Apple //c Technical Note #1

Revision of Apple //c Technical Note #1, 8 February 1984*
25 February 1984

There are differences between how the mouse works on the Apple //e and how it works on the Apple //c. This technical note explains what is causing these differences and how to write programs that work the same on both machines.

For further information contact:
PCS Developer Technical Support
M/S 22-W, Phone (408) 996-1010

Disclaimer of all Warranties and Liabilities

Notice

Apple Computer, Inc. reserves the right to make improvements in the product described in this document at any time and without notice.

* A clarification of the effects disabling interrupts has on mouse data has been added

BG

INTRODUCTION

As advertised, if you use the mouse firmware routines such as SETMOUSE to control the mouse then these routines will perform the same function in the Apple //c as they do in the Apple //e. This does not mean that a program which uses the mouse will behave the same in both computers. There are two reasons for this. One is that if a program has not properly set the environment prior to calling these routines it is possible for the program to work in one machine and not in the other. The second reason is that there are differences in the machines and although the ROM rouines perform the same functions there may be noticable differences in the 'behaviour' of the mouse. This technical note will explain the fundamental differences between the way the mice in the two machines work. It will then point out precautions that need to be taken to make sure that your machine language program will work on both machines. With the exception of mouse movement scaling described below BASIC and Pascal programs do not need to be concerned about setting the proper environment.

The Apple //e mouse card has a microprocessor on it which constantly polls the mouse to get status and position information. This data is then kept on the card and is available whenever the program requests it through the READMOUSE routine. If the mouse is in passive mode this information will be 'picked up' by the main program whenever it gets around to it. The SETMOUSE routine can set the mouse card to issue interrupts under certain conditions. When the mouse card determines that such conditions exist it issues an interrupt. This stops the main computer and goes to what ever interrupt handling routine has been set up. This routines will then read the information from where the card processor saved it and puts it in the screen holes. When using a mouse on an Apple with a mouse card your program is only interrupted if your program has requested it. And the data in the screen holes is being changed only when the program's interrupt handler or polling routine has called READMOUSE. Also enabling and inhibiting interrupts does not affect the updating of mouse information by the card's microprocessor.

The Apple //c mouse does not have a card microprocessor and so mouse information is collected by interrupting the Apple //c's microprocessor. When the interrupt happens the firmware captures it and processes it which includes updating the screen holes. The interrupt is passed on only if SETMOUSF set up the conditions to do so. However, having the mouse interrupt the computer's microprocessor means that your program is being constantly interrupted. This will affect program timming. It also means that the screen holes are constantly being updated with X and Y information even in passive mode since this information must be kept somewhere and there is no card to keep it on. Also, if you have disabled interrupts then the mouse can never interrupt the processor and so the X adn Y values are never updated and calling READMOUSE will indicate that there has been no mouse movement.

Since the Apple //c is constantly being interrupted while the mouse is on, the program's performance may be affected. To minimize this affect the Apple //c responds one-half as frequently to mouse movements as does the mouse card. The noticable result of this is that the mouse must be moved twice as far to create the same effect. If you want the same behaviour on both machines then multiply the Apple //c X and Y values by two and clamping to 1/2 the //e value before using them.

With the exception of having to double the Apple //c mouse movement your program can ignore which machine it is running on by following the precautions listed below. If you are working from BASIC or Pascal these conditions are taken care of for you.

THE FOLLOWING CONDITIONS MUST BE TAKEN INTO ACCOUNT BY MACHINE LANGUAGE PROGRAMMERS IF THE PROGRAM IS TOP RUN SIMILARLY IN ALL THF APPLE // FAMILY OF COMPUTERS:

* Do not disable interrupts unless you must. Then be sure to re-enable them.

* Disable interrupts when calling any mouse routine (SEI).

* Do not re-enable interrupts (CLI) or (PLP if previously had done a PHP) after READMOUSE until X & Y data have been removed from the screen holes.

* Be sure to disable interrupts (SEI) before placing position information in the screen holes (POSMOUSE or CLAMPMOUSE).

* Enter all mouse routines (not required for SERVEMOUSF) with the X register set to $Cn and Y register set to $n0 where n = slot number.

* Some programs may need to turn off interrupts for purposes other then reading the mouse. This is sometimes done on the Apple //e to keep from having to handle interrupts while in auxiliary memory. If interrupts are turned off and then back on, the first call to READMOUSE may give incorrect values. Subsequent calls to READMOUSE will return correct values until interrupts are turned off and on again. Turning off interrupts for mouse calls does not create this problem. If you are watching numbers coming form the mouse while moving it in a direction that would increase values you might see the following: 6, 7, 8, 9, 8, 9, 10. In practice this momentary 'glitch' in the stream of mouse data has little importance and would probably only be noticed by a programmer testing his/her program - no one's hand is that steady. If you must keep this 'glitch' from happening then do not keep interrupts off for more then 40 microseconds or be sure that at least one mouse interrupts has taken place since interupts were turned back on.

Apple //c Technical Note #2

Using 40 Column text with Double High Resolution Graphics
22 March 1984

This technical note describes how to properly handle the 40 column screen while using double high-resolution graphics on the Apple //c.

Disclaimer of all Warranties and Liabilities

Apple Computer, Inc. makes no warranties either express or implied, with respect to this documentation or with respect to the software described in this documentation, its quality, performance, merchantability, or fitness for any particular purpose. Apple Computer, Inc. software is licensed "as is". The entire risk as to its quality and performance is with the vendor. Should the programs prove defective following their purchase, the vendor (and not Apple Computer, Inc., its distributor, or retailer) assumes the entire cost of all necessary damages. In no event will Apple Computer, Inc. be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Apple Computer, Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental of consequential damages, so the above limitation may not apply to you.

Notice

Apple Computer, Inc. reserves the right to make improvements in the product described in this document at any time and without notice.

CJS

Many developers using double high resolution (dbl-hi-res) graphics may wish to use 40 column text displays so that the text can be read on a television set. There are a couple of possibilities:

1.) You can define your own dbl-hi-res character set with any size characters you desire and then plot them on the dbl-hi-res screen.

2.) You can print text to the Apple //c text screen and toggle the screen on to display it.

To use the second method, however, does require some special considerations.

The firmware in the Apple //c implements the scroll routine differently that the Apple //e 80 column firmware. The Apple //c scroll routine continues to use the window parameters when scrolling, but uses the 80COL softswitch to determine if it should scroll the 80 or 40 column screen. Since the firmware has initialized a 40-column window, the scroll routines will move only the first 40 columns. But, the 80COL flag has been turned on for dbl-hi-res! Therefore, the scrolling routine takes every even column from auxiliary memory and every odd column from main memory. As a result, only the first 40 columns get scrolled, 20 columns from auxiliary memory and 20 columns from main memory.

One possible solution to the problem is to write your own scroll routines. Another might be to write to the screen so that scrolling will not occur. But these is yet another solution. Turn on the full 80 column mode with a "PR#3" or the equivalent. Now print your text to COUT in the normal manner being careful not to exceed 40 characters per line. The 80 column firmware will scroll everything properly.When you are ready to display text, send a CONTROL-Q through COUT to switch to 40 columns. When you are ready to return to dbl-hi-res mode, send a CONTROL-R to COUT.

When making this switch, a momentary "glitch" may occur. If you send the CONTROL-Q to COUT while still in graphics mode the screen will go to regular "single" hi-res mode before finally going to text mode. If you switch to text mode first, the text will be in 80 column mode (with 40 columns displayed on the left half of the screen) before ultimately going to 40 column mode. The same potential glitch may occur goint back to dbl-hi-res. The "glitch" will be only momentary and may not present any problem for you. If it does, you may wish to make your change-over coincide with the video's verticle blanking interval. (See the Apple //c Reference Manual.)

NOTE: There is no way to display 4 lines of 40 column text at the bottom of the dbl-hi-res screen in mixed mode since the 80 column hardware must be active while dbl-hi-res is being displayed.

Apple //c Technical Note #3

Foreign Language Keyboard Layouts
1 March 1984

There are differences between the keyboard layout on the North American Apple //c and Apple //c's in other coutries. This technical note documents the layouts, along with the ASCII codes for each key, for the French, Italian, German, and United Kingdom systems.

For further information contact:
PCS Developer Technical Support
M/S 22-W Phone (408) 996-1010

Disclaimer of all Warranties and Liabilities

Notice

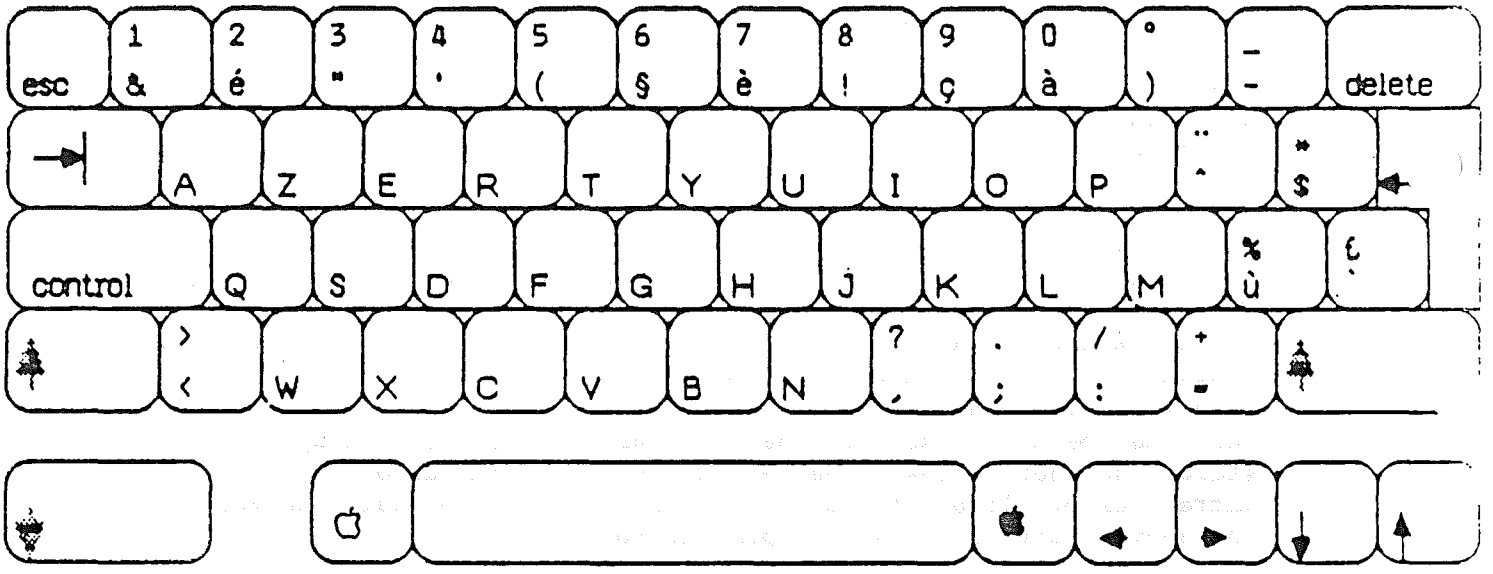
Apple Computer, Inc. reserves the right to make improvements in the product described in this document at any time and without notice.

PB

Apple //c

Standard France Keyboard Layout

October 24, 1983



Notes: 1) Uses "Shift lock" instead of "Caps lock" -- All keys are shifted.

2) When "Shift Lock" is depressed, "Shift" keys unshift all keys.

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
1	01	000	ESC	1B	ESC	1B	ESC	1B	ESC	1B
1	02	004	&	26	1	31	&	26	1	31
1	03	008	é	7B	2	32	é	7B	2	32
1	04	00C	"	22	3	33	"	22	3	33
1	05	010	'	27	4	34	'	27	4	34
1	06	018	(	28	5	35	(	28	5	35
1	07	014	GS	1D	GS	1D	§	5D	6	36
1	08	01C	è	7D	7	37	è	7D	7	37
1	09	020	!	21	8	38	!	21	8	38
1	10	024	FS	1C	FS	1C	ç	5C	9	39
1	11	0C0	NUL	00	NUL	00	à	40	0	30
1	12	0C4	ESC	1B	ESC	1B	)	29	°	5B
1	13	0BC	US	1F	US	1F	-	2D	_	5F
1	14	130	DEL	7F	DEL	7F	DEL	7F	DEL	7F
2	16	028	HT	09	HT	09	HT	09	HT	09
2	17	02C	SOH	01	SOH	01	a	61	A	41
2	18	030	SUB	1A	SUB	1A	z	7A	Z	5A
2	19	034	ENQ	05	ENQ	05	e	65	E	45
2	20	038	DC2	12	DC2	12	r	72	R	52
2	21	040	DC4	14	DC4	14	t	74	T	54
2	22	03C	EM	19	EM	19	y	79	Y	59
2	23	044	NAK	15	NAK	15	u	75	U	55
2	24	048	HT	09	HT	09	i	69	I	49
2	25	04C	SI	0F	SI	0F	o	6F	O	4F
2	26	0E4	DLE	10	DLE	10	p	70	P	50
2	27	0E8	RS	1E	RS	1E	^	5E	¨	7E
2	28	0EC	$	24	*	2A	$	24	*	2A
3	31	050	DC1	11	DC1	11	q	71	Q	51
3	32	058	DC3	13	DC3	13	s	73	S	53
3	33	054	EOT	04	EOT	04	d	64	D	44
3	34	060	ACK	06	ACK	06	f	66	F	46
3	35	064	BEL	07	BEL	07	g	67	G	47
3	36	05C	BS	08	BS	08	h	68	H	48
3	37	068	LF	0A	LF	0A	j	6A	J	4A
3	38	06C	VT	0B	VT	0B	k	6B	K	4B
3	39	074	FF	0C	FF	0C	l	6C	L	4C
3	40	070	CR	0D	CR	0D	m	6D	M	4D
3	41	114	ù	7C	%	25	ù	7C	%	25
3	41A	0B8	`	60	£	23	`	60	£	23
3	42	108	CR	0D	CR	0D	CR	0D	CR	0D
4	43A	0E0	DEL	7E	DEL	7E	<	3C	>	3E
4	44	078	ETB	17	ETB	17	w	77	W	57
4	45	07C	CAN	18	CAN	18	x	78	X	58
4	46	080	ETX	03	ETX	03	c	63	C	43
4	47	084	SYN	16	SYN	16	v	76	V	56
4	48	088	STX	02	STX	02	b	62	B	42
4	49	08C	SO	0E	SO	0E	n	6E	N	4E
4	50	090	,	2C	?	3F	,	2C	?	3F
4	51	094	;	3B	.	2E	;	3B	.	2E
4	52	098	:	3A	/	2F	:	3A	/	2F
4	53	09C	=	3D	+	2B	=	3D	+	2B
5	58	110	SP	20	SP	20	SP	20	SP	20
5	60	138	BS	08	BS	08	BS	08	BS	08
5	61	13C	NAK	15	NAK	15	NAK	15	NAK	15

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
5	62	134	LF	0A	LF	0A	LF	0A	LF	0A
5	63	10C	VT	0B	VT	0B	VT	0B	VT	0B

