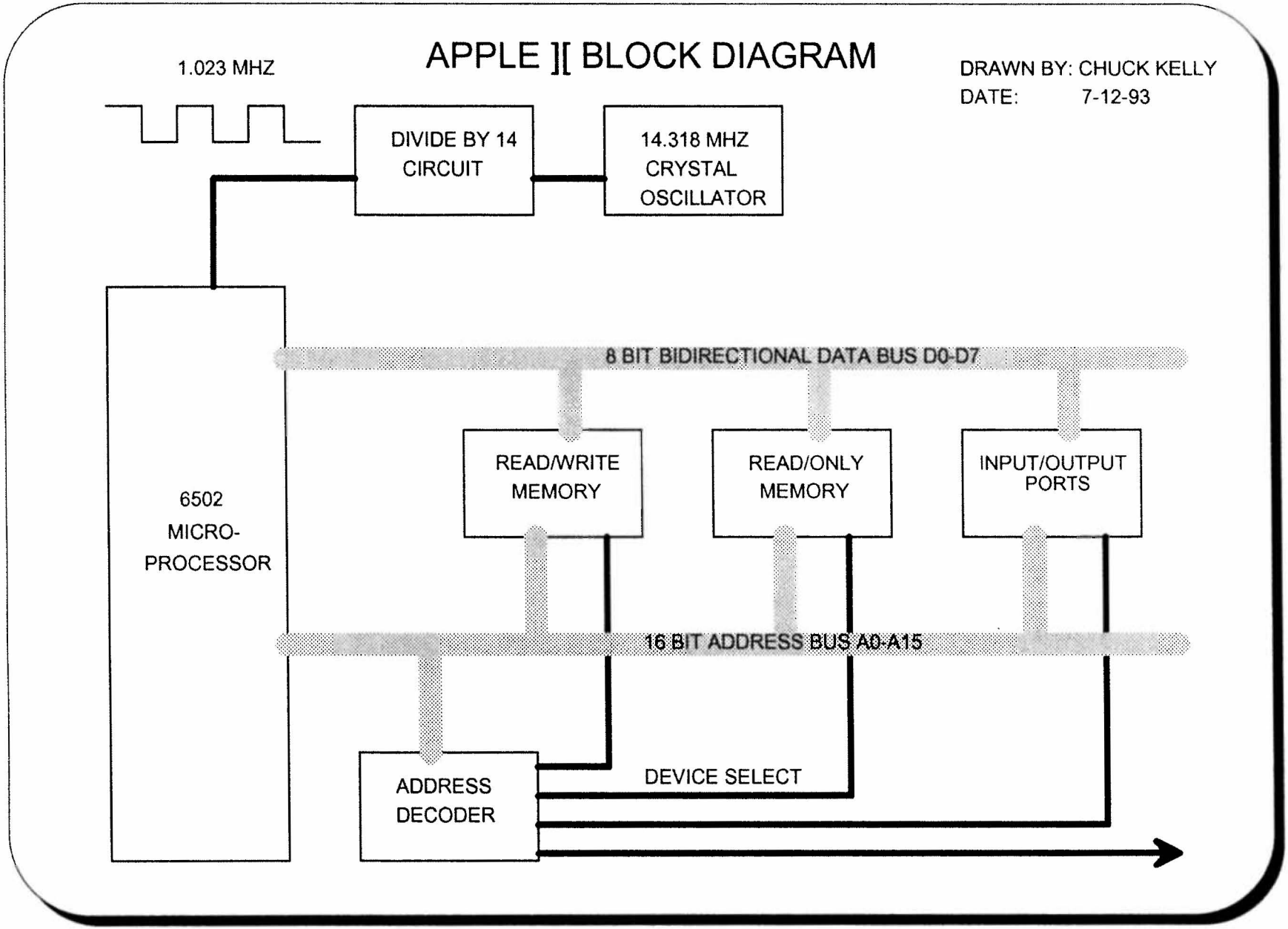
Kansas Fest 93

Session: Peripheral Design

Presenter: Chuck Kelly / ProDev, Inc.

