


Revisions


Split


Unified

 **beneater** revised this gist on May 31, 2023.

 1 changed file with 38 additions and 35 deletions.

▼ 73

 wozmon.s




...
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74


```
@@ -12,76 +12,76 @@ MODE = $2B  
  
IN = $0200 ; Input buffer  
  
- KBD = $D010 ; PIA.A keyboard  
  input  
- KBD CR = $D011 ; PIA.A keyboard  
  control register  
- DSP = $D012 ; PIA.B display  
  output register  
- DSP CR = $D013 ; PIA.B display  
  control register  
  
RESET:  
- CLD ; Clear decimal  
  arithmetic mode.  
- CLI  
- LDY #$7F ; Mask for DSP  
  data direction reg.  
- STY DSP ; Set it up.  
- LDA #$A7 ; KBD and DSP  
  control register mask.  
- STA KBD CR ; Enable  
  interrupts, set CA1, CB1 for  
- STA DSP CR ; positive edge  
  sense/output mode.  
  
NOT CR:  
- CMP #$DF ; Backspace key?  
  BEQ BACKSPACE ; Yes.  
- CMP #$9B ; ESC?  
  BEQ ESCAPE ; Yes.  
  INY ; Advance text  
  index.  
  BPL NEXTCHAR ; Auto ESC if line  
  longer than 127.  
  
ESCAPE:  
- LDA #$DC ; "\".  
  JSR ECHO ; Output it.  
  
GETLINE:  
- LDA #$8D ; Send CR  
  JSR ECHO  
  
  LDY #$01 ; Initialize text  
  index.  
  BACKSPACE: DEY ; Back up text  
  index.  
  BMI GETLINE ; Beyond start of  
  line, reinitialize.  
  
NEXTCHAR:  
- LDA KBD CR ; Key ready?  
  BPL NEXTCHAR ; Loop until  
  ready.  
- LDA KBD ; Load character.  
  B7 should be '1'.  
  
  STA IN,Y ; Add to text  
  buffer.  
  JSR ECHO ; Display  
  character.  
- CMP #$8D ; CR?  
  BNE NOT CR ; No.  
  
  LDY #$FF ; Reset text  
  index.  
  LDA #$00 ; For XAM mode.  
  TAX ; X=0.  
  
SET STOR:  
  ASL ; Leaves $7B if  
  setting STOR mode.  
- SETMODE:  
- STA MODE ; $00 = XAM, $7B =  
  STOR, $AE = BLOK XAM.  
BL SKIP:  
  INY ; Advance text  
  index.  
NEXTITEM:  
  LDA IN,Y ; Get character.  
- CMP #$8D ; CR?  
  BEQ GETLINE ; Yes, done this  
  line.  
- CMP #$AE ; "."?  
  BCC BL SKIP ; Skip delimiter.  
- BEQ SETMODE ; Set BLOCK XAM  
  mode.  
- CMP #$BA ; ":"?
```

12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```
; $00=XAM, $7F=STOR, $AE=BLOCK XAM  
  
IN = $0200 ; Input buffer  
  
+ ACIA_DATA = $5000  
+ ACIA_STATUS = $5001  
+ ACIA_CMD = $5002  
+ ACIA_CTRL = $5003  
  
RESET:  
+ LDA #$1F ; 8-N-1, 19200  
  baud.  
+ STA ACIA_CTRL  
+ LDA #$0B ; No parity, no  
  echo, no interrupts.  
+ STA ACIA_CMD  
+ LDA #$1B ; Begin with  
  escape.  
  
NOT CR:  
+ CMP #$08 ; Backspace key?  
  BEQ BACKSPACE ; Yes.  
+ CMP #$1B ; ESC?  
  BEQ ESCAPE ; Yes.  
  INY ; Advance text  
  index.  
  BPL NEXTCHAR ; Auto ESC if line  
  longer than 127.  
  
ESCAPE:  
+ LDA #$5C ; "\".  
  JSR ECHO ; Output it.  
  