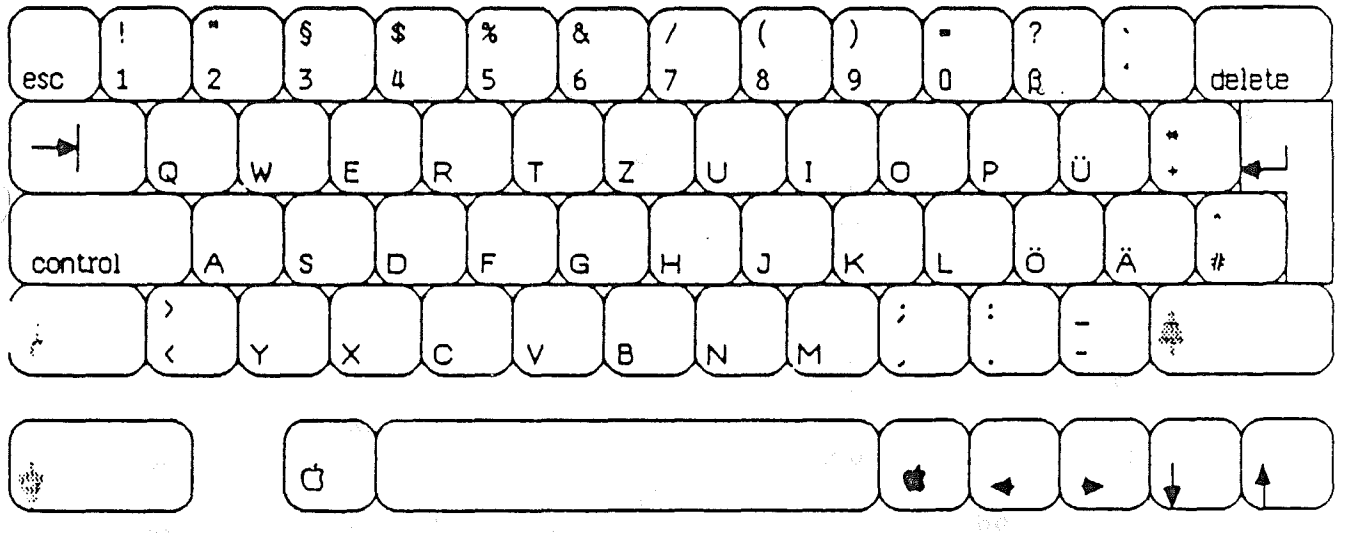
Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
1	01	200	ESC	1B	ESC	1B	ESC	1B	ESC	1B
1	02	204	1	31	&	26	1	31	&	26
1	03	208	2	32	é	7B	2	32	é	7B
1	04	20C	3	33	"	22	3	33	"	22
1	05	210	4	34	'	27	4	34	'	27
1	06	218	5	35	(	28	5	35	(	28
1	07	214	GS	1D	GS	1D	6	36	§	5D
1	08	21C	7	37	è	7D	7	37	è	7D
1	09	220	8	38	!	21	8	38	!	21
1	10	224	FS	1C	FS	1C	9	39	ç	5C
1	11	2C0	NUL	00	NUL	00	0	30	à	40
1	12	2C4	ESC	1B	ESC	1B	°	5B	)	29
1	13	2BC	US	1F	US	1F		5F	-	2D
1	14	330	DEL	7F	DEL	7F	DEL	7F	DEL	7F
2	16	228	HT	09	HT	09	HT	09	HT	09
2	17	22C	SOH	01	SOH	01	A	41	a	61
2	18	230	SUB	1A	SUB	1A	Z	5A	z	7A
2	19	234	ENQ	05	ENQ	05	E	45	e	65
2	20	238	DC2	12	DC2	12	R	52	r	72
2	21	240	DC4	14	DC4	14	T	54	t	74
2	22	23c	EM	19	EM	19	Y	59	y	79
2	23	244	NAK	15	NAK	15	U	55	u	75
2	24	248	HT	09	HT	09	I	49	i	69
2	25	24C	SI	0F	SI	0F	O	4F	o	6F
2	26	2E4	DLE	10	DLE	10	P	50	p	70
2	27	2E8	RS	1E	RS	1E	¨ ^	7E	^	5E
2	28	2EC	*	2A	$	24	*	2A	$	24
3	31	250	DC1	11	DC1	11	Q	51	q	71
3	32	258	DC3	13	DC3	13	S	53	s	73
3	33	254	EOT	04	EOT	04	D	44	d	64
3	34	260	ACK	06	ACK	06	F	46	f	66
3	35	264	BEL	07	BEL	07	G	47	g	67
3	36	25C	BS	08	BS	08	H	48	h	68
3	37	268	LF	0A	LF	0A	J	4A	j	6A
3	38	26C	VT	0B	VT	0B	K	4B	k	6B
3	39	274	FF	0C	FF	0C	L	4C	l	6C
3	40	270	CR	0D	CR	0D	M	4D	m	6D
3	41	314	%	25	ù	7C	%	25	ù	7C
3	41A	2B8	£	23	`	60	£	23	`	60
3	42	308	CR	0D	CR	0D	CR	0D	CR	0D
4	43A	2E0	>	3E	<	3C	>	3E	<	3C
4	44	278	ETB	17	ETB	17	W	57	w	77
4	45	27C	CAN	18	CAN	18	X	58	x	78
4	46	280	ETX	03	ETX	03	C	43	c	63
4	47	284	SYN	16	SYN	16	V	56	v	76
4	48	288	STX	02	STX	02	B	42	b	62
4	49	28C	SO	0E	SO	0E	N	4E	n	6E
4	50	290	?	3F	,	2C	?	3F	,	2C
4	51	294	.	2E	;	3B	.	2E	;	3B
4	52	298	/	2F	:	3A	/	2F	:	3A
4	53	29C	+	2B	=	3D	+	2B	=	3D
5	58	310	SP	20	SP	20	SP	20	SP	20
5	60	338	BS	08	BS	08	BS	08	BS	08
5	61	33C	NAK	15	NAK	15	NAK	15	NAK	15

Row	Key#	Rom Ad	Char	C-S	Code	C-S	Char	C	Code	C	Char	S	Code	S	Char	Code
5	62	334	LF		0A		LF		0A		LF		0A			
5	63	30C	VT		0B		VT		0B		VT		0B			

Apple //c

Standard German Keyboard Layout

October 24, 1983



Super][German Keyboard ROM Map — Alpha Lock

Key Num.	Key Cap	Matrix Number	ROM Addr.	Cntl/Shft Char	Cntl/Shft Code	Control Char	Control Code	Shift Char	Shift Code	Normal Char	Normal Code
01	ESC	00	000:	ESC	1B	ESC	1B	ESC	1B	ESC	1B
02	1!	01	004:	!	21	1	31	!	21	1	31
03	2"	02	008:	"	22	2	32	"	22	2	32
04	3§	03	00C:	NUL	00	NUL	00	§	40	3	33
05	4$	04	010:	$	24	4	34	$	24	4	34
07	6&	05	014:	&	26	6	36	&	26	6	36
06	5%	06	018:	%	25	5	35	%	25	5	35
08	7/	07	01C:	/	2F	7	37	/	2F	7	37
09	8(	08	020:	(	28	8	38	(	28	8	38
10	9)	09	024:	)	29	9	39	)	29	9	39
16	TAB	10	028:	HT	09	HT	09	HT	09	HT	09
17	Q	11	02C:	DC1	11	DC1	11	Q	51	Q	51
18	W	12	030:	ETB	17	ETB	17	W	57	W	57
19	E	13	034:	ENQ	05	ENQ	05	E	45	E	45
20	R	14	038:	DC2	12	DC2	12	R	52	R	52
22	Z	15	03C:	SUB	1A	SUB	1A	Z	5A	Z	5A
21	T	16	040:	DC4	14	DC4	14	T	54	T	54
23	U	17	044:	NAK	15	NAK	15	U	55	U	55
24	I	18	048:	HT	09	HT	09	I	49	I	49
25	O	19	04C:	SI	0F	SI	0F	O	4F	O	4F
31	A	20	050:	SOH	01	SOH	01	A	41	A	41
33	D	21	054:	EOT	04	EOT	04	D	44	D	44
32	S	22	058:	DC3	13	DC3	13	S	53	S	53
36	H	23	05C:	BS	08	BS	08	H	48	H	48
34	F	24	060:	ACK	06	ACK	06	F	46	F	46
35	G	25	064:	BEL	07	BEL	07	G	47	G	47
37	J	26	068:	LF	0A	LF	0A	J	4A	J	4A
38	K	27	06C:	VT	0B	VT	0B	K	4B	K	4B
40	Ö	28	070:	FS	1C	FS	1C	Ö	5C	Ö	5C
39	L	29	074:	FF	0C	FF	0C	L	4C	L	4C
44	Y	30	078:	EM	19	EM	19	Y	59	Y	59
45	X	31	07C:	CAN	18	CAN	18	X	58	X	58
46	C	32	080:	ETX	03	ETX	03	C	43	C	43
47	V	33	084:	SYN	16	SYN	16	V	56	V	56
48	B	34	088:	STX	02	STX	02	B	42	B	42
49	N	35	08C:	SO	0E	SO	0E	N	4E	N	4E
50	M	36	090:	CR	0D	CR	0D	M	4D	M	4D
51	,;	37	094:	;	3B	,	2C	;	3B	,	2C
52	.:	38	098:	:	3A	.	2E	:	3A	.	2E
53	-_	39	09C:	US	1F	US	1F	_	5F	-	2D
E1		40	0A0:	/	2F	/	2F	/	2F	/	2F
E2		41	0A4:	BS	08	BS	08	BS	08	BS	08
E3		42	0A8:	0	30	0	30	0	30	0	30
E4		43	0AC:	1	31	1	31	1	31	1	31
E5		44	0B0:	2	32	2	32	2	32	2	32
E6		45	0B4:	3	33	3	33	3	33	3	33
29	#^	46	0B8:	RS	1E	RS	1E	^	5E	#	23
13	´`	47	0BC:	`	60	´	27	`	60	´	27
11	0=	48	0C0:	=	3D	0	30	=	3D	0	30
12	ß ?	49	0C4:	?	3F	ß	7E	?	3F	ß	7E
E7		50	0C8:	)	29	)	29	)	29	)	29
E8		51	0CC:	ESC	1B	ESC	1B	ESC	1B	ESC	1B

E9	52	0D0:	4	34	4	34	4	34	4	34
E10	53	0D4:	5	35	5	35	5	35	5	35
E11	54	0D8:	6	36	6	36	6	36	6	36
E12	55	0DC:	7	37	7	37	7	37	7	37
56 <>	56	0E0:	>	3E	<	3C	>	3E	<	3C
26 P	57	0E4:	DLE	10	DLE	10	P	50	P	50
27 U	58	0E8:	GS	1D	GS	1D	U	5D	U	5D
28 +*	59	0EC:	*	2A	+	2B	*	2A	+	2B
E13	60	0F0:	*	2A	*	2A	*	2A	*	2A
E14	61	0F4:	NAK	15	NAK	15	NAK	15	NAK	15
E15	62	0F8:	8	38	8	38	8	38	8	38
E16	63	0FC:	9	39	9	39	9	39	9	39
E17	64	100:	.	2E	.	2E	.	2E	.	2E
E18	65	104:	+	2B	+	2B	+	2B	+	2B
42 RETURN	66	108:	CR	0D	CR	0D	CR	0D	CR	0D
63 up	67	10C:	VT	0B	VT	0B	VT	0B	VT	0B
58 space	68	110:	SP	20	SP	20	SP	20	SP	20
41 A	69	114:	ESC	1B	ESC	1B	A	5B	A	5B
E19	70	118:	?	3F	?	3F	?	3F	?	3F
E20	71	11C:	SP	20	SP	20	SP	20	SP	20
E21	72	120:	(	28	(	28	(	28	(	28
E22	73	124:	-	2D	-	2D	-	2D	-	2D
E23	74	128:	CR	0D	CR	0D	CR	0D	CR	0D
E24	75	12C:	,	2C	,	2C	,	2C	,	2C
14 delete	76	130:	DEL	7F	DEL	7F	DEL	7F	DEL	7F
62 down	77	134:	LF	0A	LF	0A	LF	0A	LF	0A
60 left	78	138:	BS	08	BS	08	BS	08	BS	08
61 right	79	13C:	NAK	15	NAK	15	NAK	15	NAK	15

Fill all unused locations with A0.

Super][German Keyboard ROM Map -- Upper/Lower Case

Key Num.	Key Cap	Matrix Number	ROM Addr.	Cntl/Shft Char	Cntl/Shft Code	Control Char	Control Code	Shift Char	Shift Code	Normal Char	Normal Code
01	ESC	00	200:	ESC	1B	ESC	1B	ESC	1B	ESC	1B
02	1!	01	204:	!	21	1	31	!	21	1	31
03	2"	02	208:	"	22	2	32	"	22	2	32
04	3 §	03	20C:	NUL	00	NUL	00	§	40	3	33
05	4$	04	210:	$	24	4	34	$	24	4	34
07	6&	05	214:	&	26	6	36	&	26	6	36
06	5%	06	218:	%	25	5	35	%	25	5	35
08	7/	07	21C:	/	2F	7	37	/	2F	7	37
09	8(	08	220:	(	28	8	38	(	28	8	38
10	9)	09	224:	)	29	9	39	)	29	9	39
16	TAB	10	228:	HT	09	HT	09	HT	09	HT	09
17	Q	11	22C:	DC1	11	DC1	11	Q	51	q	71
18	W	12	230:	ETB	17	ETB	17	W	57	w	77
19	E	13	234:	ENQ	05	ENQ	05	E	45	e	65
20	R	14	238:	DC2	12	DC2	12	R	52	r	72
22	Z	15	23C:	SUB	1A	SUB	1A	Z	5A	z	7A
21	T	16	240:	DC4	14	DC4	14	T	54	t	74
23	U	17	244:	NAK	15	NAK	15	U	55	u	75
24	I	18	248:	HT	09	HT	09	I	49	i	69
25	O	19	24C:	SI	0F	SI	0F	O	4F	o	6F
31	A	20	250:	SOH	01	SOH	01	A	41	a	61
33	D	21	254:	EOT	04	EOT	04	D	44	d	64
32	S	22	258:	DC3	13	DC3	13	S	53	s	73
36	H	23	25C:	BS	08	BS	08	H	48	h	68
34	F	24	260:	ACK	06	ACK	06	F	46	f	66
35	G	25	264:	BEL	07	BEL	07	G	47	g	67
37	J	26	268:	LF	0A	LF	0A	J	4A	j	6A
38	K	27	26C:	VT	0B	VT	0B	K	4B	k	6B
40	Ö Ö	28	270:	FS	1C	FS	1C	Ö	5C	ö	7C
39	L	29	274:	FF	0C	FF	0C	L	4C	l	6C
44	Y	30	278:	EM	19	EM	19	Y	59	y	79
45	X	31	27C:	CAN	18	CAN	18	X	58	x	78
46	C	32	280:	ETX	03	ETX	03	C	43	c	63
47	V	33	284:	SYN	16	SYN	16	V	56	v	76
48	B	34	288:	STX	02	STX	02	B	42	b	62
49	N	35	28C:	SO	0E	SO	0E	N	4E	n	6E
50	M	36	290:	CR	0D	CR	0D	M	4D	m	6D
51	,;	37	294:	;	3B	,	2C	;	3B	,	2C
52	.:	38	298:	:	3A	.	2E	:	3A	.	2E
53	-_	39	29C:	US	1F	US	1F		5F	-	2D
E1		40	2A0:	/	2F	/	2F	\	2F	/	2F
E2		41	2A4:	BS	08	BS	08	BS	08	BS	08
E3		42	2A8:	0	30	0	30	0	30	0	30
E4		43	2AC:	1	31	1	31	1	31	1	31
E5		44	2B0:	2	32	2	32	2	32	2	32
E6		45	2B4:	3	33	3	33	3	33	3	33
29	#^	46	2B8:	RS	1E	RS	1E	^	5E	#	23
13	´ `	47	2BC:	`	60	´	27	`	60	´	27
11	0=	48	2C0:	=	3D	0	30	=	3D	0	30
12	ß ?	49	2C4:	?	3F	ß	7E	?	3F	ß	7E
E7		50	2C8:	)	29	)	29	)	29	)	29
E8		51	2CC:	ESC	1B	ESC	1B	ESC	1B	ESC	1B