APPLE][BLOCK DIAGRAM

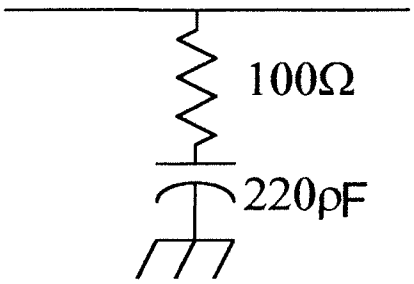
DRAWN BY: CHUCK KELLY
DATE: 7-12-93



BUS TERMINATION

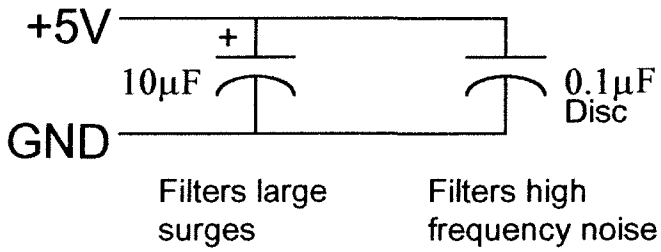
APPLE BUS IS NOT TERMINATED
ASYNCHRONOUS* SIGNALS MAY NEED TO BE TERMINATED.

* This is any signal used without a clock qualifier. Address and data lines should always be used with a clock qualifier.

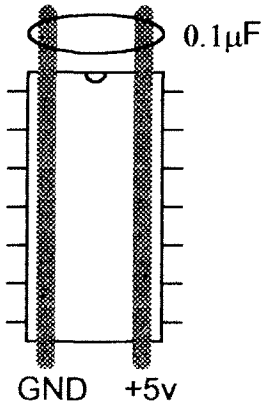


This termiantion method puts a minimal load on the bus but still provides noise reduction.