GETLINE:  
+ LDA #$0D ; Send CR  
  JSR ECHO  
  
  LDY #$01 ; Initialize text  
  index.  
  BACKSPACE: DEY ; Back up text  
  index.  
  BMI GETLINE ; Beyond start of  
  line, reinitialize.  
  
NEXTCHAR:  
+ LDA ACIA_STATUS ; Check status.  
+ AND #$08 ; Key ready?  
  
+ BEQ NEXTCHAR ; Loop until  
  ready.  
+ LDA ACIA_DATA ; Load character.  
  B7 will be '0'.  
  STA IN,Y ; Add to text  
  buffer.  
  JSR ECHO ; Display  
  character.  
+ CMP #$0D ; CR?  
  BNE NOT CR ; No.  
  
  LDY #$FF ; Reset text  
  index.  
  LDA #$00 ; For XAM mode.  
  TAX ; X=0.  
  
+ SETBLOCK:  
+ ASL  
SET STOR:  
  ASL ; Leaves $7B if  
  setting STOR mode.  
+ STA MODE ; $00 = XAM, $74 =  
  STOR, $B8 = BLOK XAM.  
  
BL SKIP:  
  INY ; Advance text  
  index.  
NEXTITEM:  
  LDA IN,Y ; Get character.  
+ CMP #$0D ; CR?  
  BEQ GETLINE ; Yes, done this  
  line.  
+ CMP #$2E ; "."?  
  BCC BL SKIP ; Skip delimiter.  
+ BEQ SETBLOCK ; Set BLOCK XAM  
  mode.  
+ CMP #$3A ; ":"?
```

					Revisions · wozmon.s				
75		BEQ	SETSTOR	; Yes, set STOR	75		BEQ	SETSTOR	; Yes, set STOR
	mode.					mode.			
76	-	CMP	#\$D2	; "R"?	76	+	CMP	#\$52	; "R"?
77		BEQ	RUN	; Yes, run user	77		BEQ	RUN	; Yes, run user
	program.					program.			
78		STX	L	; \$00 -> L.	78		STX	L	; \$00 -> L.
79		STX	H	; and H.	79		STX	H	; and H.
80		STY	YSAV	; Save Y for	80		STY	YSAV	; Save Y for
	comparison					comparison			
81					81				
82	NEXTHEX:				82	NEXTHEX:			
83		LDA	IN,Y	; Get character	83		LDA	IN,Y	; Get character
	for hex test.					for hex test.			
84	-	EOR	#\$B0	; Map digits to	84	+	EOR	#\$30	; Map digits to
	\$0-9.					\$0-9.			
85		CMP	#\$0A	; Digit?	85		CMP	#\$0A	; Digit?
86		BCC	DIG	; Yes.	86		BCC	DIG	; Yes.
87		ADC	#\$88	; Map letter "A"-	87		ADC	#\$88	; Map letter "A"-
	"F" to \$FA-FF.					"F" to \$FA-FF.			
...	@@ -132,17 +132,17 @@ SETADR:	LDA	L-1,X						; Copy hex data to
132					132				
133	NXTPRNT:				133	NXTPRNT:			
134		BNE	PRDATA	; NE means no	134		BNE	PRDATA	; NE means no
	address to print.					address to print.			
135	-	LDA	#\$8D	; CR.	135	+	LDA	#\$0D	; CR.
136		JSR	ECHO	; Output it.	136		JSR	ECHO	; Output it.
137		LDA	XAMH	; 'Examine index'	137		LDA	XAMH	; 'Examine index'
	high-order byte.					high-order byte.			
138		JSR	PRBYTE	; Output it in hex	138		JSR	PRBYTE	; Output it in hex
	format.					format.			
139		LDA	XAML	; Low-order	139		LDA	XAML	; Low-order