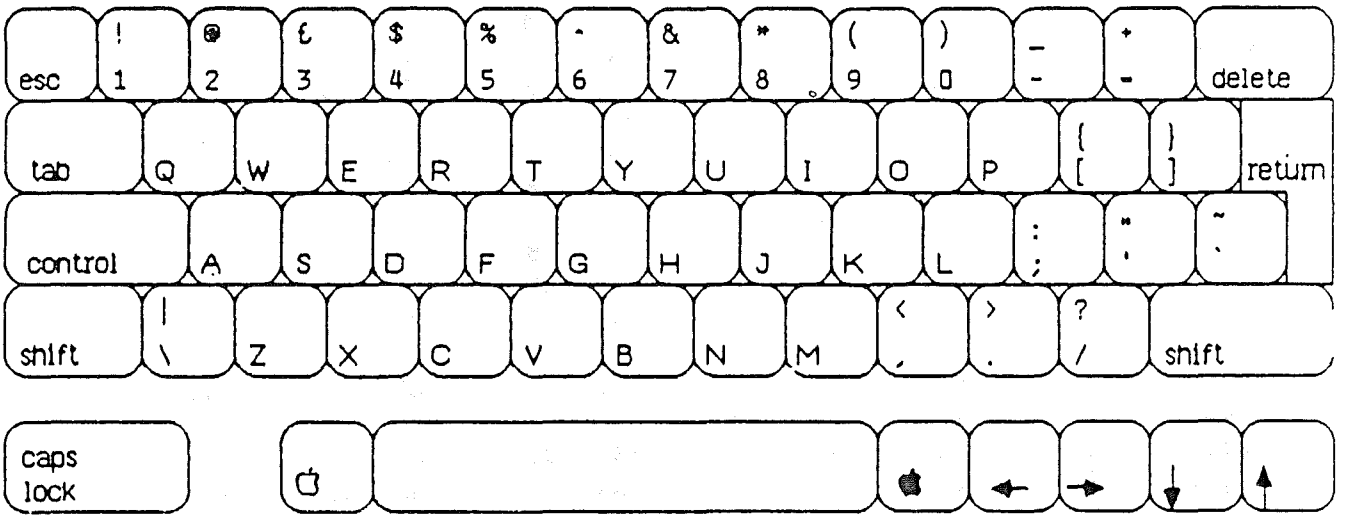
E9	52	2D0:	4	34	4	34	4	34	4	34
E10	53	2D4:	5	35	5	35	5	35	5	35
E11	54	2D8:	6	36	6	36	6	36	6	36
E12	55	2DC:	7	37	7	37	7	37	7	37
56 <>	56	2E0:	>	3E	<	3C	>	3E	<	3C
26 P	57	2E4:	DLE	10	DLE	10	P	50	p	70
27 ü Ü	58	2E8:	GS	1D	GS	1D	Ü	5D	ü	7D
28 +*	59	2EC:	*	2A	+	2B	*	2A	+	2B
E13	60	2F0:	*	2A	*	2A	*	2A	*	2A
E14	61	2F4:	NAK	15	NAK	15	NAK	15	NAK	15
E15	62	2F8:	8	38	8	38	8	38	8	38
E16	63	2FC:	9	39	9	39	9	39	9	39
E17	64	300:	.	2E	.	2E	.	2E	.	2E
E18	65	304:	+	2B	+	2B	+	2B	+	2B
42 RETURN	66	308:	CR	0D	CR	0D	CR	0D	CR	0D
63 up	67	30C:	VT	0B	VT	0B	VT	0B	VT	0B
58 space	68	310:	SP	20	SP	20	SP	20	SP	20
41 ä Ä	69	314:	ESC	1B	ESC	1B	Ä	5B	ä	7B
E19	70	318:	?	3F	?	3F	?	3F	?	3F
E20	71	31C:	SP	20	SP	20	SP	20	SP	20
E21	72	320:	(	28	(	28	(	28	(	28
E22	73	324:	-	2D	-	2D	-	2D	-	2D
E23	74	328:	CR	0D	CR	0D	CR	0D	CR	0D
E24	75	32C:	,	2C	,	2C	,	2C	,	2C
14 delete	76	330:	DEL	7F	DEL	7F	DEL	7F	DEL	7F
62 down	77	334:	LF	0A	LF	0A	LF	0A	LF	0A
60 left	78	338:	BS	08	BS	08	BS	08	BS	08
61 right	79	33C:	NAK	15	NAK	15	NAK	15	NAK	15

Fill all unused locations with A0.

Apple //c

Standard UK Keyboard Layout

October 24, 1983



Notes: Per Neil Davison (October 18, 1983)

Use no symbols on keycaps; instead use:

"shift"

"return"

"caps lock"

Super][British Keyboard ROM Map — Alpha Lock

Key Num.	Key Cap	Matrix Number	ROM Addr.	Cntl/Shft Char	Cntl/Shft Code	Control Char	Control Code	Shift Char	Shift Code	Normal Char	Normal Code
01	ESC	00	000:	ESC	1B	ESC	1B	ESC	1B	ESC	1B
02	1!	01	004:	!	21	1	31	!	21	1	31
03	2@	02	008:	NUL	00	NUL	00	@	40	2	32
04	3£	03	00C:	£	23	3	33	£	23	3	33
05	4$	04	010:	$	24	4	34	$	24	4	34
07	6&	05	014:	RS	1E	RS	1E	^	5E	6	36
06	5%	06	018:	%	25	5	35	%	25	5	35
08	7&	07	01C:	&	26	7	37	&	26	7	37
09	8*	08	020:	*	2A	8	38	*	2A	8	38
10	9(	09	024:	(	28	9	39	(	28	9	39
16	TAB	10	028:	HT	09	HT	09	HT	09	HT	09
17	Q	11	02C:	DC1	11	DC1	11	Q	51	Q	51
18	W	12	030:	ETB	17	ETB	17	W	57	W	57
19	E	13	034:	ENQ	05	ENQ	05	E	45	E	45
20	R	14	038:	DC2	12	DC2	12	R	52	R	52
22	Y	15	03C:	EM	19	EM	19	Y	59	Y	59
21	T	16	040:	DC4	14	DC4	14	T	54	T	54
23	U	17	044:	NAK	15	NAK	15	U	55	U	55
24	I	18	048:	HT	09	HT	09	I	49	I	49
25	O	19	04C:	SI	0F	SI	0F	O	4F	O	4F
31	A	20	050:	SOH	01	SOH	01	A	41	A	41
33	D	21	054:	EOT	04	EOT	04	D	44	D	44
32	S	22	058:	DC3	13	DC3	13	S	53	S	53
36	H	23	05C:	BS	08	BS	08	H	48	H	48
34	F	24	060:	ACK	06	ACK	06	F	46	F	46
35	G	25	064:	BEL	07	BEL	07	G	47	G	47
37	J	26	068:	LF	0A	LF	0A	J	4A	J	4A
38	K	27	06C:	VT	0B	VT	0B	K	4B	K	4B
40	;:	28	070:	:	3A	;	3B	:	3A	;	3B
39	L	29	074:	FF	0C	FF	0C	L	4C	L	4C
44	Z	30	078:	SUB	1A	SUB	1A	Z	5A	Z	5A
45	X	31	07C:	CAN	18	CAN	18	X	58	X	58
46	C	32	080:	ETX	03	ETX	03	C	43	C	43
47	V	33	084:	SYN	16	SYN	16	V	56	V	56
48	B	34	088:	STX	02	STX	02	B	42	B	42
49	N	35	08C:	SO	0E	SO	0E	N	4E	N	4E
50	M	36	090:	CR	0D	CR	0D	M	4D	M	4D
51	,<	37	094:	<	3C	,	2C	<	3C	,	2C
52	.>	38	098:	>	3E	.	2E	>	3E	.	2E
53	/?	39	09C:	?	3F	/	2F	?	3F	/	2F
E1		40	0A0:	/	2F	/	2F	/	2F	/	2F
E2		41	0A4:	BS	08	BS	08	BS	08	BS	08
E3		42	0A8:	0	30	0	30	0	30	0	30
E4		43	0AC:	1	31	1	31	1	31	1	31
E5		44	0B0:	2	32	2	32	2	32	2	32
E6		45	0B4:	3	33	3	33	3	33	3	33
29	`~	46	0B8:	~	7E	`	60	~	7E	`	60
13	=+	47	0BC:	+	2B	=	3D	+	2B	=	3D
11	0)	48	0C0:	)	29	0	30	)	29	0	30
12	-_	49	0C4:	US	1F	US	1F	_	5F	-	2D
E7		50	0C8:	)	29	)	29	)	29	)	29
E8		51	0CC:	ESC	1B	ESC	1B	ESC	1B	ESC	1B

E9	52	0D0:	4	34	4	34	4	34	4	34
E10	53	0D4:	5	35	5	35	5	35	5	35
E11	54	0D8:	6	36	6	36	6	36	6	36
E12	55	0DC:	7	37	7	37	7	37	7	37
56 |	56	0E0:	FS	1C	FS	1C	|	7C	\	5C
26 P	57	0E4:	DLE	10	DLE	10	P	50	P	50
27 [{	58	0E8:	ESC	1B	ESC	1B	{	7B	[	5B
28]}	59	0EC:	GS	1D	GS	1D	}	7D	]	5D
E13	60	0F0:	*	2A	*	2A	*	2A	*	2A
E14	61	0F4:	NAK	15	NAK	15	NAK	15	NAK	15
E15	62	0F8:	8	38	8	38	8	38	8	38
E16	63	0FC:	9	39	9	39	9	39	9	39
E17	64	100:	.	2E	.	2E	.	2E	.	2E
E18	65	104:	+	2B	+	2B	+	2B	+	2B
42 RETURN	66	108:	CR	0D	CR	0D	CR	0D	CR	0D
63 up	67	10C:	VT	0B	VT	0B	VT	0B	VT	0B
58 space	68	110:	SP	20	SP	20	SP	20	SP	20
41 '"	69	114:	"	22	'	27	"	22	'	27
E19	70	118:	?	3F	?	3F	?	3F	?	3F
E20	71	11C:	SP	20	SP	20	SP	20	SP	20
E21	72	120:	(	28	(	28	(	28	(	28
E22	73	124:	-	2D	-	2D	-	2D	-	2D
E23	74	128:	CR	0D	CR	0D	CR	0B	CR	0D
E24	75	12C:	,	2C	,	2C	,	2C	,	2C
14 delete	76	130:	DEL	7F	DEL	7F	DEL	7F	DEL	7F
62 down	77	134:	LF	0A	LF	0A	LF	0A	LF	0A
60 left	78	138:	BS	08	BS	08	BS	08	BS	08
61 right	79	13C:	NAK	15	NAK	15	NAK	15	NAK	15

Fill all unused locations with A0.

Super][British Keyboard ROM Map -- Upper/Lower Case

Key Num.	Key Cap	Matrix Number	ROM Addr.	Cntl/Shft Char	Cntl/Shft Code	Control Char	Control Code	Shift Char	Shift Code	Normal Char	Normal Code
01	ESC	00	200:	ESC	1B	ESC	1B	ESC	1B	ESC	1B
02	1!	01	004:	!	21	1	31	!	21	1	31
03	2@	02	008:	NUL	00	NUL	00	@	40	2	32
04	3£	03	00C:	£	23	3	33	£	23	3	33
05	4$	04	010:	$	24	4	34	$	24	4	34
07	6&	05	014:	RS	1E	RS	1E	^	5E	6	36
06	5%	06	018:	%	25	5	35	%	25	5	35
08	7&	07	01C:	&	26	7	37	&	26	7	37
09	8*	08	020:	*	2A	8	38	*	2A	8	38
10	9(	09	024:	(	28	9	39	(	28	9	39
16	TAB	10	228:	HT	09	HT	09	HT	09	HT	09
17	Q	11	22C:	DC1	11	DC1	11	Q	51	q	71
18	W	12	230:	ETB	17	ETB	17	W	57	w	77
19	E	13	234:	ENQ	05	ENQ	05	E	45	e	65
20	R	14	238:	DC2	12	DC2	12	R	52	r	72
22	Y	15	03C:	EM	19	EM	19	Y	59	y	79
21	T	16	240:	DC4	14	DC4	14	T	54	t	74
23	U	17	244:	NAK	15	NAK	15	U	55	u	75
24	I	18	248:	HT	09	HT	09	I	49	i	69
25	O	19	24C:	SI	0F	SI	0F	O	4F	o	6F
31	A	20	250:	SOH	01	SOH	01	A	41	a	61
33	D	21	254:	EOT	04	EOT	04	D	44	d	64
32	S	22	258:	DC3	13	DC3	13	S	53	s	73
36	H	23	25C:	BS	08	BS	08	H	48	h	68
34	F	24	260:	ACK	06	ACK	06	F	46	f	66
35	G	25	264:	BEL	07	BEL	07	G	47	g	67
37	J	26	268:	LF	0A	LF	0A	J	4A	j	6A
38	K	27	26C:	VT	0B	VT	0B	K	4B	k	6B
40	;:	28	070:	:	3A	;	3B	:	3A	;	3B
39	L	29	274:	FF	0C	FF	0C	L	4C	l	6C
44	Z	30	278:	SUB	1A	SUB	1A	Z	5A	z	~~5A~~ 7A
45	X	31	27C:	CAN	18	CAN	18	X	58	x	78
46	C	32	280:	ETX	03	ETX	03	C	43	c	63
47	V	33	284:	SYN	16	SYN	16	V	56	v	76
48	B	34	288:	STX	02	STX	02	B	42	b	62
49	N	35	28C:	SO	0E	SO	0E	N	4E	n	6E
50	M	36	290:	CR	0D	CR	0D	M	4D	m	6D
51	,<	37	094:	<	3C	,	2C	<	3C	,	2C
52	.>	38	098:	>	3E	.	2E	>	3E	.	2E
53	/?	39	09C:	?	3F	/	2F	?	3F	/	2F
E1		40	2A0:	/	2F	/	2F	/	2F	/	2F
E2		41	2A4:	BS	08	BS	08	BS	08	BS	08
E3		42	2A8:	0	30	0	30	0	30	0	30
E4		43	2AC:	1	31	1	31	1	31	1	31
E5		44	2B0:	2	32	2	32	2	32	2	32
E6		45	2B4:	3	33	3	33	3	33	3	33
29	`~	46	0B8:	~	7E	`	60	~	7E	`	60
13	=+	47	0BC:	+	2B	=	3D	+	2B	=	3D
11	0)	48	0C0:	)	29	0	30	)	29	0	30
12	-_	49	0C4:	US	1F	US	1F	_	5F	-	2D
E7		50	2C8:	)	29	)	29	)	29	)	29
E8		51	2CC:	ESC	1B	ESC	1B	ESC	1B	ESC	1B

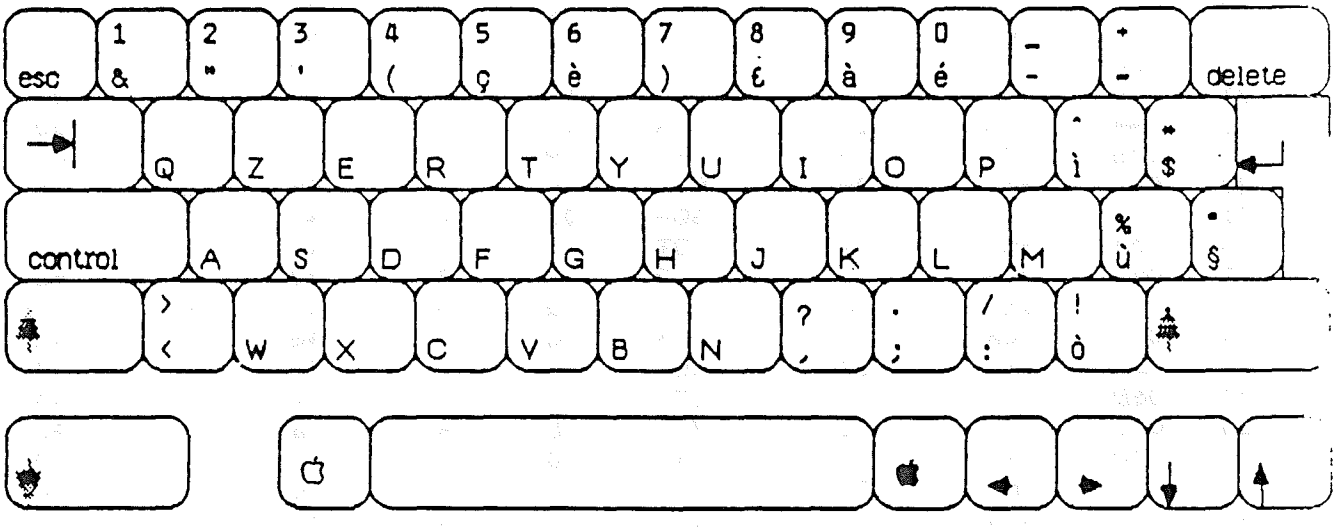
E9		52	2D0:	4	34	4	34	4	34	4	34
E10		53	2D4:	5	35	5	35	5	35	5	35
E11		54	2D8:	6	36	6	36	6	36	6	36
E12		55	2DC:	7	37	7	37	7	37	7	37
56	|	56	0E0:	FS	1C	FS	1C	|	7C	\	5C
26	P	57	0E4:	DLE	10	DLE	10	P	50	p	70
27	[{	58	0E8:	ESC	1B	ESC	1B	{	7B	[	5B
28	]}	59	0EC:	GS	1D	GS	1D	}	7D	]	5D
E13		60	2F0:	*	2A	*	2A	*	2A	*	2A
E14		61	2F4:	NAK	15	NAK	15	NAK	15	NAK	15
E15		62	2F8:	8	38	8	38	8	38	8	38
E16		63	2FC:	9	39	9	39	9	39	9	39
E17		64	300:	.	2E	.	2E	.	2E	.	2E
E18		65	304:	+	2B	+	2B	+	2B	+	2B
42	RETURN	66	308:	CR	0D	CR	0D	CR	0D	CR	0D
63	up	67	30C:	VT	0B	VT	0B	VT	0B	VT	0B
58	space	68	310:	SP	20	SP	20	SP	20	SP	20
41	'"	69	314:	"	22	'	27	"	22	'	27
E19		70	318:	?	3F	?	3F	?	3F	?	3F
E20		71	31C:	SP	20	SP	20	SP	20	SP	20
E21		72	320:	(	28	(	28	(	28	(	28
E22		73	324:	-	2D	-	2D	-	2D	-	2D
E23		74	328:	CR	0D	CR	0D	CR	0D	CR	0D
E24		75	32C:	,	2C	,	2C	,	2C	,	2C
14	delete	76	330:	DEL	7F	DEL	7F	DEL	7F	DEL	7F
62	down	77	334:	LF	0A	LF	0A	LF	0A	LF	0A
60	left	78	338:	BS	08	BS	08	BS	08	BS	08
61	right	79	33C:	NAK	15	NAK	15	NAK	15	NAK	15

Fill all unused locations with A0.

