.

S hhhh hhhh

SAVE - a block of data between the given start and end addresses will be saved on cassette tape. A filename may be specified. The screen will blank for the duration of the SAVE operation.

Special Keyboard Functions

ESCAPE

The ESCAPE key is used to suspend output to the video screen - tapping once will stop output, tapping the key again will restart it.

BREAK

The BREAK key is used to stop program execution. The effect of this key depends upon which program is running.

CAPS LOCK

The CAPS LOCK key inverts the sense of the shift key, so that upper or lower case letters may be typed in.

PANIC

The PANIC button is inside the Pegasus box, on the board itself. This key should be used rather than turning the power off - it will always jump to the menu.

Control Function Summary

The following table is a list of the control codes used by the Pegasus, these codes can be implemented by holding the CTRL key down and pressing the associated letter.E.G.: to obtain a horizontal tab hold the CTRL key down and press the I button down. Note that a horizontal tab could more easily have been obtained using the TAB key. Many of the control codes have separate keys which are equivalent to using the CTRL key.

Another method of outputting a control code is to output the associated value, e.g.in BASIC to output a form-feed one would type PRINT CHR($0C), or in FORTH: 12 EMIT , or in assembly language : LDA #0C
JSR ECHO

Hexadecimal value	Decimal value	Control character	Separate key	Meaning
00	0	@	00	Null
01	1	A	01	Set inverse video mode
02	2	B	02	Clear inverse video mode
03	3	C	03	no function
04	4	D	04	Move cursor right
05	5	E	05	Move cursor up
06	6	F	06	Turn cursor on
07	7	G	07	no function
08	8	H	08 BACK SPACE	Back space
09	9	I	09 TAB	Horizontal Tab
0A	10	J	0A LINE FEED	Line Feed
0B	11	K	0B	Set cursor position (Followed by X, and Y)
0C	12	L	0C	Form feed (clear Screen)
0D	13	M	0D RETURN	Return cursor to left margin
0E	14	N	0E	Scroll screen up
0F	15	O	0F	Turn cursor off

(Continued overleaf).

Shorthand

! = CHR (
= HEX (
% = >=
' = <=
; = <>
\ = INKEY

b = USR
c = FREE
- = RND
. = MOD
^ = ABS
@ = @(
w = LET
e = PEEK(
d = NOT
z = RAN(

MEMORY

FFFF – MONITOR
F000 – MONITOR
EFFF I/O
E000 I/O
DFFF SKT 2
D000 "
CFFF SKT 1
C000 "
BFFF VDU RAM (48640 – 49151)
BE00 "
BDFF MONITOR SCRATCH PAD RAM
BD00 ↓
BDBF STACK
B000 ↑ AFFF USR RAM