	'examine index' byte.					'examine index' byte.			
140		JSR	PRBYTE	; Output it in hex	140		JSR	PRBYTE	; Output it in hex
	format.					format.			
141	-	LDA	#\$BA	; ":".	141	+	LDA	#\$3A	; ":".
142		JSR	ECHO	; Output it.	142		JSR	ECHO	; Output it.
143					143				
144	PRDATA:				144	PRDATA:			
145	-	LDA	#\$A0	; Blank.	145	+	LDA	#\$20	; Blank.
146		JSR	ECHO	; Output it.	146		JSR	ECHO	; Output it.
147		LDA	(XAML,X)	; Get data byte at	147		LDA	(XAML,X)	; Get data byte at
	'examine index'.					'examine index'.			
148		JSR	PRBYTE	; Output it in hex	148		JSR	PRBYTE	; Output it in hex
	format.					format.			
...	@@ -173,15 +173,18 @@ PRBYTE:								
173					173				
174	PRHEX:				174	PRHEX:			
175		AND	#\$0F	; Mask LSD for hex	175		AND	#\$0F	; Mask LSD for hex
	print.					print.			
176	-	ORA	#\$B0	; Add "0".	176	+	ORA	#\$30	; Add "0".
177	-	CMP	#\$BA	; Digit?	177	+	CMP	#\$3A	; Digit?
178		BCC	ECHO	; Yes, output it.	178		BCC	ECHO	; Yes, output it.
179		ADC	#\$06	; Add offset for	179		ADC	#\$06	; Add offset for
	letter.					letter.			
180					180				
181	ECHO:				181	ECHO:			
182	-	BIT	DSP	; DA bit (B7)	182	+	PHA		; Save A.
	cleared yet?								
183	-	BMI	ECHO	; No, Wait for	183	+	STA	ACIA_DATA	; Output
	display.					character.			
184	-	STA	DSP	; Output	184	+	LDA	#\$FF	; Initialize delay
	character. Sets DA.					loop.			
					185	+	TXDELAY:	DEC	; Decrement A.
					186	+	BNE	TXDELAY	; Until A gets to
						0.			
					187	+	PLA		; Restore A.
185		RTS		; Return.	188		RTS		; Return.
186					189				
187	.org \$FFFA				190	.org \$FFFA			
...									

 beneater created this gist on May 31, 2023.

▼ 191 wozmon.s 				
...	@@ -0,0 +1,191 @@			
		1	+	.org \$8000
		2	+	.org \$ff00
		3	+	
		4	+	XAML = \$24 ; Last "opened"
				location Low
		5	+	XAMH = \$25 ; Last "opened"
				location High
		6	+	STL = \$26 ; Store address
				Low
		7	+	STH = \$27 ; Store address
				High
		8	+	L = \$28 ; Hex value
				parsing Low
		9	+	H = \$29 ; Hex value
				parsing High
		10	+	YSAV = \$2A ; Used to see if
				hex value is given
		11	+	MODE = \$2B ; \$00=XAM,
				\$7F=STOR, \$AE=BLOCK XAM
		12	+	
		13	+	IN = \$0200 ; Input buffer
		14	+	
		15	+	KBD = \$D010 ; PIA.A keyboard
				input
		16	+	KBDCR = \$D011 ; PIA.A keyboard
				control register
		17	+	DSP = \$D012 ; PIA.B display
				output register
		18	+	DSPCR = \$D013 ; PIA.B display
				control register