Apple //c

Standard Italy Keyboard Layout

October 24, 1983



Notes: 1) Uses "Shift lock" not "Caps Lock" -- All keys are shifted.

2) Alternate character set is U.S. but kbd layout is identical to the Italian -- only characters which are not common to both character sets change.

3) The following characters change to their US equivalents:

Hex	Italian	US	Hex	Italian	US
23	£	#	60	ù	`
40	§	@	7B	à	{
5B	°	[	7C	ò	|
5C	ç	\	7D	è	}
5D	é	]	7E	ì	~

File: ITALIANUC Pag

Report: ROMCODE DEC 1,

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
1	01	000	ESC	1B	ESC	1B	ESC	1B	ESC	1B
1	02	004	1	31	1	31	1	31	1	31
1	03	008	2	32	2	32	2	32	2	32
1	04	00C	3	33	3	33	3	33	3	33
1	05	010	4	34	4	34	4	34	4	34
1	07	014	6	36	6	36	6	36	6	36
1	06	018	FS	1C	FS	1C	5	35	5	35
1	08	01C	7	37	7	37	7	37	7	37
1	09	020	8	38	8	38	8	38	8	38
1	10	024	9	39	9	39	9	39	9	39
2	16	028	HT	09	HT	09	HT	09	HT	09
2	17	02C	DC1	11	DC1	11	Q	51	Q	51
2	18	030	SUB	1A	SUB	1A	Z	5A	Z	5A
2	19	034	ENQ	05	ENQ	05	E	45	E	45
2	20	038	DC2	12	DC2	12	R	52	R	52
2	22	03C	EM	19	EM	19	Y	59	Y	59
2	21	040	DC4	14	DC4	14	T	54	T	54
2	23	044	NAK	15	NAK	15	U	55	U	55
2	24	048	HT	09	HT	09	I	49	I	49
2	25	04C	SI	0F	SI	0F	O	4F	O	4F
3	31	050	SOH	01	SOH	01	A	41	A	41
3	33	054	EOT	04	EOT	04	D	44	D	44
3	32	058	DC3	13	DC3	13	S	53	S	53
3	36	05C	BS	08	BS	08	H	48	H	48
3	34	060	ACK	06	ACK	06	F	46	F	46
3	35	064	BEL	07	BEL	07	G	47	G	47
3	37	068	LF	0A	LF	0A	J	4A	J	4A
3	38	06C	VT	0B	VT	0B	K	4B	K	4B
3	40	070	CR	0D	CR	0D	M	4D	M	4D
3	39	074	FF	0C	FF	0C	L	4C	L	4C
4	44	078	ETB	17	ETB	17	W	57	W	57
4	45	07C	CAN	18	CAN	18	X	58	X	58
4	46	080	ETX	03	ETX	03	C	43	C	43
4	47	084	SYN	16	SYN	16	V	56	V	56
4	48	088	STX	02	STX	02	B	42	B	42
4	49	08C	SO	0E	SO	0E	N	4E	N	4E
4	50	090	?	3F	?	3F	?	3F	?	3F
4	51	094	.	2E	.	2E	.	2E	.	2E
4	52	098	/	2F	/	2F	/	2F	/	2F
4	53	09C	!	21	!	21	!	21	!	21
3	41A	0B8	NUL	00	ESC	1B	°	5B	°	5B
1	13	0BC	+	2B	+	2B	+	2B	+	2B
1	11	0C0	GS	1D	GS	1D	0	30	0	30
1	12	0C4	US	1F	US	1F	_	5F	_	5F
4	43A	0E0	>	3E	>	3E	>	3E	>	3E
2	26	0E4	DLE	10	DLE	10	P	50	P	50
2	27	0E8	RS	1E	RS	1E	^	5E	^	5E
2	28	0EC	*	2A	*	2A	*	2A	*	2A
3	42	108	CR	0D	CR	0D	CR	0D	CR	0D
5	63	10C	VT	0B	VT	0B	VT	0B	VT	0B

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
5	58	110	SP	20	SP	20	SP	20	SP	20
~	41	114	%	25	%	25	%	25	%	25
	14	130	DEL	7F	DEL	7F	DEL	7F	DEL	7F
5	62	134	LF	0A	LF	0A	LF	0A	LF	0A
5	60	138	BS	08	BS	08	BS	08	BS	08
5	61	13C	NAK	15	NAK	15	NAK	15	NAK	15

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
1	01	200	ESC	1B	ESC	1B	ESC	1B	ESC	1B
1	02	204	1	31	&	26	1	31	&	26
1	03	208	2	32	"	22	2	32	"	22
1	04	20C	3	33	'	27	3	33	'	27
1	05	210	4	34	(	28	4	34	(	28
1	07	214	6	36	è	7D	6	36	è	7D
1	06	218	FS	1C	FS	1C	5	35	ç	5C
1	08	21C	7	37	)	29	7	37	)	29
1	09	220	8	38	£	23	8	38	£	23
1	10	224	9	39	à	7B	9	39	à	7B
2	16	228	HT	09	HT	09	HT	09	HT	09
2	17	22C	DC1	11	DC1	11	Q	51	q	71
2	18	230	SUB	1A	SUB	1A	Z	5A	z	7A
2	19	234	ENQ	05	ENQ	05	E	45	e	65
2	20	238	DC2	12	DC2	12	R	52	r	72
2	22	23c	EM	19	EM	19	Y	59	y	79
2	21	240	DC4	14	DC4	14	T	54	t	74
2	23	244	NAK	15	NAK	15	U	55	u	75
2	24	248	HT	09	HT	09	I	49	i	69
2	25	24C	SI	0F	SI	0F	O	4F	o	6F
3	31	250	SOH	01	SOH	01	A	41	a	61
3	33	254	EOT	04	EOT	04	D	44	d	64
3	32	258	DC3	13	DC3	13	S	53	s	73
3	36	25C	BS	08	BS	08	H	48	h	68
3	34	260	ACK	06	ACK	06	F	46	f	66
3	35	264	BEL	07	BEL	07	G	47	g	67
3	37	268	LF	0A	LF	0A	J	4A	j	6A
3	38	26C	VT	0B	VT	0B	K	4B	k	6B
3	40	270	CR	0D	CR	0D	M	4D	m	6D
3	39	274	FF	0C	FF	0C	L	4C	l	6C
4	44	278	ETB	17	ETB	17	W	57	w	77
4	45	27C	CAN	18	CAN	18	X	58	x	78
4	46	280	ETX	03	ETX	03	C	43	c	63
4	47	284	SYN	16	SYN	16	V	56	v	76
4	48	288	STX	02	STX	02	B	42	b	62
4	49	28C	SO	0E	SO	0E	N	4E	n	6E
4	50	290	?	3F	,	2C	?	3F	,	2C
4	51	294	.	2E	;	3B	.	2E	;	3B
4	52	298	/	2F	:	3A	/	2F	:	3A
4	53	29C	!	21	ò	7C	!	21	ò	7C
3	41A	2B8	ESC	1B	NUL	00	°	5B	§	40
1	13	2BC	+	2B	=	3D	+	2B	=	3D
1	11	2C0	GS	1D	GS	1D	ø	30	é	5D
1	12	2C4	US	1F	US	1F	_	5F	-	2D
4	43A	2E0	>	3E	<	3C	>	3E	<	3C
2	26	2E4	DLE	10	DLE	10	P	50	p	70
2	27	2E8	RS	1E	RS	1E	^	5E	ì	7E
2	28	2EC	*	2A	$	24	*	2A	$	24
3	42	308	CR	0D	CR	0D	CR	0D	CR	0D
5	63	30C	VT	0B	VT	0B	VT	0B	VT	0B

File: ITALIANLC

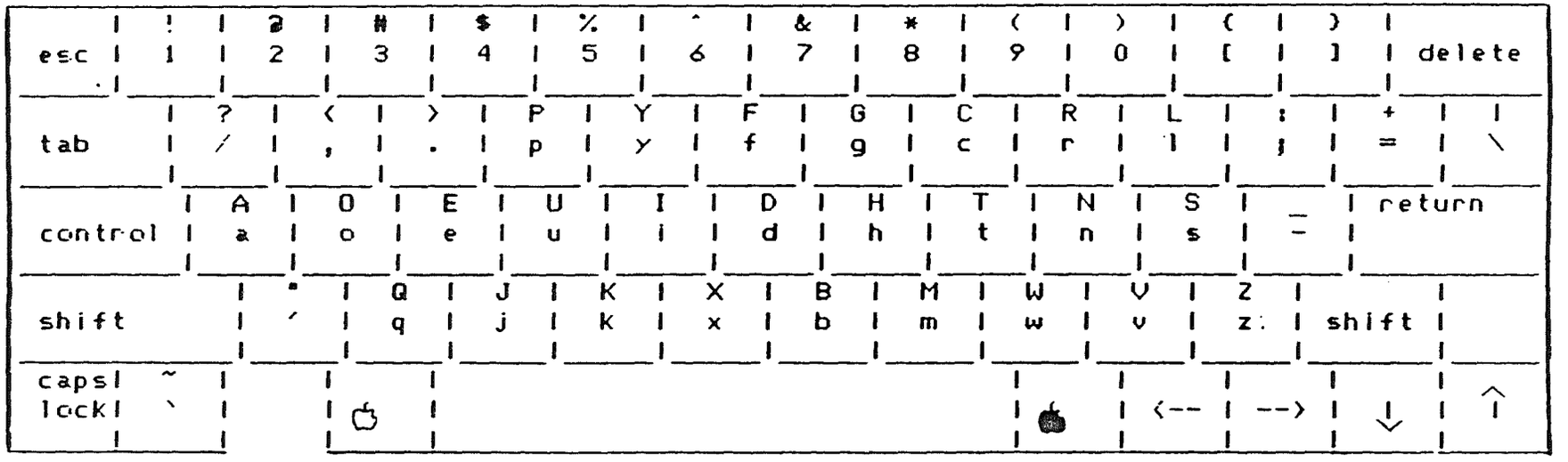
Report: ROMCODE

DEC

Row	Key#	Rom Ad	Char C-S	Code C-S	Char C	Code C	Char S	Code S	Char	Code
5	58	310	SP	20	SP	20	SP	20	SP	20
	41	314	%	25	ù	60	%	25	ù	60
	14	330	DEL	7F	DEL	7F	DEL	7F	DEL	7F
5	62	334	LF	0A	LF	0A	LF	0A	LF	0A
5	60	338	BS	08	BS	08	BS	08	BS	08
5	61	33C	NAK	15	NAK	15	NAK	15	NAK	15

Apple //c Technical Note #4

Corrected DVORAK Keyboard Layout



This is the Dvorak keyboard layout as implemented on the Apple //c computer when the keyboard switch is depressed. There has only been one version of the keyboard layout ROM. All Apple //c's, pre-release and final production machines, have the above layout and no other. The pre-release documentation was in error as is the final Apple //c Reference Manual.

Apple IIc Delta Guide

Table of Contents

Introduction

This document compares the Apple IIc to the Apple IIe, but it also reiterates most of differences between the Apple IIe and the Apple II Plus that were originally noted in the *Guide to the New Features of the Apple IIe* (Apple Product Number A2F2114). In addition, it points out differences between the Apple II and II Plus.

This draft does not include a list of the keyboard and video character sets and other large tables of information. Unless otherwise noted, this information can be found in the *Apple IIe Reference Manual*.

The keyboard and character set differences between different countries' models of the Apple IIc are the same as for the IIe. The *International Supplement to the Apple IIe Owner's Manual* (Part number 030-0525) contains tables and illustrations describing these differences. Note, however, that the Apple IIc has NTSC video circuitry inside the case for all countries; external PAL (and presumably SECAM) video adapters will get their signals from the video expansion connector.

Categories

The characteristics that vary from one machine to another fall under a handful of categories, starting with concrete physical elements and ending with more abstract and technical items:

- Equations

  each machine equals its predecessor plus or minus certain overall characteristics—merely an overview

- External physical

  keyboard layout and front of machine
  sides (yes, sides)
  top (removable or no)
  back panel

- Internal physical

  slots/no slots
  game I/O, aux video pins, LEDs, etc.
  power supply
  disk drive
  speaker

- Input and output

  keyboard character sets
  display character sets
  display modes
  cassette I/O
  disk I/O
  game I/O
  mouse input

- Hardware in general

  type of central processing unit
  amount and address ranges of RAM
  amount and address ranges or ROM
  power supplies

- Firmware in general

  monitor
  video firmware
  diagnostic firmware
  slot/port firmware

- Software in general

  languages
  operating systems

- Hardware specifics

  important RAM locations
  hardware locations
  important ROM locations
  use of ICs (customs, hybrids, sockets)
  signals available to the outside world

Equations

These equations are merely an overview of what each model of Apple II is with respect to its predecessor. The remainder of this guide spells out these differences in detail.

Note: These equations are in terms of functional equivalence, not strict equality. For example,

Apple IIe = Apple II Plus + Language Card

does not mean there is an actual language card or slot—just that the one machine functions as if it were the other with such a card in a slot.

There is a related document (a *Configuration Guide*) that describes how to configure an Apple IIe to make it (almost) equivalent to an Apple IIc..

Apple II Plus	=	Apple II + Autostart ROM + Applesoft firmware + 48K RAM standard
		Apple II - Integer BASIC firmware - Old Monitor ROM

Apple IIe	=	Apple II Plus + language card + additional 16K RAM + 80-column firmware + built-in diagnostics + full ASCII keyboard + internal power-on light + FCC EMC approval + improved back panel + 9-pin back panel game connector + auxiliary slot (with possibility of 80-column card + extra 64K RAM)
		Apple II Plus - slot 0
Apple IIc	=	Apple IIe + extended 80-column text card + 40/80 column switch + language switch + disk light + disk controller port + disk drive + mouse port + serial printer port + serial communication port + built-in port firmware + video expansion connector
		Apple IIe - removable cover - slots 1 to 7 - auxiliary slot - internal power-on light - cassette I/O connectors - internal game I/O connector (hence no game output) - RF modulator connector - auxiliary video pin - diagnostic firmware - miniassembler - monitor cassette support

External Physical

The Apple II and II Plus were identical in external appearance. The Apple IIe and Apple IIc differ from the earlier machines in their keyboard layouts and back panels.

Keyboard

The Apple II and II Plus have identical 52-key keyboards. The Apple IIe and Apple IIc keyboards have the same 63-key, full ASCII keyboard layouts, with new and repositioned keys and characters compared to the Apple II and II Plus. While the Apple II and II Plus have a REPT key, the Apple IIe and IIc have an auto-repeat feature built into each character key.

The Apple IIc has additional switches near its keyboard: one for changing between 40-column and 80-column displays, the other for selecting keyboard layouts (Sholes versus Dvorak on USA models) or keyboard layout and character set (on international models).

The power-on light position differs for the Apple II/II Plus, Apple IIe and Apple IIc. The RESET key also appears in different positions.

Some Apple II and II Plusses have a slide switch inside the case, near the edge of the cover, for selecting whether or not RESET has to be accompanied by CONTROL to work. On the Apple IIe and Apple IIc, there is no choice: CONTROL-RESET works, and RESET alone does not.

Some notable differences in key captions:

	ESCAPE	TAB	CONTROL	SHIFT	CAPS LOCK	DELETE	RETURN	RESET	Other
Apple II	ESC	n/a	CTRL	SHIFT	n/a	n/a	RETURN	RESET	REPT
Apple II +	ESC	n/a	CTRL	SHIFT	n/a	n/a	RETURN	RESET	REPT
Early IIe†	ESC	TAB	CONTROL	SHIFT	CAPS LOCK	DELETE	RETURN	RESET	Apple keys
Later IIe†	Esc	Tab	Control	Shift	Caps Lock	Delete	Return	Reset	Apple keys
Europe IIe	Esc	→|	Control	↑	↓	Delete	←	Reset	Apple keys

†Early Apple IIe's had "two-shot" injection-molded keys (until about June, 1983). After that, manufacturing switched over to a "sublimation" process for applying captions to keys, and changed the captions.