POWER FILTER

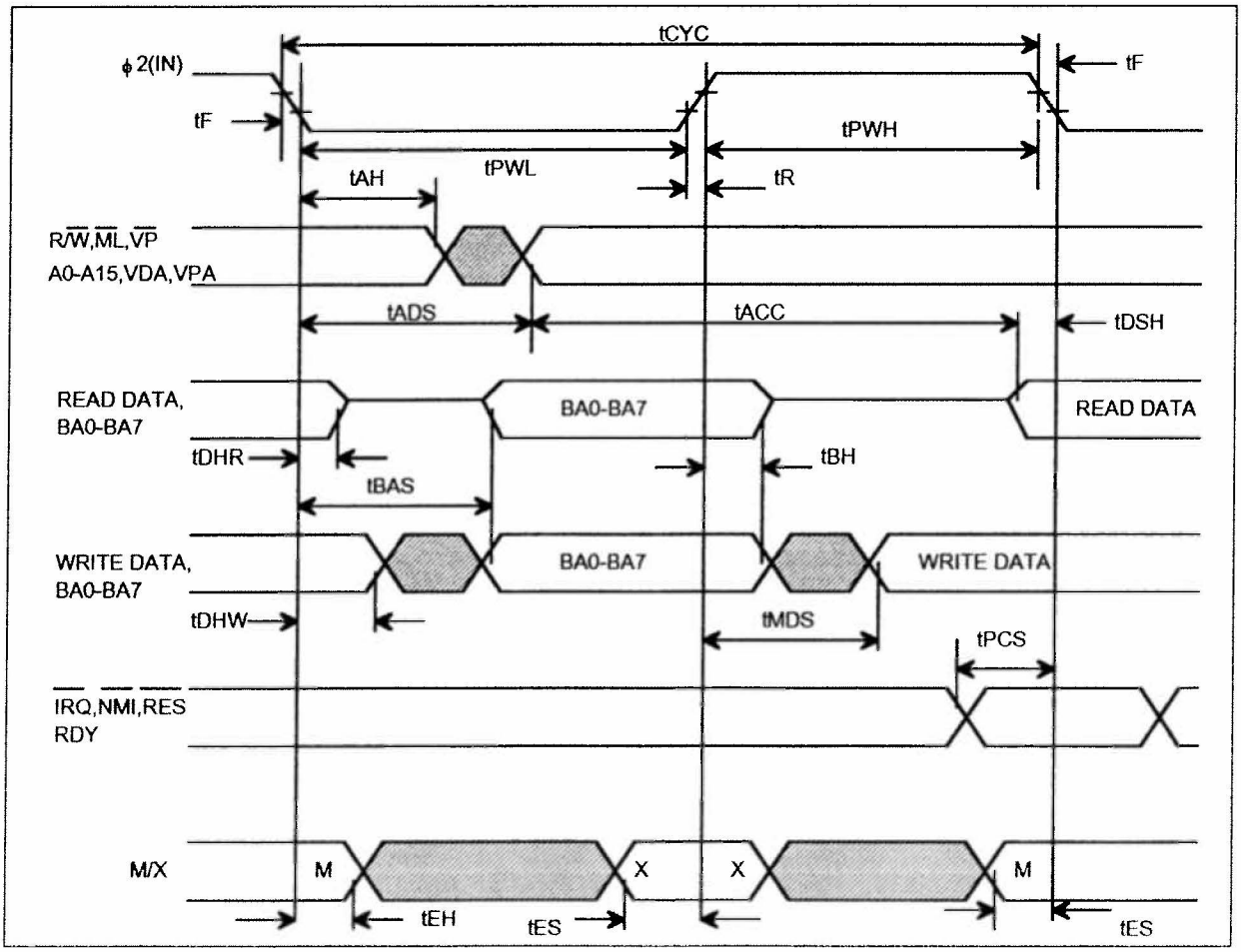


USE A BYPASS CAP AT EACH I.C.



Parameter	Symbol	2 MHz		4 MHz		Unit
		Min	Max	Min	Max	
Cycle Time	t_{CYC}	500	DC	250	DC	nS
Clock Pulse Width Low	t_{PWL}	0.24	10	0.12	10	µS
Clock Pulse Width High	t_{PWH}	240		120		nS
Fall Time, Rise Time	t_F,t_R	-	10	-	10	nS
A0-A15 Hold Time	t_{AH}	10	-	10	-	nS
A0-A15 Setup Time	t_{ADS}	-	100	-	75	nS
BA0-BA7 Hold Time	t_{BH}	10	-	10	-	nS
BA0-BA7 Setup Time	t_{BAS}	-	100	-	90	nS
Access Time	t_{ACC}	365	-	130	-	nS
Read Data Hold Time	t_{DHR}	10	-	10	-	nS
Read Data Setup Time	t_{DSR}	40	-	30	-	nS
Write Data Delay Time	t_{MDS}	-	100	-	70	nS
Write Data Hold Time	t_{DHW}	10	-	10	-	nS
Processor Control Setup Time	t_{PCS}	40	-	30	-	nS
Processor Control Hold Time	t_{PCH}	10	-	10	-	nS
E,MX Output Hold Time	t_{EH}	10	-	10	-	nS
E,MX Output Setup Time	t_{ES}	50	-	50	-	nS
Capacitive Load (Address,Data,R/W)	C_{EXT}	-	100	-	100	pF
BE to High Impedance State	t_{BHZ}	-	30	-	30	nS
BE to Valid Data	t_{BVD}	-	30	-	30	nS

AC Characteristics (W65C816)