```
19 +
20 + RESET:
21 +             CLD             ; Clear decimal
   + arithmetic mode.
22 +             CLI
23 +             LDY     #$7F     ; Mask for DSP
   + data direction reg.
24 +             STY     DSP      ; Set it up.
25 +             LDA     #$A7     ; KBD and DSP
   + control register mask.
26 +             STA     KBDCR    ; Enable
   + interrupts, set CA1, CB1 for
27 +             STA     DSPCR    ;  positive edge
   + sense/output mode.
28 +
29 + NOTCR:
30 +             CMP     #$DF     ; Backspace key?
31 +             BEQ     BACKSPACE ; Yes.
32 +             CMP     #$9B     ; ESC?
33 +             BEQ     ESCAPE   ; Yes.
34 +             INY
   + index.
35 +             BPL     NEXTCHAR ; Auto ESC if line
   + longer than 127.
36 +
37 + ESCAPE:
38 +             LDA     #$DC     ; "\".
39 +             JSR     ECHO     ; Output it.
40 +
41 + GETLINE:
42 +             LDA     #$8D     ; Send CR
43 +             JSR     ECHO
44 +
45 +             LDY     #$01     ; Initialize text
   + index.
46 + BACKSPACE:  DEY
   + index.
47 +             BMI     GETLINE  ; Beyond start of
   + line, reinitialize.
48 +
49 + NEXTCHAR:
50 +             LDA     KBDCR    ; Key ready?
51 +             BPL     NEXTCHAR ; Loop until
   + ready.
52 +             LDA     KBD      ; Load character.
   + B7 should be '1'.
53 +             STA     IN,Y     ; Add to text
   + buffer.
54 +             JSR     ECHO     ; Display
   + character.
55 +             CMP     #$8D     ; CR?
56 +             BNE     NOTCR    ; No.
57 +
58 +             LDY     #$FF     ; Reset text
   + index.
59 +             LDA     #$00     ; For XAM mode.
60 +             TAX            ; X=0.
61 + SETSTOR:
62 +             ASL
   + setting STOR mode.
63 + SETMODE:
64 +             STA     MODE     ; $00 = XAM, $7B =
   + STOR, $AE = BLOK XAM.
65 + BLSKIP:
66 +             INY
   + index.
67 + NEXTITEM:
68 +             LDA     IN,Y     ; Get character.
69 +             CMP     #$8D     ; CR?
70 +             BEQ     GETLINE  ; Yes, done this
   + line.
71 +             CMP     #$AE     ; ", "?
72 +             BCC     BLSKIP   ; Skip delimiter.
73 +             BEQ     SETMODE  ; Set BLOCK XAM
   + mode.
74 +             CMP     #$BA     ; ": "?
75 +             BEQ     SETSTOR  ; Yes, set STOR
   + mode.
76 +             CMP     #$D2     ; "R"?
77 +             BEQ     RUN      ; Yes, run user
   + program.
78 +             STX     L        ; $00 -> L.
79 +             STX     H        ;   and H.
80 +             STY     YSAV     ; Save Y for
   + comparison
81 +
82 + NEXTHEX:
83 +             LDA     IN,Y     ; Get character
   + for hex test.
84 +             EOR     #$B0     ; Map digits to
   + $0-9.
85 +             CMP     #$0A     ; Digit?
86 +             BCC     DIG      ; Yes.
87 +             ADC     #$88     ; Map letter "A"-
   + "F" to $FA-FF.
88 +             CMP     #$FA     ; Hex letter?
89 +             BCC     NOTHEX   ; No, character
   + not hex.
90 + DIG:
91 +             ASL
92 +             ASL
   + of A.
93 +             ASL
94 +             ASL
95 +
96 +             LDX     #$04     ; Shift count.