Back Panel

The Apple II and II Plus have three deep notches and two shallow ones on their back panels. The Apple IIe has a metal back panel with 12 numbered rectangular openings with pop-out inserts.

The Apple II, II Plus, and IIe have a video-output phono jack and mini-phono jacks for cassette input and cassette output. The Apple IIe has a DB-9 game input connector that the Apple II and II Plus do not have.

The Apple IIc has the following back-panel connectors, moving from left to right as viewed from the back:

- a game input DB-9 (like the IIe) that is also for the mouse
- a 5-pin DIN connector for serial input and output (Port 2)
- a video expansion output DB-15 for RGB monitor adapter, etc.
- a video output phono jack (same as on all other Apple II's)
- a DB-19 connector for connecting a second disk drive (like IIe)

- a 5-pin DIN connector for serial input and output (Port 1)
- a special recessed male 7-pin DIN connector for 12-volt DC power input (unlike any of the other Apple II's)

The power switch is in the same position (left rear corner) and same orientation (push in top to turn on) for all Apple II's.

Internal Physical

The internal layout of the Apple IIc is irrelevant to this discussion: the user is not to open the Apple IIc case.

The Apple IIe internal layout differs from that of the Apple II and II Plus in several general ways. There are, of course, far fewer components:

- Component layout is different.
- There is no place for plug-in ROMs (like the Programmer's Aid ROM).
- Cards that had a connection on the main logic board on the II and II Plus will not work on the IIe.
- There is a power-on light near the back panel.
- Slot 0 is gone.
- The auxiliary slot is set away from the back panel.

Slots/No Slots

The Apple II and II Plus have 8 identical slots; the IIe has 7 identical slots plus a 60-pin auxiliary slot for video, add-on memory, and test cards. The Apple IIc has no slots; instead, it has built-in hardware and firmware equivalents to slots with cards in them. These are called ports on the Apple IIc.

Game I/O and Other Connectors

The Apple II, II Plus, and IIe have a 16-pin game I/O connector inside the case that supports 3 switch inputs, 4 analog (paddle) inputs, and 4 annunciator outputs. The Apple IIe and IIc have a DB-9 back-panel connector that supports the 3 switch inputs and 4 paddle inputs (2 on the Apple IIc). The Apple IIc does not support the 4 annunciator outputs.

Power Supply

The power supplies for the Apple II, II Plus, and IIe are basically identical; the one for the Apple IIc is quite different from the rest. For further comparisons, see the section under "Hardware in General."

Disk Drive

All of the Apple II series computers are designed to operate with a Disk II drive or its equivalent: 16 sectors, 35 tracks, and so on.

Speaker

The Apple IIe has the same size speaker as the II and II Plus, although it is face down and baffled better. The Apple IIc has a smaller speaker, and, in addition, has a 2-channel (but monaural) mini-phone jack for headphones (which disconnects the internal speaker when something is plugged into it) and a volume control.

Input and Output

This section describes the variations in character sets and other I/O among the Apple II models.

Keyboard Character Sets

The Apple II and II Plus keyboard character sets are the same. They are described in the *Apple II Reference Manual*.

The Apple IIe and IIc keyboard character sets are the same: full ASCII. The standard (Sholes) layout and key assignments are described in the *Apple IIe Reference Manual*. The Dvorak layout and key assignments will be described in the *Apple IIc Reference Manual*.

Display Character Sets

The Apple II and II Plus display character sets are the same: 64 characters of uppercase ASCII (see the *Apple II Reference Manual*). Both the Apple IIe and IIc make available this character set with the addition of lowercase (called the primary set) and an alternate character set (which has inverse lowercase at the expense of flashing characters). Both these sets are described in the *Apple IIe Reference Manual*.

Display Modes

All models have 40-column text mode, low-resolution graphics mode, mixed low-res and 40-column text mode, and high-resolution graphics mode. The Apple IIe (Rev B motherboard) with 80-column text card, and the Apple IIc also have double-high-resolution graphics mode.

Cassette I/O

The Apple II, II Plus and IIe all have cassette input and output jacks, memory locations, and monitor support. The Apple IIc does not.

Disk I/O

The Apple II, II Plus, and IIe can support up to 6 (4 is recommended maximum) disk drives attached to controller cards plugged into slots 6, 5 and 4. The Apple IIc supports its built-in drive (treated as slot 6 drive 1) and one external disk drive (treated as slot 6 drive 2, or as slot 7 drive 1 for external-drive startup purposes.

Game I/O

The Apple II, II Plus, and IIe support game input and output via a 16-pin Dual Inline Pin (DIP) connector inside the case. The Apple IIe and IIc both support game input via a DB-9 connector on their back panels.

Mouse Input

The Apple IIc provides built-in firmware support for a mouse connected to the DB-9 game/mouse connector. The Apple IIe will provide interface card firmware support for a mouse connected to a DB-9 connector that the user installs with the card.

Hardware in General

Type of CPU

The Apple II and II Plus CPU is the 6502. The Apple IIe uses a 6502A, which is capable of a faster clock speed than 1 megahertz (because it is hand-selected from 6502 production), but in fact is not clocked faster than that in the Apple IIe.

The Apple IIc uses the 65C02 as its CPU: this is a redesigned CMOS CPU that has 27 new instructions, new addressing modes, and for some instructions a differing execution scheme. Programs written for the Apple IIc will run on the earlier machines only if they do not contain instructions unique to the 65C02.

Amount and Address Ranges of RAM

Apple II's had as little as 4K of RAM at the time of purchase, but could be upgraded to as much as 48K of RAM by replacing one or more rows of 4 kilobit chips with the (then) newer and noticeably costlier 16 kilobit chips. Changing a matched set of jumper blocks completed the address mapping portion of the conversion. This process is described in the *Apple II Reference Manual*.

The Apple II Plus has 48K of RAM ($0000 through $BFFF) as a standard feature. Addresses $C000 through $FFFF are occupied by ROM only.

Installing an Apple Language Card in an Apple II or II Plus adds the 16K of RAM from $C000 through $FFFF.

The Apple IIe has a full 64K of RAM. The top 12K addresses overlap with the ROM addresses $D000 through $FFFF. There is an additional area of 4K from $D000 through $DFFF. This arrangement is equivalent to an Apple II Plus with an Apple Language Card installed. A program selects between the RAM and

ROM address spaces and between the $Dxxx banks by changing soft switches located in memory. (This process is often called "bank switching.")

With an Apple 80-column Text Card installed in its auxiliary slot, an Apple IIe has an additional 1K of RAM available, for displaying the other 40 columns of 80-column text.

With an Apple Extended 80-Column Text Card installed in its auxiliary slot, an Apple IIe has an additional 64K of RAM available, although no more than half of the 128K of RAM space is available at any given time. Soft switches located in memory control these address space selections.

The RAM in the Apple IIc is equivalent to the RAM in an Apple IIe with an Extended 80-column Card (in other words, with 64K + 64K).

Amount and Address Ranges of ROM

The Apple II and II Plus have from 2K to 12K of firmware in ROM. The uppermost addresses ($F800 through $FFFF) are always used, while other address ranges are optional. Users can plug their own ROMs into the sockets provided. The ROM address range is from $D000 through $FFFF.

The Apple IIe has 16K of ROM (addresses $C100 through $FFFF; page $C0 addresses are for I/O hardware). ROM addresses $C300 through $C3FF (normally assigned to the ROM in a card in slot 3) and $C800 through $CFFF contain 80-column video firmware; ROM addresses $C100 through $C2FF and $C400 through $C7FF (normally assigned to the ROM on cards in slots 1, 2, 4, 5, 6 and 7) contain built-in self-test routines.

A soft switch controls whether the video firmware or slot 3 card ROM is active. Invoking the self-tests with 🍎-CONTROL-RESET causes the self-test firmware to take over the slot ROM address spaces.

The Apple IIc ROM also uses the 16K from $C100 through $FFFF, and its 80-column video firmware occupies the same addresses as on the IIe. However, there are no built-in self-tests. Instead, addresses $C100 through $C2FF and $C400 through $C7FF contain the firmware supporting the four built-in I/O ports (printer, communication, mouse, and disk).

Power Supplies

The power supplies for the Apple II, II Plus, and IIe are essentially the same: they convert 110 VAC (220 VAC on most international models) to the voltages required by the circuitry. The Apple IIc, on the other hand, has an external floor transformer that converts 110 VAC (or 220 VAC) to 12 VDC (nominal); the internal power supply then derives the required voltages.

Firmware in General

This section discusses overall blocks of firmware, not about individual routines and their entry points. A full description of those will appear in the *Apple IIc Reference Manual*.

Monitor

The Apple II comes with the so-called Old Monitor ROM, which would put the user into the monitor (* prompt) at startup. The resident interpreter is for Integer BASIC, with ROM space left over for other firmware (such as programmer's aids).

The Apple II Plus, IIe, and IIc come with the Autostart ROM, which tries to load software from the highest slot containing a Disk II controller card or its equivalent. If this attempt fails, the autostart monitor puts the user in the resident Applesoft interpreter (] prompt).

Video Firmware

The video firmware for the Apple IIc is identical to that for the IIe. Because the Apple IIc has no slots, the 80-column video firmware is always present (switched in); there is no possibility of conflict with firmware on a card in slot 3. Also note that there is only one $C800-$CFFF address space: this, too, belongs to 80-column video firmware.

Diagnostic Firmware

Apple II and II Plus do not have built-in diagnostics. The IIe does; it is invoked by pressing 🍎-CONTROL-RESET. The Apple IIc has a 🍎 key, too, but no built-in diagnostics.

Slot/Port Firmware

The Apple IIc is the only Apple II of the four that has built-in firmware for slots other than "slot 3" (80-column video). In fact, the Apple IIc has hardware, firmware and back-panel connectors that provide the equivalent of:

- a subset of Super Serial Card hardware and firmware, preconfigured for a 1200-baud (maybe 9600-baud) printer in slot 1, with a 5-pin DIN back-panel connector;
- a subset of Super Serial Card hardware and firmware, preconfigured for a 300-baud modem in slot 2, with a 5-pin DIN back-panel connector;
- mouse-interface hardware, firmware in "slot 4" addresses, and a DB-9 back-panel connector shared with game input;
- an enhanced set of disk controller card hardware and firmware, designed to run the built-in drive as Slot 6 Drive 1 (and its equivalents in other operating systems), and the external drive as Slot 6 Drive 2, or even as Slot 7 Drive 1 (PR#7) for system startup from the external drive.

These equivalents of slot-card-firmware-connector are called ports 1, 2, 4, 6 and 7, respectively. By extension, the 80-column video firmware can be called port 3, but only with caution. The Apple IIe and IIc Reference Manuals discuss how to turn the 80-column firmware on and off correctly.

Software in General

This section points out differences to watch out for with respect to programming languages and operating systems that can (or can't) run on the four machines.

Languages

The Apple IIc does not support Pascal 1.0 firmware (I/O) protocols, because its required fixed entry points are impossible to match with the new firmware. Pascal 1.1 is more flexible, and so the Apple IIc can and does support it. Here the entry points are addressed indirectly via a jump table.

The Apple IIc as shipped will not support Integer BASIC because that interpreter does not work under ProDOS. To use Integer BASIC, start the system using the *DOS 3.3 System Master* disk, and invoke Integer BASIC from the keyboard or program.

Former cassette I/O locations now belong to the 40/80-column switch ($C060; was cassette input) and firmware functions ($C02x; was cassette output).

Operating Systems

The Apple IIc will be a ProDOS, rather than a DOS, machine. That does not mean that DOS will not run on it. Rather, we will describe ProDOS as the operating system, ship it and not DOS unless otherwise requested.

CP/M will not currently run on the Apple IIc because it requires plugging a Z80 card into a slot. (Slot? What slot?) Some day there may be another way to make CP/M available, but there isn't right now.

Operating system cassette I/O commands will cause error messages or unpredictable weirdness, depending on how fail-safe the OS is.

Hardware Specifics

The specifics of firmware and I/O storage assignments will be presented in the *Apple IIc Reference Manual*. The sections here discuss the use of integrated and hybrid circuits, and the hard-wired I/O locations in the $C0xx address range.

Use of ICs

The IIe custom chips (Memory Management Unit and Input/Output Unit) replaced more than 50 chips, and added the functionality of dozens more. The IIe PAL replaced several logic chips. The Apple IIc has custom MMU and IOU chips, too, but they have differenct "bonding options"; that is, some of the pins are attached to different parts of the logic inside for the IIe and Apple IIc versions.

In addition, the Apple IIc has a custom General Logic Unit (GLU), Timing Generator (TMG), and Disk Controller Unit (IWM, Integrated Woz (or Wendell) Machine). The Apple IIc has two

hybrid units (AUD and VID) for audio and video amplification; these save space on the PC board and consume less power that the seperate components ("discretes") that they replace.

The trend as one moves from Apple II and II Plus to Apple IIe and IIc is toward fewer and fewer chip sockets. Directly soldering ICs to the circuit board saves money and increases reliability. However, certain key parts (like character generator ROMs) still have sockets. The Apple IIc, in fact, is not intended to be opened by the user—only by Apple manufacturing and service—so for most people, sockets/no sockets is not important.

Hardware Locations

The following table compares the functions that have been hard-wired into the Apple IIe and IIc. Those hard-wired into the Apple II and II Plus are explained in the *Apple II Reference Manual*.

		Apple IIe	Apple IIc
C000	KBD	Keyboard data (0-6) & strobe (read)	Same as on IIe
C000	80STORE	Store in main memory (write)	Same as on IIe
C001		Store in aux memory (write)	Same as on IIe
C002	RAMRD	Read main memory (write)	Same as on IIe
C003		Read aux memory (write)	Same as on IIe
C004	RAMWRT	Write main memory (write)	Same as on IIe
C005		Write aux memory (write)	Same as on IIe
C006	SLOTCXROM	Slot ROMs at Cx00 (write)	Reserved (write)
C007		Internal ROM at Cx00 (write)†	Reserved (write)
C008	ALTZP	Main stack & zero page (write)	Same as on IIe
C009		Aux stack & zero page (write)	Same as on IIe
C00A	SLOTC3ROM	Internal ROM at C300 (write)	Reserved (write)
C00B		Slot ROM at C300 (write)	Reserved (write)
C00C	80COL	80-column display off (write)	Same as on IIe
C00D		80-column display on (write)	Same as on IIe
C00E	ALTCHARSET	Alt. char. set off (write)	Same as on IIe
C00F		Alt. char. set on (write)	Same as on IIe
C01x	KBDSTRB	Clear keyboard strobe (write)	Same as on IIe
C010	RDAKD	Any key down (bit 7)	Same as on IIe
C011	RDBANK	Read bank 1,2 (bit 7 = 1 = bank 2)	Same as on IIe
C012	RDRAM	Read RAM protect/enable (C08x)	Same as on IIe
C013	RDRAMRD	Read RAMRD switch (C002, C003)	Same as on IIe
C014	RDRAMWRT	Read RAMWRT switch (C004, C005)	Same as on IIe
C015	RDSLOTCXROM	Read SLOTCXROM switch (C006, C007)	Reset XINT (read)
C016	RDALTZP	Read ALTZP switch (C008, C009)	Same as on IIe
C017	RDSLOTC3ROM	Read SLOTC3ROM switch (C00A, C00B)	Reset YINT (read)
C018	RD80STORE	Read switch (C000, C001)	Same as on IIe
C019	RDVBL	Read vertical blanking (VBL)	Reset VBLINT (read)‡
C01A	RDTEXT	Read TEXT switch (C050, C051)	Same as on IIe
C01B	RDMIXED	Read MIXED switch (C052, C053)	Same as on IIe
C01C	RDPAGE2	Read PAGE2 switch (C054, C055)	Same as on IIe
C01D	RDHIRES	Read HIRES switch (C057, C058)	Same as on IIe
C01E	RDALTCHARSET	Read ALTCHARSET switch (C00E, C00F)	Same as on IIe
C01F	RD80COL	Read 80COL switch (C00C, C00D)	Same as on IIe
C020			
C021			
C022			
C023			
C024			
C025			
C026			Same as on IIe
C027		Toggle cassette output (read only)	
C028			Reserved (write)
C029			
C02A			
C02B			
C02C			
C02D			
C02E			
C02F			

† This would be more appropriately called INTCXROM

‡ Use $C07x to reset VBLINT and also trigger paddle timers

	Apple IIe	Apple IIc
C030	Toggle speaker (read only)	Same as on IIe
C031		
C032		
C033		
C034		
C035		
C036		
C037		Reserved (write)
C038		
C039		
C03A		
C03B		
C03C		
C03D		
C03E		
C03F		
C040	Read annunciator 0 (bit 7)	Read X0/Y0 mask status (bit 7)
C041	Read annunciator 1 (bit 7)	Read VBL mask status (bit 7)
C042	Read annunciator 2 (bit 7)	Read X0 edge (1 = falling)
C043	Read annunciator 3 (bit 7)	Read Y0 edge (1 = falling)
C044		Reserved
C045		Reserved
C046		Reserved
C047		Reserved
C048		Read or write resets XINT & YINT
C049		Read or write resets XINT & YINT
C04A		Read or write resets XINT & YINT
C04B		Read or write resets XINT & YINT
C04C		Read or write resets XINT & YINT
C04D		Read or write resets XINT & YINT
C04E		Read or write resets XINT & YINT
C04F		Read or write resets XINT & YINT
C050	Text mode off	Same as on IIe
C051	Text mode on	Same as on IIe
C052	Mixed mode off (if text mode off)	Same as on IIe
C053	Mixed mode on (if text mode off)	Same as on IIe
C054	Page 2 off (depends on 80STORE)	Same as on IIe
C055	Page 2 on (depends on 80STORE)	Same as on IIe
C056	Hi-res clear: use RAMRD and RAMWRT	Same as on IIe
C057	Hi-res set: access hi-res page	Same as on IIe
C058	Annunciator 0 off	Disable mouse X0 & Y0 interrupts†
C059	Annunciator 0 on	Enable mouse X0 & Y0 interrupts†
C05A	Annunciator 1 off	Disable VBL interrupts*
C05B	Annunciator 1 on	Enable VBL interrupts*
C05C	Annunciator 2 off	Interrupt on rising edge of X0†
C05D	Annunciator 2 on	Interrupt on falling edge of X0†
C05E	Annunciator 3 off; set dbl-hi-res	If IOUDIS off: interrupt on rising edge of Y0 If IOUDIS on: set dbl-hi-res
C05F	Annunciator 3 on; clear dbl-hi-res	If IOUDIS off: interrupt on falling edge of Y0 If IOUDIS on: clear dbl-hi-res

† IOUDIS must be off for all these to work; all are R/W reserved if IOUDIS on.