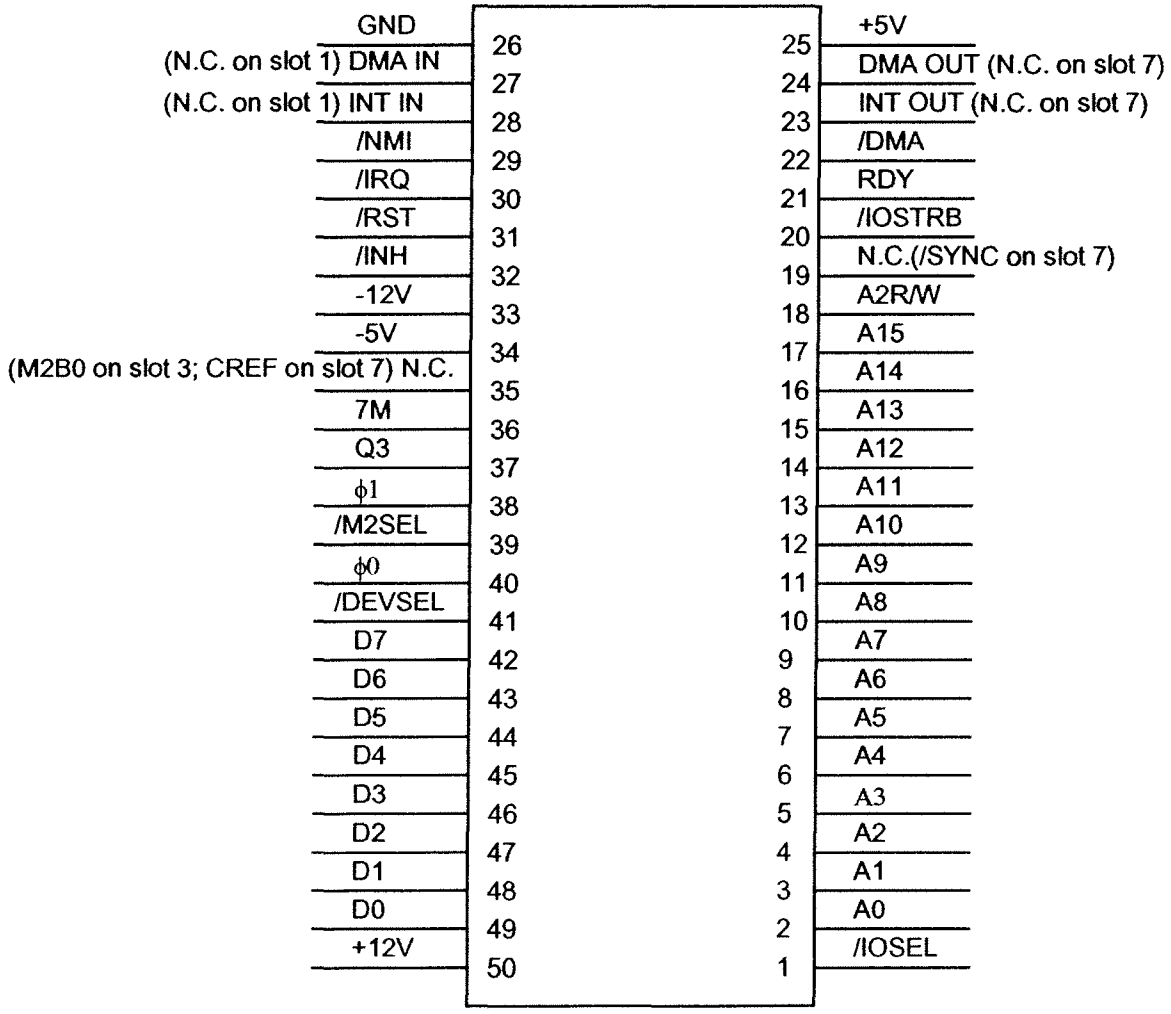
Timing Diagram (W65C816)

The expansion slots

The figure below is a diagram of one of the seven expansion slots from an Apple][GS computer. The signals present on this connector are essentially the same as those present on every Apple][computer ever built.