1FFF 2 EPROM
1000
0FFF 1 EPROM
0000

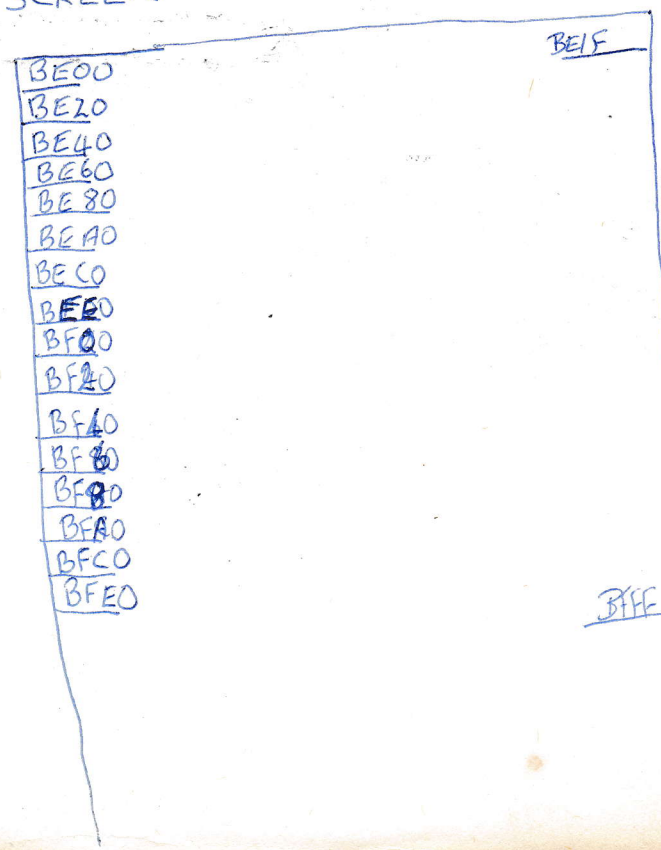
BE00 [ ] BFFF = POKE$Bxxx, character

Control function summary contd.

Hexadecimal value	Decimal value	Control character	Separate key	Meaning
10	16	P	10	Turn on digital clock
11	17	Q	11	Turn off digital clock
12	18	R	12	no function
13	19	S	13	Move cursor left
14	20	T	14	Home cursor to top left
15	21	U	15	Clear line that cursor is on
16	22	V	16	Scroll screen down
17	23	W	17	no function
18	24	X	18	Move cursor down
19	25	Y	19	no function
1A	26	Z	1A	no function
1B	27	[	1B ESCAPE	Suspend output **
1C	28	\	1C	no function
1D	29	]	1D	no function
1E	30	^	1E	no function
1F	31	_	1F	no function

** This code can only be implemented from the keyboard

SCREEN



GRAFIX SAVE RUN 2000

LOAD " 3000

Gfx XBAS

```
?Chr$(27);Chr$(N);

For x= 0 to 15
  Print Chr$(Y);
next x
```

ASCII CHARACTER TABLE

The members from 0 to 31 are reserved for special screen function controls.

CHARACTER	ASCII (decimal)		HEX	CHARACTER	ASCII (decimal)		HEX	CHARACTER	ASCII (decimal)	HEX
Space	160	32	20	@	192	64	40	`	96	60
!	161	33	21	A	193	65	41	a	97	61
"	162	34	22	B	194	66	42	b	98	62
#	163	35	23	C	195	67	43	c	99	63
$	164	36	24	D	196	68	44	d	100	64
%	165	37	25	E	197	69	45	e	101	65
&	166	38	26	F	198	70	46	f	102	66
'	167	39	27	G	199	71	47	g	103	67
(	168	40	28	H	200	72	48	h	104	68
)	169	41	29	I	201	73	49	i	105	69
*	170	42	2A	J	202	74	4A	j	106	6A
+	171	43	2B	K	203	75	4B	k	107	6B
,	172	44	2C	L	204	76	4C	l	108	6C
-	173	45	2D	M	205	77	4D	m	109	6D
.	174	46	2E	N	206	78	4E	n	110	6E
/	175	47	2F	O	207	79	4F	o	111	6F
0	176	48	30	P	208	80	50	p	112	70
1	177	49	31	Q	209	81	51	q	113	71
2	178	50	32	R	210	82	52	r	114	72
3	179	51	33	S	211	83	53	s	115	73
4	180	52	34	T	212	84	54	t	116	74
5	181	53	35	U	213	85	55	u	117	75
6	182	54	36	V	214	86	56	v	118	76
7	183	55	37	W	215	87	57	w	119	77
8	184	56	38	X	216	88	58	x	120	78
9	185	57	39	Y	217	89	59	y	121	79
:	186	58	3A	Z	218	90	5A	z	122	7A
;	187	59	3B	[	219	91	5B	{	123	7B
<	188	60	3C	\	220	92	5C	|	124	7C
=	189	61	3D	]	221	93	5D	}	125	7D
>	190	62	3E	^	222	94	5E	~	126	7E
?	191	63	3F	_	223	95	5F	■	127	7F

Note the difference between the digit 0 and the letter O