	Apple IIe	Apple IIc
C06x		Reserved (write)
C060	Cassette in (read)	Read 80/40 colume switch (bit 7) (1 = 40 Col {switch down})
C061	Switch input 0 & d key	Same as on IIe (bit 7 = 1 = pressed)
C062	Switch input 1 & CLOSED-APPLE key	Same as on IIe (bit 7 = 1 = pressed)
C063	Switch input 2 (read)†	Read mouse switch (bit 7)
C064	Read analog input 0 (bit 7)	Same as on IIe
C065	Read analog input 1 (bit 7)	Same as on IIe
C066	Read analog input 2 (bit 7)	Read mouse X1 (direction) on bit 7
C067	Read analog input 3 (bit 7)	Read mouse Y1 (direction) on bit 7
C068		Reserved (read)
C069		Reserved (read)
C06A		Reserved (read)
C06B		Reserved (read)
C06C		Reserved (read)
C06D		Reserved (read)
C06E		Reserved (read)
C06F		Reserved (read)
C07x	Analog input reset (paddle trigger)	
C070		Read or write: trigger paddle timer; reset VBLINT
C071		Reserved
C072		Reserved
C073		Reserved
C074		Reserved
C075		Reserved
C076		Reserved
C077		Read: bit 7 = GR (1 = current line is graphics; 0 = it is text)
C078		Reserved
C079		Reserved
C07A		Reserved
C07B		Reserved
C07C		Reserved
C07D		Reserved
C07E		Read: bit 7 = IOUDIS; trigger paddle timer; reset VBLINT Write: set IOUDIS (that is, disable C058-F IOU access & enable DHIRES switch); trigger paddle timer; reset VBLINT
C07F		Read: bit 7 = DHIRES; trigger paddle timer; reset VBLINT Write: clear IOUDIS (that is, enable C058-F IOU access & disable DHIRES switch); trigger paddle timer; reset VBLINT

† Commonly used as shift-key mod on II/II Plus

	Apple IIe	Apple IIc	
C080	Protect RAM Read RAM 2nd D000 Bank	Same as on IIe	
C081	Write RAM Read ROM 2nd D000 Bank†	Same as on IIe	
C082	Protect RAM Read ROM 2nd D000 Bank	Same as on IIe	
C083	Write RAM Read RAM 2nd D000 Bank†	Same as on IIe	
C084	Protect RAM Read RAM 2nd D000 Bank	Reserved	
C085	Write RAM Read ROM 2nd D000 Bank†	Reserved	
C086	Protect RAM Read ROM 2nd D000 Bank	Reserved	
C087	Write RAM Read RAM 2nd D000 Bank†	Reserved	
C088	Protect RAM Read RAM 1st D000 Bank	Same as on IIe	
C089	Write RAM Read ROM 1st D000 Bank†	Same as on IIe	
C08A	Protect RAM Read ROM 1st D000 Bank	Same as on IIe	
C08B	Write RAM Read RAM 1st D000 Bank†	Same as on IIe	
C08C	Protect RAM Read RAM 1st D000 Bank	Reserved	
C08D	Write Ram Read ROM 1st D000 Bank†	Reserved	
C08E	Protect RAM Read ROM 1st D000 Bank	Reserved	
C08F	Write RAM Read RAM 1st D000 Bank†	Reserved	
C090		Reserved (Serial port 1)	
C091		Reserved	
C092		Reserved	
C093		Reserved	
C094		Reserved	
C095		Reserved	
C096		Reserved	
C097		Reserved	
C098	Slot 1 perphieral card I/O	Transmit/Receive Reg	ACIA
C099		Status Register	
C09A		Command Register	
C09B		Control Register	
C09C		Reserved	
C09D		Reserved	
C09E		Reserved	
C09F		Reserved	
C0A0		Reserved (Serial port 2)	
C0A1		Reserved	
C0A2		Reserved	
C0A3		Reserved	
C0A4		Reserved	
C0A5		Reserved	
C0A6		Reserved	
C0A7		Reserved	
C0A8	Slot 2 perphieral card I/O	Transmit/Receive Reg	ACIA
C0A9		Status Register	
C0AA		Command Register	
C0AB		Control Register	
C0AC		Reserved	
C0AD		Reserved	
C0AE		Reserved	
C0AF		Reserved	

† Write RAM requires 2 consecutive read accesses; protect RAM does not.

	Apple IIe	Apple IIc
C0B0		Reserved
C0B1		Reserved
C0B2		Reserved
C0B3		Reserved
C0B4		Reserved
C0B5		Reserved
C0B6		Reserved
C0B7		Reserved
C0B8	Slot 3 perphieral card I/O	Reserved
C0B9		Reserved
C0BA		Reserved
C0BB		Reserved
C0BC		Reserved
C0BD		Reserved
C0BE		Reserved
C0BF		Reserved
C0C0		Reserved
C0C1		Reserved
C0C2		Reserved
C0C3		Reserved
C0C4		Reserved
C0C5		Reserved
C0C6		Reserved
C0C8	Slot 4 perphieral card I/O	Reserved
C0C9		Reserved
C0CA		Reserved
C0CB		Reserved
C0CC		Reserved
C0CD		Reserved
C0CE		Reserved
C0CF		Reserved
C0D0		Reserved
C0D1		Reserved
C0D2		Reserved
C0D3		Reserved
C0D4		Reserved
C0D5		Reserved
C0D6		Reserved
C0D7		Reserved
C0D8	Slot 5 perphieral card I/O	Reserved
C0D9		Reserved
C0DA		Reserved
C0DB		Reserved
C0DC		Reserved
C0DD		Reserved
C0DE		Reserved
C0DF		Reserved

	Apple IIe	Apple IIc
C0E0		Phase 0 = 0 (Disk Controller)
C0E1		Phase 0 = 1
C0E2		Phase 1 = 0
C0E3		Phase 1 = 1
C0E4		Phase 2 = 0
C0E5		Phase 2 = 1
C0E6		Phase 3 = 0
C0E7		Phase 3 = 1
C0E8	Slot 6 perphieral card I/O	Motor off
C0E9		Motor on
C0EA		Drive 1
C0EB		Drive 2
C0EC		L6 = 0
C0ED		L6 = 1
C0EE		L7 = 0
C0EF		L7 = 1
C0F0		Reserved
C0F1		Reserved
C0F2		Reserved
C0F3		Reserved
C0F4		Reserved
C0F5		Reserved
C0F6		Reserved
C0F7		Reserved
C0F8	Slot 7 perphieral card I/O	Reserved
C0F9		Reserved
C0FA		Reserved
C0FB		Reserved
C0FC		Reserved
C0FD		Reserved
C0FE		Reserved
C0FF		Reserved

Monitor Entry Point Labels

This section presents a complete compilation of all $F800 Monitor ROM label occurrences in the various source file listings and the various lists of built-in subroutines. An "X" indicates that the label appears in the source code listing and a "supported" indicates it was found in the list of built-in subroutines.

Sources for this information were:

- *Apple II Reference Manual*

  Page 61 - Some Useful Monitor Subroutines
  Page 155 - Monitor ROM Listing
  Page 136 - Autostart ROM Listing

- *Apple IIe Reference Manual*

  Appendix C - Directory of Built-in Subroutines

- *Apple II Reference Manual Addendum:Monitor ROM Listings*

  Page 3 - Monitor Firmware Listing

- *Apple IIc Reference Manual*

  Appendix C - Important Firmware Locations
  C.5 Monitor Addresses

- *Apple IIc Firmware Assembly List*

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$F800	PLOT	X supported	X supported	X	X supported
$F80C	RTMASK	X	X	X	X
$F80E	PLOT1	X	X	X	X
$F819	HLINE	X supported	X supported	X supported	X supported
$F81C	HLINE1	X	X	X	X
$F826	VLINEZ	X	X	X	X
$F828	VLINE	X supported	X supported	X supported	X supported
$F831	RTS1	X	X	X	X
$F832	CLRSCR	X supported	X supported	X supported	X supported
$F836	CLRTOP	X supported	X supported	X supporter	X supported
$F838	CLRSC2	X	X	X	X
$F83C	CLRSC3	X	X	X	X
$F847	GBASCALC	X	X	X	X
$F856	GBCALC	X	X	X	
$F85F	NEXTCOL	supported	supported	supported	
$F85F	NXTCOL	X	X	X	
$F864	SETCOL	X supported	X supported	X supported	X supported
$F871	SCRN	X supported	X supported	X supported	X supported
$F879	SCRN2	X	X	X	X
$F87F	RTMSKZ	X	X	X	X
$F882	INSDS1	X	X	X	X
$F88C	INSDS2				X
$F88E	INSDS2	X	X	X	
$F897	IEVEN				X
$F89B	IEVEN	X	X	X	
$F8A1	ERR				X
$F8A5	ERR	X	X	X	
$F8A5	GETFMT				X
$F8A9	GETFMT	X	X	X	
$F8BE	MNNDX1	X	X	X	X
$F8C2	MNNDX2	X	X	X	X
$F8C9	MNNDX3	X	X	X	X
$F8CD	GOTONE				X
$F8D0	INSTDSP	X	X	X	X
$F8D4	PRNTOP	X	X	X	X
$F8DB	PRNTBL	X	X	X	X
$F8F5	NXTCOL		X		
$F8F5	PRMN1	X		X	X
$F8F9	PRMN2	X	X	X	X
$F910	PRADR1	X	X	X	X
$F914	PRADR2	X	X	X	X
$F926	PRADR3	X	X	X	X
$F92A	PRADR4	X	X	X	X
$F930	PRADR5	X	X	X	X
$F938	RELADR	X	X	X	X
$F940	PRNTYX	X	X	X	X
$F941	PRNTAX	X supported	X supported	X supported	X supported
$F944	PRNTX	X	X	X	X
$F948	PRBLNK	X supported	X supported	X supported	X
$F94A	PRBL2	X supported	X supported	X supported	X supported

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$F94C	PRBL3	X	X	X	X
$F953	PCADJ	X	X	X	X
$F954	PCADJ2	X	X	X	X
$F956	PCADJ3	X	X	X	X
$F95C	PCADJ4	X	X	X	X
$F961	RTS2	X	X	X	X
$F962	FMT1	X	X	X	X
$F9A6	FMT2	X	X	X	X
$F9B4	CHAR2				X
$F9B4	CHAR1	X	X	X	
$F9BA	CHAR1				X
$F9BA	CHAR2	X	X	X	
$F9C0	MNEML	X	X	X	X
$FA00	MNEMR	X	X	X	X
$FA40	IRQ		X	X	X
$FA47	NEWBREAK				X
$FA43	STEP	X			
$FA4C	BREAK		X	X	X
$FA4E	XQINIT	X			
$FA59	OLDBRK		X	X	X
$FA62	RESET		X	X	X
$FA6F	INITAN		X	X	
$FA78	XQ1	X			
$FA7A	XQ2	X			
$FA81	NEWMON		X	X	X
$FA86	IRQ	X			
$FA92	BREAK	X			
$FA9B	FIXSEV		X	X	X
$FA9C	XBRK	X			
$FAA3	BEEPFIX				X
$FAA3	NOFIX		X	X	X
$FAA5	XRTI	X			
$FAA6	PWRUP		X	X	X
$FAA6	SETPG3		X	X	X
$FAA9	XRTS	X			
$FAAB	SETPLP		X	X	X
$FAAD	PCINC2	X			
$FAAF	PCINC3	X			
$FAB9	XJSR	X			
$FABA	SLOOP		X	X	
$FABD	RESET.X				
$FAC4	XJMP	X			
$FAC5	XJMPAT	X			
$FAC7	NXTBYT		X	X	
$FACD	NEWPCL	X			
$FACF	NOFIX				X
$FAD1	RTNJMP	X			
$FAD2	RTBL				X
$FAD7	REGDSP	X	X	X	X
$FADA	RGDSP1	X	X	X	X

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$FAE4	RDSP1	X	X	X	X
$FAFD	PWRCON		X	X	X
$FAFD	BRANCH	X			
$FB02	RGDSP2				X
$FB05	DISKID		X	X	
$FB09	TITLE		X	X	X
$FB0B	NBRNCH	X			
$FB11	XLTBL		X	X	
$FB11	INITBL	X			
$FB12	PWRUP2				X
$FB19	RTBL	X	X	X	
$FB1E	PREAD	X supported	X supported	X supported	X supported
$FB25	PREAD2	X	X	X	X
$FB2E	RTS2D	X	X	X	X
$FB2F	INIT	X	X	X	X
$FB39	SETTXT	X	X	X	X
$FB40	SETGR	X	X	X	X
$FB4B	SETWND	X	X	X	X
$FB59	VTAB23				X
$FB5B	TABV	X	X	X	X
$FB60	APPLEII		X	X	X
$FB60	MULPM	X			
$FB63	MUL	X			
$FB65	STITLE		X	X	X
$FB65	MUL2	X			
$FB6D	MUL3	X			
$FB6F	SETPWRC		X	X	X supported
$FB76	MUL4	X			
$FB78	VIDWAIT		X	X	X
$FB78	MUL5	X			
$FB81	DIVPM	X			
$FB84	DIV	X			
$FB86	DIV2	X			
$FB88	KBDWAIT		X	X	X
$FB94	NOWAIT		X	X	X
$FB97	ESCOLD		X	X	
$FB9B	ESCNOW		X	X	
$FBA0	NEWADV				X
$FBA0	DIV3	X			
$FBA4	MD1	X			
$FBA5	ESCNEW		X	X	
$FBAF	MD2	X			
$FBB0	NEWADV1				X
$FBB3	F8VERSION				X
$FBB3	VERSION			X	
$FBB4	GOTOCX			X	
$FBB4	DOCOUT1				X
$FBB4	MD3	X			
$FBBC	DCX				X
$FBC0	MDRTS	X			