In order to accommodate expansion cards built for the][+ and //e the][GS bus operates at the same clock speed as the older machines.

You can find a complete description of the signals present on the expansion connector in the Apple][GS hardware reference manual.

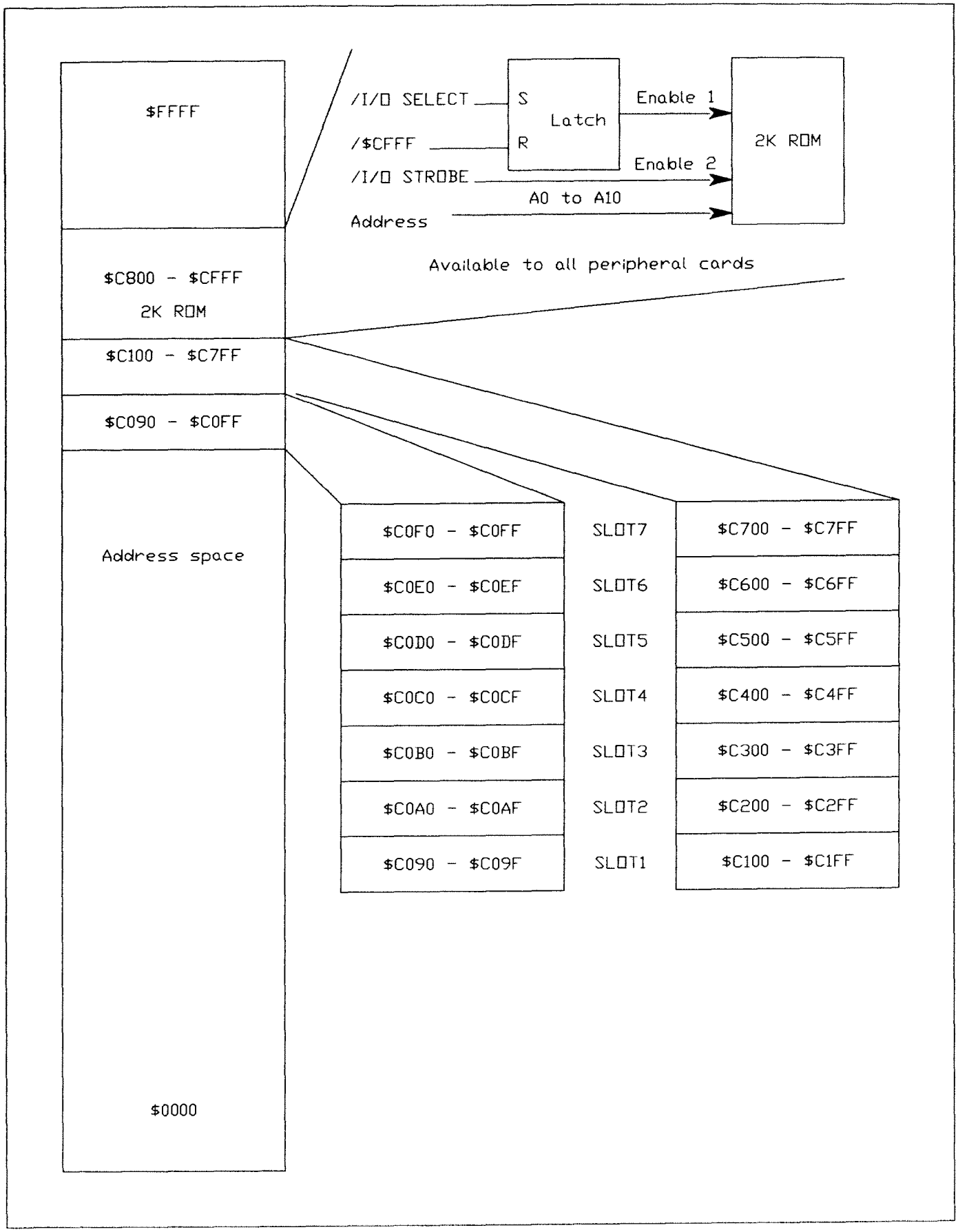


Signal Descriptions

/IOSEL Goes low when a peripheral cards $Cnxx space is active.