97 + HEXSHIFT:
```

98	+		ASL		; Hex digit left,
		MSB to carry.			
99	+		ROL	L	; Rotate into LSD.
100	+		ROL	H	; Rotate into
		MSD's.			
101	+		DEX		; Done 4 shifts?
102	+		BNE	HEXSHIFT	; No, loop.
103	+		INY		; Advance text
		index.			
104	+		BNE	NEXTHEX	; Always taken.
		Check next character for hex.			
105	+				
106	+	NOTHEX:			
107	+		CPY	YSAV	; Check if L, H
		empty (no hex digits).			
108	+		BEQ	ESCAPE	; Yes, generate
		ESC sequence.			
109	+				
110	+		BIT	MODE	; Test MODE byte.
111	+		BVC	NOTSTOR	; B6=0 is STOR, 1
		is XAM and BLOCK XAM.			
112	+				
113	+		LDA	L	; LSD's of hex
		data.			
114	+		STA	(STL,X)	; Store current
		'store index'.			
115	+		INC	STL	; Increment store
		index.			
116	+		BNE	NEXTITEM	; Get next item
		(no carry).			
117	+		INC	STH	; Add carry to
		'store index' high order.			
118	+	TONEXTITEM:	JMP	NEXTITEM	; Get next command
		item.			
119	+				
120	+	RUN:			
121	+		JMP	(XAML)	; Run at current
		XAM index.			
122	+				
123	+	NOTSTOR:			
124	+		BMI	XAMNEXT	; B7 = 0 for XAM,
		1 for BLOCK XAM.			
125	+				
126	+		LDX	#\$02	; Byte count.
127	+	SETADR:	LDA	L-1,X	; Copy hex data to
128	+		STA	STL-1,X	; 'store index'.
129	+		STA	XAML-1,X	; And to 'XAM
		index'.			
130	+		DEX		; Next of 2 bytes.
131	+		BNE	SETADR	; Loop unless X =
		0.			
132	+				
133	+	NXTPRNT:			
134	+		BNE	PRDATA	; NE means no
		address to print.			
135	+		LDA	#\$8D	; CR.
136	+		JSR	ECHO	; Output it.
137	+		LDA	XAMH	; 'Examine index'
		high-order byte.			
138	+		JSR	PRBYTE	; Output it in hex
		format.			
139	+		LDA	XAML	; Low-order
		'examine index' byte.			
140	+		JSR	PRBYTE	; Output it in hex
		format.			
141	+		LDA	#\$BA	; ":".
142	+		JSR	ECHO	; Output it.
143	+				
144	+	PRDATA:			
145	+		LDA	#\$A0	; Blank.
146	+		JSR	ECHO	; Output it.
147	+		LDA	(XAML,X)	; Get data byte at
		'examine index'.			
148	+		JSR	PRBYTE	; Output it in hex
		format.			
149	+	XAMNEXT:	STX	MODE	; 0 -> MODE (XAM
		mode).			
150	+		LDA	XAML	
151	+		CMP	L	; Compare 'examine
		index' to hex data.			
152	+		LDA	XAMH	
153	+		SBC	H	
154	+		BCS	TONEXTITEM	; Not less, so no
		more data to output.			
155	+				
156	+		INC	XAML	
157	+		BNE	MOD8CHK	; Increment
		'examine index'.			
158	+		INC	XAMH	
159	+				
160	+	MOD8CHK:			
161	+		LDA	XAML	; Check low-order
		'examine index' byte			
162	+		AND	#\$07	; For MOD 8 = 0
163	+		BPL	NXTPRNT	; Always taken.
164	+				
165	+	PRBYTE:			
166	+		PHA		; Save A for LSD.
167	+		LSR		
168	+		LSR		
169	+		LSR		; MSD to LSD
		position.			
170	+		LSR		
171	+		JSR	PRHEX	; Output hex
		digit.			
172	+		PLA		; Restore A.
173	+				
174	+	PRHEX:			

		Revisions · wozmon.s			
	175	+	AND	#\$0F	; Mask LSD for hex
		print.			
	176	+	ORA	#\$B0	; Add "0".
	177	+	CMP	#\$BA	; Digit?
	178	+	BCC	ECH0	; Yes, output it.
	179	+	ADC	#\$06	; Add offset for
		letter.			
	180	+			
	181	+	ECH0:		
	182	+	BIT	DSP	; DA bit (B7)
		cleared yet?			
	183	+	BMI	ECHO	; No, Wait for
		display.			
	184	+	STA	DSP	; Output
		character. Sets DA.			
	185	+	RTS		; Return.
	186	+			
	187	+	.org	\$FFFA	
	188	+			
	189	+	.word	\$0F00	; NMI vector
	190	+	.word	RESET	; RESET vector
	191	+	.word	\$0000	; IRQ vector