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$FBC1	BASCALC	X	X	X	X
$FBD0	BSCLC2	X	X	X	X
$FBD9	CHKBELL				
$FBD9	BELL1	X	X	X	
$FBDD	BELL1	supported	supported	supported	X supported
$FBE4	BELL2	X	X	X	X
$FBEF	RTS2B	X	X	X	X
$FBF0	STORADV		X	X	X
$FBF0	STOADV	X			
$FBF4	ADVANCE	X	X	X	X
$FBF8	ADV2				X
$FBFC	RTS3	X	X	X	X
$FBFD	VIDOUT	X	X	X	X
$FC04	VIDOUT1				X
$FC10	BS	X	X	X	X
$FC1A	UP	X	X	X	X
$FC22	VTAB	X	X	X	X
$FC24	VTABZ	X	X	X	X
$FC2B	RTS4	X	X	X	X
$FC2C	ESC1	X	X	X	
$FC35	NEWOPS				X
$FC38	NEWOP1				X
$FC42	CLREOP	X	X	X supported	X supported
$FC44	CLREOP2				X
$FC46	CLEOP1	X	X	X	X
$FC58	HOME	X	X	X supported	X supported
$FC5D	CLREOP1				X
$FC62	CR	X	X	X	X
$FC66	LF	X	X	X	X
$FC70	SCROLL	X	X	X	X
$FC72	XGOTOCX			X	
$FC73	NEWCR				X
$FC76	SCRL1	X	X		
$FC80	GETINDX				X
$FC84	RDCX			X	
$FC85	CRRTS				X
$FC86	NEWVTAB				X
$FC8C	SCRL2	X	X		
$FC8D	NEWCLREOL				X
$FC90	NEWCLEOLZ				X
$FC91	ISSLOTS			X	
$FC95	SCRL3	X	X		
$FC99	NEWC1				X
$FC99	ISPAGE1			X	
$FC9C	CLREOL	X	X	X supported	X supported
$FC9E	CLEOLZ	X	X	X supported	X supported
$FCA0	CLRLIN				X
$FCA0	CLEOL2	X	X		
$FCA4	CTLDO				X
$FCA8	WAIT	X supported	X supported	X supported	X supported

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$FCA9	WAIT2	X	X	X	X
$FCAA	WAIT3	X	X	X	X
$FCB4	NXTA4	X	X	X	X
$FCBA	NXTA1	X	X	X	X
$FCC8	RTS4B	X	X	X	X
$FCC9	HEADR	X	X	X	
$FCCA	COLDSTART				X
$FCD0	BLAST				X
$FCD6	WRBIT	X	X	X	
$FCDB	ZERDLY	X	X	X	
$FCE2	ONEDLY	X	X	X	
$FCE5	WRTAPE	X	X	X	
$FCE7	COM1				X
$FCEC	RDBYTE	X	X	X	
$FCEE	RDBYT2	X	X	X	
$FCF6	COM2				X
$FCFA	RD2BIT	X	X	X	
$FCFC	COM3				X
$FCFD	RDBIT	X	X	X	
$FD03	APPLE2C				X
$FD0C	RDKEY	X supported	X supported	X supported	X supported
$FD18	KEYIN0				X
$FD1B	KEYIN	X supported	X supported	X supported	X supported
$FD20	DONXTCUR				X
$FD21	RDESC			X	
$FD21	KEYIN2	X	~~X~~		
$FD25	GOTKEY				X
$FD2F	ESC	X	X	X	
$FD35	RDCHAR	X supported	X supported	X supported	X supported
$FD38	LOOKPICK				X
$FD3D	NOTCR	X	X	X	
$FD44	NOESCAPE				X
$FD45	NOESC1				X
$FD4A	NOESC2				X
$FD5F	NOTCR1	X	X	X	X
$FD62	CANCEL	X	X	X	X
$FD67	GETLNZ	X supported	X supported	X supported	X supported
$FD6A	GETLN	X supported	X supported	X supported	X supported
$FD6F	GETLN1	supported	supported	supported	X supported
$FD71	BCKSPC	X	X	X	X
$FD75	NXTCHAR	X	X	X	X
$FD7E	CAPTST	X	X	X	
$FD84	ADDINP	X	X	X	X
$FD8B	CROUT1	supported	supported	supported	X supported
$FD8E	CROUT	X supported	X supported	X supported	X supported
$FD92	PRA1	X	X	X	X
$FD96	PRYX2	X	X	X	X
$FDA3	XAM8	X	X	X	X
$FDAD	MOD8CHK	X	X	X	X
$FDB3	XAM	X	X	X	X

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$FDB6	DATAOUT	X	X	X	X
$FDC5	RTS4C	X	X	X	X
$FDC6	XAMPM	X	X	X	X
$FDD1	ADD	X	X	X	X
$FDDA	PRBYTE	X supported	X supported	X supported	X supported
$FDE3	PRHEX	X supported	X supported	X supported	X supported
$FDE5	PRHEXZ	X	X	X	X
$FDED	COUT	X supported	X supported	X supported	X supported
$FDF0	COUT1	X supproted	X supported	X supported	X supported
$FDF6	COUTZ	X	X	X	X
$FE00	BL1	X	X	X	X
$FE04	BLANK	X	X	X	X
$FE0B	STOR	X	X	X	X
$FE17	RTS5	X	X	X	X
$FE18	SETMODE	X	X	X	X
$FE1D	SETMDZ	X	X	X	X
$FE20	LT	X	X	X	X
$FE22	LT2	X	X	X	X
$FE2C	MOVE	X	X	X supported	X supported
$FE36	VERIFY			supported	X supported
$FE36	VFY	X	X	X	
$FE58	VFYOK	X	X	X	X
$FE5E	LIST	X	X	X	X
$FE63	LIST2	X	X	X	X
$FE75	A1PC	X	X	X	X
$FE78	A1PCLP	X	X	X	X
$FE7F	A1PCRTS	X	X	X	X
$FE80	SETINV	X supported	X supported	X supported	X
$FE84	SETNORM	X supported	X supported	X supported	X
$FE86	SETIFLG	X	X	X	X
$FE89	SETKBD	X	X	X	X
$FE8B	INPORT	X	X	X	X
$FE8D	INPRT	X	X	X	X
$FE93	SETVID	X	X	X	X
$FE95	OUTPORT	X	X	X	X
$FE97	OUTPRT	X	X	X	X
$FE9B	IOPRT	X	X	X	X
$FEA7	NOTPRT0				X
$FEA7	IOPRT1	X	X	X	
$FEA9	IOPRT2	X	X	X	
$FEAB	IOPRT2				X
$FEAF	CKSUMFIX			X	
$FEB0	XBASIC	X	X	X	X
$FEB3	BASCONT	X	X	X	X
$FEB6	GO	X	X	X	X
$FEBF	REGZ	X	X	X	X
$FEC2	OPRT2				X
$FEC2	TRACE	X	X	X	
$FEC4	STEPZ	X	X	X	
$FECA	USR	X	X	X	X

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Address	Label	Apple II	Apple II Plus	Apple IIe	Apple IIc
$FECD	WRITE	X	X	X supported	X
$FECE	DOPR0				X
$FED4	WR1	X	X	X	
$FEDE	IOPRT1				X
$FEE2	DECCH				X
$FEE9	CLRCH				X
$FEEB	WDTHCH				X
$FEEC	SETCUR				X
$FEED	WRBYTE	X	X	X	
$FEEE	SETCUR1				X
$FEEF	WRBYT2	X	X	X	
$FEF6	CRMON	X	X	X	X
$FEFD	READ	X	X	X supported	X
$FEFE	OPTBL				X
$FF0A	RD2	X	X	X	
$FF15	INDX				X
$FF16	RD3	X	X	X	
$FF2D	PRERR	X supported	X supported	X supported	X supported
$FF3A	BELL	X supported	X supported	X supported	X supported
$FF3F	IOREST	supported	supported	supported	supported
$FF3F	RESTORE	X	X	X	X
$FF44	RESTR1	X	X	X	X
$FF4A	IOSAVE	supported	supported	supported	supported
$FF4A	SAVE	X	X	X	X
$FF4C	SAV1	X	X	X	X
$FF58	IORTS			supported	
$FF59	OLDRTS		X	X	X
$FF59	RESET	X			
$FF65	MON	X	X	X	X
$FF69	MONZ	X	X	X	X supported
$FF73	NXTITM	X	X	X	X
$FF7A	CHRSRCH	X	X	X	X
$FF8A	DIG	X	X	X	X
$FF90	NXTBIT	X	X	X	X
$FF98	NXTBAS	X	X	X	X
$FFA2	NXTBS2	X	X	X	X
$FFA7	GETNUM	X	X	X	X
$FFAD	NXTCHR	X	X	X	X
$FFBE	TOSUB	X	X	X	X
$FFC7	ZMODE	X	X	X	X
$FFCC	CHRTBL	X	X	X	
$FFCD	CHRTBL				X
$FFE0	SUBTBL				X
$FFE3	SUBTBL	X	X	X	
$FFF7	GETHEX				X
$FFFE	IRQVECT				X

"X" = Label appears in source listing
"supported" = Documented as a supported built-in subroutine

Machine Identification

By looking at the identification bytes in the Monitor ROM, it is possible to identify which machine your software is running on so that it can take advantage of the special features of that particular machine.

The original Apple II and Apple II Plus used two different monitor ROMs: the "original" monitor and the "auto-start" monitor. They are interchangeable between the two machines. In almost every case, it makes no difference whether your software is running on an Apple II or an Apple II Plus, since the hardware was identical, only the Monitor and BASIC ROM sets were changed. This section explains how to determine which Monitor ROM is present, and, if you need to test the BASIC ROMs, you may look at $E000 for a $4C (JMP instruction) to identify an Applesoft ROM set, or a $20 (JSR instruction) to identify an Integer BASIC ROM set.

All other revisions of the Apple II have Applesoft BASIC built in. Note, however, that the Apple III has an Apple II emulation mode, which permits it to emulate a 48K Apple II Plus with either Applesoft or Integer BASIC.

To identify the various Monitor ROMs, look for the following:

Machine	**$FBB3 (64435)**	**$FB1E (64286)**	**$FBC0 (64435)**
Apple II (original monitor)	$38 (56)		
Apple II Plus (autostart monitor)	$EA (234)	$AD (173)	
Apple III emulation mode	$EA (234)	$8A (138)	
Apple IIe	$06 (6)		$EA (234)
Apple IIe with ICON support	$06 (6)		$E0 (224)
Apple IIc	$06 (6)		$00 (0)

Apple's Developer Technical Support group has routines that identify the various versions of the Apple II family. To obtain a copy, write to:

Apple Computer, Inc.
Developer Technical Support
20525 Mariani Ave., MS 22-W
Cupertino, CA 95014

or phone:

(408) 554-5213

Apple IIc Applesoft Firmware Differences

The vectors for the following Applesoft key words:

- SHLOAD
- RESTORE
- STORE
- LOAD
- SAVE

have been changed since they are associated with cassette tape, which is no longer supported. The vectors now point to the ampersand vector so that you can write routines to intercept control when any of these key words appear. If you simply leave the ampersand vector as it is at boot-up, the commands are not rejected with a SYNTAX ERROR, but become "do-nothing" commands.

Under DOS 3.3, hook your routine directly into the ampersand hook at $3F5.

Under ProDOS, $3F5 points to the external command vector in the BASIC.SYSTEM global page. You can hook your routine into the ampersand vector, or into the external command vector in the global page.

In either case, the pointing to the ampersand and/or external command vectors is automatic. No ampersand prefix or "PRINT CONTROL-D" prefix is needed.

Since the Apple IIc has a true uppercase/lowercase keyboard, Applesoft on the Apple IIc will accept and upshift lowercase characters when input in immediate mode. No upshifting will occur

inside of quotes, REMs, DATA statements, or while a BASIC program is executing. All keywords and variable names will be uppercase only when the program is listed.

The Apple IIc firmware supports MouseText. The video firmware, when properly enabled, is able to display a set of graphic characters that were designed to be used with the mouse. To use the mouse characters:

- Turn on the video firmware (PR#3)
- Enable mouse characters (PRINT CHR$(27) {Hex $1B})
- Set inverse mode (INVERSE or PRINT CHR$(15) {Hex $0F})
- Print capital letters or PRINT CHR$(64 to 95)
- Disable mouse characters (PRINT CHR$(24) {Hex $18})
- Set normal mode (PRINT CHR$(14) {Hex $0E})

When actually in screen memory, the 32 mouse characters have ASCII codes 64 - 95 ($40 - $5F). Inverse characters that previously occupied that range are remapped to ASCII codes 0 - 31 ($00 = $1F).

Interrupt Handling on the Apple IIc

This document contains excerpts from the *Apple IIc Reference Manual* and describes the handling of IRQ interrupts. It is intended as an overview of the interrupt capabilities of the Apple IIc. It is not intended as a programmer's guide. The full details are in the *Apple IIc Reference Manual*.

What Is an Interrupt?

On a computer, an interrupt is a signal that tells the computer to stop what it is currently doing and devote its attention to a more important task. For example, the Apple IIc mouse sends an interrupt to the computer every time it moves. This is necessary because, unless the mouse is read shortly after it moves, the signal indicating its direction is lost.

Interrupts on the Apple IIc Computer

The Apple IIc built-in interrupt handler, unlike earlier systems in the Apple II family, now saves the accumulator on the stack instead of in location $45. Thus, both DOS and the Monitor work with interrupts on the Apple IIc.

Interrupts are effective only if they are enabled most of the time. Interrupts that occur while interrupts are disabled cannot be detected. Due to the critical timing nature of disk reads and writes, Pascal, DOS, and ProDOS turn off interrupts while performing disk operations. Thus, it is important to remember that while a disk drive is being accessed, all sources of IRQ interrupts are, in effect, turned off.

Interrupt Handling on the 65C02

From the point of view of the 65C02 in the Apple IIc, there are two possible causes of interrupts:

1. If interrupts to the 65C02 are not masked (that is, the CLI instruction has been used), the IRQ line on the microprocessor could be pulled low.
2. The processor executed a break instruction (BRK = opcode $00)

   (NOTE: The NMI line in the Apple IIc is not used, thus an NMI interrupt can never happen.)

These two options cause the 65C02 to save the current program counter and status byte on the stack and then jump to the routine whose address is stored in $FFFE and $FFFF. The sequence performed by the 65C02 is:

- If IRQ, finish executing the current instruction.
- Push high byte of program counter onto stack.
- Push low byte of program counter onto stack.
- Push status byte onto stack.
- Jump to address stored in $FFFE, $FFFF [JMP ($FFFE)].

The Interrupt Vector at $FFFE

In the Apple IIc computer, there are three separate regions of memory that contain address $FFFE: the built-in ROM, the bank-switched memory in main RAM, and the bank-switched memory in auxiliary RAM. The vector at $FFFE in the ROM points to the Apple IIc's built-in interrupt-handling routine. Because the interrupts in the Apple IIc are complex, we recommend that you use it rather than write your own interrupt-handling routine.

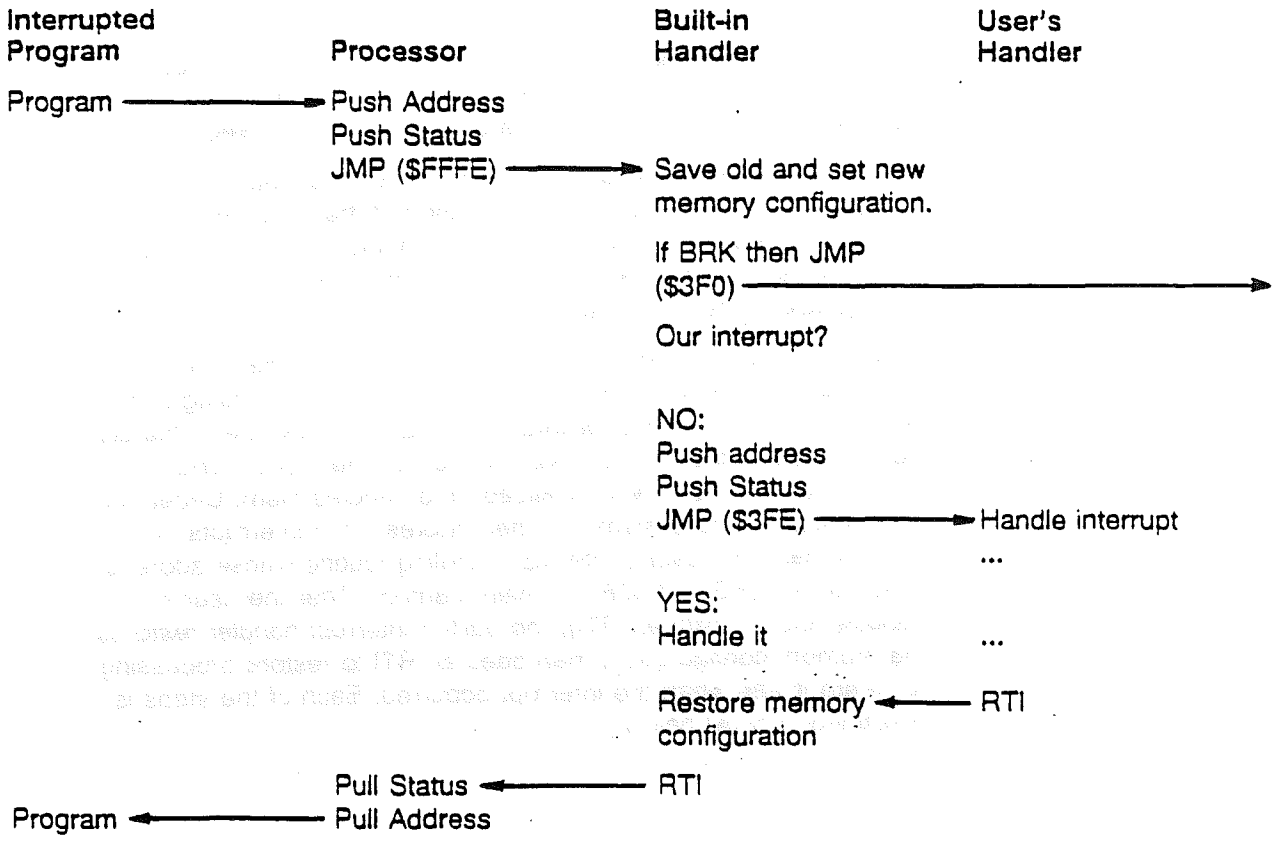
When you initialize the mouse firmware or the communications firmware, copies of the ROM's interrupt vector are placed in the interrupt vector's addresses in both main and auxiliary bank-switched memory. If you plan to use interrupts and the bank-switched memory without the mouse or communications firmware, you must copy the ROM's interrupt vector yourself.