/IOSTRB Goes low when $C800 - $CFFF is accessed.

/DEVSEL Goes low when a peripheral cards $C0nx space is active, where n is the connector number plus 8.


$FFFF
$C800 - $CFFF
2K ROM
$C100 - $C7FF
$C090 - $C0FF
Address space
$0000
/I/O SELECT
/$CFFF
/I/O STROBE
Address
S
R
Latch
Enable 1
Enable 2
A0 to A10
2K ROM
Available to all peripheral cards
$C0F0 - $C0FF
$C0E0 - $C0EF
$C0D0 - $C0DF
$C0C0 - $C0CF
$C0B0 - $C0BF
$C0A0 - $C0AF
$C090 - $C09F
SLOT7
SLOT6
SLOT5
SLOT4
SLOT3
SLOT2
SLOT1
$C700 - $C7FF
$C600 - $C6FF
$C500 - $C5FF
$C400 - $C4FF
$C300 - $C3FF
$C200 - $C2FF
$C100 - $C1FF

***** NOTICE ***** This print is confidential property of PRODEV, Incorporated. It is not to be used or reproduced in any manner, nor submitted to other parties for examination without the prior approval of PRODEV, Incorporated.

NOTE!

THESE DIMENSIONS EXCEED GS APPLE SPECS. SEE APPLE TECH NOTE #28.

THIS IS AN EXAMPLE OF WHAT YOU WOULD SEND TO A PC BOARD MAKER.

NOTE 1 RADIUS OF CORNERS 0.06 UNLESS OTHERWISE INDICATED.

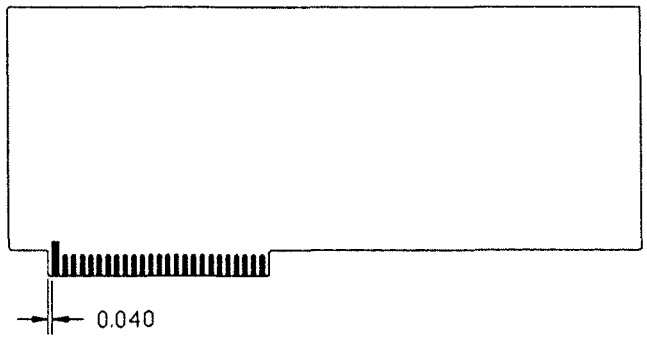
NOTE 2 ALL HOLE SIZES GIVEN ARE AFTER PLATING.

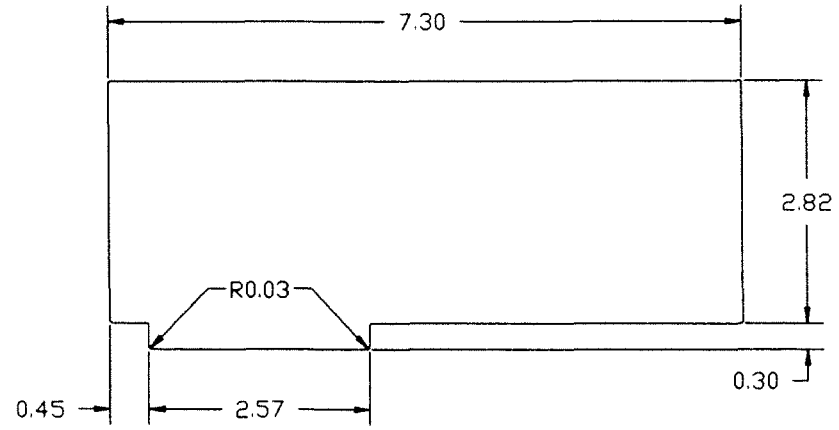
NOTE 3 CONNECTOR FINGERS ARE GOLD OVER NICKEL PLATING.

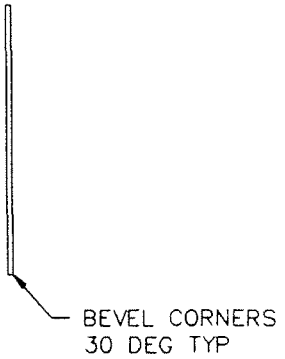
NOTE 4 MATERIAL FR4 0.062 THICK

HOLE CHART

LABEL	SIZE	QUANTITY
NONE	0.037"	542

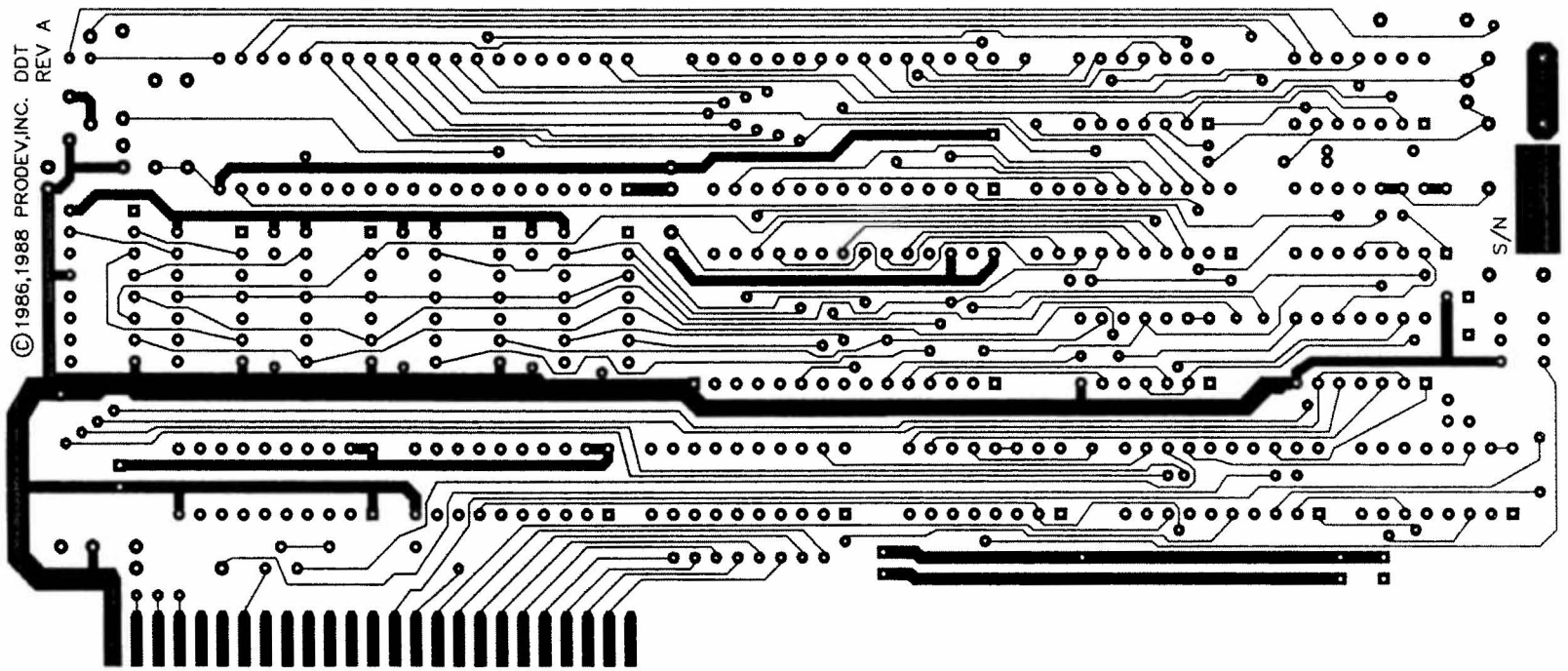






TOLERANCES*						
Fractional dimensions	± 1/32	DRN C.K.	DATE 8/30/1988		PRODEV, INC. MONROE, MICHIGAN	DDT rev A
Decimal dimensions	± 0.01	CK'D		SCALE		
Angles	± 20 DEG	APP'D		PART NAME DDT		
(X) unless otherwise specified						

EXAMPLE ARTWORK



SOLDER SIDE DDT rev A

1.0000

```
*******************************
*  COMDOFF - TURN EXT DISPLAY OFF
*******************************

S_COMDOFF =    ]segnum     ;segment number of this routine
COMDOFF  BIT   OFFFLAG
                           ;DISPLAY ALREADY OFF ?
         BMI   GETCOM0
                           ;IF YES
         JSR   TRANSFR0    ;RESTORE DISPLAY SWITCHES
         DFB   RESTTEXTC   ;code
         LDA   #$80
         STA   OFFFLAG
                           ;DON'T DISPLAY TO SCREEN

*----------------------------------------
*   GETCOM0 - do <CR> & wait for user command
*----------------------------------------
GETCOM0
         PEA   GETCOMCR-1  ;address of command
         PEA   S_GETCOM    ;segment # of command
         JMP   JUMPSEG0    ;goto command in other segment

*-------------------------------------------------
***** GLOBAL SUBROUTINES IN THIS SEGMENT *****
***** MAXIMUM OF 32 *****
***** CODE BYTES ARE EQUATED AS FOLLOWS
*