The Built-in Interrupt Handler

The built-in interrupt handler is responsible for determining whether a break or an interrupt occurred. If an interrupt occurred, the built-in handler decides whether the interrupt should be handled internally, handled by the user, or simply ignored.

The built-in interrupt-handling routine records the state of the computer's current memory configuration. It then sets the computer's memory configuration to a standard state. This allows a user's interrupt handler to know the precise memory configuration when it is called.

Next, the built-in interrupt handler checks to see if the interrupt was caused by a break instruction and handles it accordingly. If it was not a break, it looks for interrupts that it knows how to handle (for example, if the interrupt was caused by the mouse, and the mouse has been properly initialized) and handles them. Depending on the state of the system, it either ignores other interrupts or passes them to a user's interrupt-handling routine whose address is stored at $3FE and $3FF of main memory. After the user's handler returns (with an RTI), the built-in interrupt handler restores the memory configuration, then does an RTI to restore processing to where it was when the interrupt occurred. Each of the steps is explained in detail below.



Saving the Memory Configuration

The built-in interrupt handler saves the state of the system and sets it to a known state according to these rules:

- If 80STORE and PAGE2 are on, then text page 1 is switched in so that main screen holes are accessible (PAGE2 off).
- Main memory is switched in for reading (RAMRD off).
- Main memory is switched in for writing (RAMWRT off).
- $D000-$FFF ROM is switched in for reading (RDLCRAM off).

- Main stack and zero page are switched in (ALTZP off).
- Auxiliary stack pointer is preserved, and the main stack is restored.

Since main memory is switched in, all memory addresses used later in this section are in main memory unless otherwise specified.

Managing the Memory Configuration

Because the Apple IIc has two stack pages, we have adopted a convention that allows the system to be run with two separate stack pointers. Two bytes in the auxiliary stack page are to be used as storage for inactive stack pointers: $100 for the main stack pointer when the auxiliary stack is active, and $101 for the auxiliary stack pointer when the main stack is active.

When a program uses interrupt switches in the auxiliary stack for the first time, it should place the value of the main stack pointer at $100 in the auxiliary stack and initialize the auxiliary stack pointer to $FF (the top of the stack). When it subsequently switches from one stack to the other, it should save the current stack pointer before loading the pointer for the other stack.

User's Interrupt Handler at $3FE

The screen hole locations can be set up to indicate that the user's interrupt handler should be called when certain interrupts occur. To use such a routine, place the address of the routine at $3FE and $3FF in main memory (low byte first).

The user's interrupt handler should

- verify that the interrupt came from the expected source;
- handle the interrupt as desired;
- ccear the interrupt, if necessary;
- return with an RTI.

In general, there is no guaranteed response time for interrupts because the system may be doing a disk operation that could last for several seconds.

Once the built-in interrupt handler has been called, it takes about 250 to 300 microseconds for it to call your interrupt handling routine. After you routine returns, it takes 40 to 140 microseconds to restore memory and return to the interrupted program.

Sources of Interrupts

The Apple IIc can receive interrupts from many different sources. Each source is enabled and used slightly differently than the others. There are two basic classes of interrupt sources: those associated with use of the mouse, and those associated with the two 6551 ACIA circuits.

The interrupts associated with the mouse are

- an interrupt generated when the mouse is moved in the horizontal (X) direction
- an interrupt generated when the mouse is moved in the verticle (Y) direction
- an interrupt generated every 1/60 second, synchronized with the video vertical blanking signal
- using the firmware, an interrupt generated when the mouse button is pressed.

The interrupts associated with the ACIA's are

- an interrupt generated when a key is pressed
- an interrupt generated by a device attached to the external disk drive port
- an interrupt generated when either ACIA has received a byte of data from its port
- an interrupt generated when pin 5 of either serial port changed state
- an interrupt generated when either ACIA is ready to accept another character to be transmitted
- an interrupt generated when the keyboard strobe is cleared.

Firmware-Handling of Interrupts

The following sections present an overview of how the built-in firmware handles interrupts.

Firmware for Mouse and Vertical Blanking

When the mouse is initialized, the interrupt vector is copied to main and auxiliary bank-switched RAM. When the mouse is active, possible sources of interrupts are

- mouse movement in the X direction
- mouse movement in the Y direction
- change of state of the button
- leading edge of the vertical blanking signal.

When an interrupt occurs, the built-in interrupt handler determines whether that particular interrupt source was enabled by the SETMOUSE call. If so, the user's interrupt handler, whose address is stored at $3FE, is called.

The interrupt handler should first call SERVEMOUSE to determine the source of the interrupt. If the interrupt was due to mouse movement or button, the interrupt handler should then do a call to READMOUSE. The interrupt should then be serviced and terminated with an RTI.

Remember: An interrupt may be missed during disk accesses.

If you turn on mouse interrupts without initializing the mouse, the built-in interrupt handler will absorb the interrupts. If you wish to handle mouse interrupts yourself, you must write your own interrupt handler and place vectors to it in bank-switched RAM. Interrupts will be ignored whenever the $D000-$FFFF ROM is switched in.

Firmware for Keyboard Interrupts

The Apple IIc is able to generate an interrupt when a key is pressed. Keyboard interrupts are received through the ACIA for port 2. When the user's interrupt handler is called, it can identify the interrupt source as the keyboard rather than the serial port.

The firmware is able to buffer up to 128 keystrokes. After the buffer is full, any additional keystrokes are ignored. Because interrupts are generated only when a key is pressed; auto-repeated characters are not buffered.

Once keyboard buffering has been turned on, the next key should be read by calling RDKEY ($FD0C). Pressing ⌘-CONTROL-X clears the buffer.

Keyboard buffering is automatically turned on when the serial firmware is placed in Terminal mode. Otherwise, you must turn it on yourself. A PR#2 or IN#2 or the equivalent will shut off keyboard buffering.

Using External Interrupts Through Firmware

Pin 9 of the external disk drive connector (EXTINT) can be used to generate interrupts through the ACIA for port 1. It can be used as a source of interrupts (on a high-to-low transition) if enabled.

When the user's interrupt handler is called, it can identify the source of the interrupt.

Firmware for Serial Interrupts

The Apple IIc is able to generate interrupts both when the ACIA received data and when it is ready to send data. The built-in interrupt handler responds to incoming data only. The firmware is able to buffer up to 128 incoming bytes of serial data from either serial port. After the buffer is full, data are ignored. Only one port can be buffered at a time.

Serial buffering is automatically turned on when serial firmware is placed in Terminal mode. Otherwise, you must turn it on yourself. When enabled, normal reads from the serial port firmware fetch data from the buffer rather than directly from the ACIA.

It is also possible to use the firmware to call the user interrupt handler whenever a byte of data is read by the ACIA. In this mode, buffering is not performed by the firmware. When thus enabled, the user's interrupt handler is called each time the port receives a byte of data. The handler can identify the source of the interrupt.

The serial firmware does not implement buffering for serial output. Instead, it waits for two conditions to be true before transmitting a character:

- The ACIA's transmit register must be ready to accept a character.
- The device must signal that it is ready to accept data.

A Loophole in the Firmware

So that programs can make use of interrupts on the ACIAs without affecting mouse interrupt handling, we left a time loophole in the built-in handler. If transmit interrupts are enabled on the ACIA, then control is passed to the user's interrupt handler if the interrupt is not intended for the mouse (movement, button, or VBL).

This means that you can write more sophisticated serial interrupt-handling routines than we could provide (such as printer spooling). The firmware will still set memory to its standard state, handle mouse interrupts, and restore memory after your routine is finished.

When you receive the interrupt, neither ACIA's status register has been read. It is your responsibility to check for interrupts on both ACIAs. You must determine which of the four interrupt sources on each ACIA caused the interrupt and how to handle them. The built-in firmware itself is an excellent example of how interrupts on the ACIA can be handled.

Apple IIc Firmware

This section is a brief user's guide to the firmware of the Apple IIc. It assumes that you are familiar with the use and operation of the Apple IIe, and it places emphasis on the differences between the IIe and IIc.

Video Firmware

40 Columns Versus 80 Columns

The Apple IIe has two distinct video modes: Apple II mode (checkerboard cursor) and Apple IIe mode (solid cursor). The system boots up in Apple II mode; you switch to Apple IIe mode with the PR#3 commmand and return to Apple II mode using ESC CONTROL-Q. On the Apple IIc, the commands ESC 4 and ESC 8 will also switch into Apple IIe mode.

Diagnostics

The Apple IIc does not have a diagnostic program as we know it in the Apple IIe. Instead, it has a memory exerciser that exercises all the RAM and I/O switches. To activate it, press

-CONTROL-RESET

To reboot the system, press

CONTROL-RESET

65C02 Microprocessor

The Apple IIc uses the 65C02 microprocessor, an extended version of the 6502 chip used in the IIe. If you use the Monitor program in the Apple IIc, you will find that the L command (List) disassembles the extended instruction set provided by the 65C02. (The ProDOS version of EDASM supports this extended instruction set if you use the X6502 directive.)

Window Widths

The Apple IIe video firmware allows only even window widths and window left edges when you are using 80-column mode. The Apple IIc video firmware allows you to use both odd and even window widths in all situations.

Mouse Firmware

Mouse Character Set

The Apple IIc is endowed with the world-famous mouse character set. The Apple IIc character ROM, when properly enabled, is able to display a set of graphics characters that were designed to be used with the mouse. To use the mouse characters

- Turn on the video firmware (use the PR#3 command).
- Enable mouse characters (PRINT CHR$(27) (hex $1B)).
- Set inverse mode.
- Print capital letters.
- Disable mouse characters (PRINT CHR$(24) (hex $18)).
- Set normal mode.

The mouse character set itself is included at the end of this document. Here is a BASIC program that prints all the mouse characters:

```
10  D$=CHR$(4)
20  PRINT D$;"PR#3"
30  INVERSE
40  PRINT
CHR$(27);"@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^";CHR$(24);
50  NORMAL
```

The 32 mouse characters have ASCII codes 64-95 ($40-$5F).

Using the Mouse as Paddles

With the Apple IIc, the mouse can either be used instead of the paddles (not true of the IIe), or as an X-Y pointing device in slot 4. If the mouse is turned on, the monitor ROM paddle routines will take input from the mouse instead of from the paddles. This is acceptable because the mouse and the paddles (and the joystick) are all plugged into the same port in the back of the Apple IIc. For example, a BASIC program that uses the PDL function to read from the paddles works just as well reading from the mouse. Try this:

1. Boot DOS 3.3 (the old one with LITTLE BRICK OUT on it).
2. Type `PR#4` and press RETURN to turn on the mouse.
3. Press CONTROL-A and then press RETURN to initialize the mouse.
4. Type `PR#0` and press RETURN to restore output to the screen.
5. Type `RUN LITTLE BRICK OUT` and press RETURN to run the program.

Play LITTLE BRICK OUT using the mouse instead of the paddles. Ignore the clicking noise when you move the mouse. This is a diagnostic aid that tells us that the mouse is alive and squeaking.

Using the Mouse From BASIC

If you would rather use the mouse in a more conventional manner, you can treat it as a device in slot 4. The general method is like this:

1. Initialize the mouse by printing a 1 to it.
2. Set input to come from slot 4.
3. INPUT X, Y, and button status from the mouse.
4. When done, set input to come from slot 0 (or 3).

Here is a BASIC program that demonstrates the use of the mouse. It reads from the mouse and prints the current values to the screen. When you press and then release the mouse button, the X and Y settings are reinitialized to 0. When a (readable) key is pressed, the program ends.

The X and Y coordinates are initialized to 0 when you print a 1 to the mouse firmware. They have a range from 0 to 1023. The mouse button returns values are as follows:

```
+/- 2  =  just pressed
+/- 1  =  still pressed
+/- 3  =  just released
+/- 4  =  still up
```

The value of the button status is normally positive. It becomes negative if a key is pressed.

The Built-in Printer Firmware

The Apple IIc printer firmware is intact and works from BASIC. However, its ID bytes do not identify it as any existing peripheral card. Thus, anyone (i.e. Pascal) that looks at ID bytes will not be able to use it. To use the serial Dot Matrix Printer (Imagewriter) from BASIC:

1. Set printer DIP switches like this:

   DN DN DN UP DN DN DN DN

   DN DN UP UP

2. Type `PR#1` to direct output to the printer.
3. Subsequent output goes to the printer.
4. Type `PR#0` (or `PR#3`) to redirect output to the screen.

By default, the printer firmware has the following settings:

- 9600 baud
- 8 data bits, 1 stop bit
- no parity
- 80-column line width with no video echo
- Line feed generated after RETURN
- Delay after line feed of 250 ms (1/4 second)
- Default command character is set to CONTROL-I.

These settings can be changed as described below.

Printer Firmware Commands

Once the printer firmware has been activated (by a PR#1), it operates very much like the Apple II Super Serial Card when it is in printer mode. Refer to the *Super Serial Card Manual* for more details on using the following commands. ^I means CONTROL-I.

^InnB Set baud rate to nn

Baud Rate	nn
50	1
75	2
110	3
135	4
150	5
300	6
600	7
1200	8
1800	9
2400	10
3600	11
4800	12
7200	13
9600	14
19200	15

^InnD Set data format bits to nn

Data Format	nn
8 data, 1 stop	0
7 data, 1 stop	1
6 data, 1 stop	2
5 data, 1 stop	3
8 data, 2 stop	4
7 data, 2 stop	5
6 data, 2 stop	6
5 data, 2 stop	7

^II Enable video echo

^IK Disable linefeed after CR

^IL Enable linefeed after CR

^InnN Disable video echo and set printer width to nn. nn is printer width in decimal.

^InnP Set parity bits to nn

Parity	nn
none	0,2,4,6
odd	1
even	3
MARK	5
SPACE	7

^IZ Zap control commands

^IX Set command char to ^X (default, ^I)

^InnCR Set printer width (CR = carriage return). Video echo must be disabled.

The Built-in Communcations Firmware

The Apple IIc communications firmware is intact and works from BASIC. Its ID bytes identify it as an Apple II communications card to most programs, and as a Super Serial Card to Access II.

Refer to the Apple II Super Serial Card manual for a description of the use of the communications firmware (Chapter 3, Communications Mode).

Communications Firmware Commands

Refer to the Super Serial Card manual for more details on the use of the following commands. ^A means CONTROL-A.

^AnnB Set baud rate to nn

Baud Rate	nn
50	1
75	2
110	3
135	4
150	5
300	6
600	7
1200	8
1800	9
2400	10
3600	11
4800	12
7200	13
9600	14
19200	15

^AnnD Set data formmat bits to nn

Data Format	nn
8 data, 1 stop	0
7 data, 1 stop	1
6 data, 1 stop	2
5 data, 1 stop	3
8 data, 2 stop	4
7 data, 2 stop	5
6 data, 2 stop	6
5 data, 2 stop	7

^AI Enable video echo

^AK Disable linefeed after CR

^AL Enable linefeed after CR

^AnnN Disable video echo and set printer width to nn. nn is printer width in decimal.

^AnnP Set parity bits to nn

Parity	nn
none	0,2,4,6
odd	1
even	3
MARK	5
SPACE	7

^AQ Quit terminal mode

^AR Reset the ACIA, IN#0, PR#0

^AS Send a 233 ms break character

^AT Enter Terminal mode

^AZ Zap control commands

^AX Set command char to ^X (default, ^A)

^AnnCR Set printer width (CR = carriage return). Video echo must be disabled.