* BITS 0-2 = SEGMENT NUMBER $0 THRU $7 OF SUBROUTINE
* BITS 3-7 = NUMBER OF SUBROUTINE IN SUBTABL

SUBTABL0

LDAINDYC EQU   *-SUBTABL0*4+0+$100
         DA    LDAINDY-1

STAINDYC EQU   *-SUBTABL0*4+0+$100
         DA    STAINDY-1

STEP1C   EQU   *-SUBTABL0*4+0+$100
         DA    STEP1-1

EXECUTEC EQU   *-SUBTABL0*4+0+$100
         DA    EXECUTE-1

WSTKRESC EQU   *-SUBTABL0*4+0+$100
         DA    WSTKRES-1

AX2S1_AC EQU   *-SUBTABL0*4+0+$100
         DA    AX2S1_A-1

************* SEGMENT CROSSOVER AREA *************

         LST   ON
S0END    =     $E0CF91-*
         do    nolist
         LST   OFF
         fin
         ERR   *-1/$E0CF91
```

THESE 2 LINES MAKE A CALL TO THE SUBROUTINE RESTTEXT THAT IS LOCATED IN A DIFFERENT SEGMENT OF THE ROM.

THESE 3 LINES DO A JUMP TO GETCOMCR THAT IS LOCATED IN A DIFFERENT SEGMENT

THIS TABLE CONTAINS ALL THE SUBROUTINES THAT RESIDE IN THIS SEGMENT OF THE ROM AND ARE CALLABLE FROM OTHER SEGMENTS.

```
         DS    $E0CF91-*,$FF

******** SAVE THE ACC, X, Y AND P REGISTERS *******
* Returns with MX = 11, saves registers

SAVEAXP0
         PHP               ;SAVE STATUS
         MX16
         STX   XSAVESEG    ;save 16 bits
         STY   YSAVESEG    ;save 16 bits
         STA   ASAVESEG    ;save 16 bits
         MX8
         PLA               ;GET STATUS
         STA   PSAVESEG    ;SAVE
         RTS

****** RESTORE THE ACC, X, Y AND P REGISTERS ******
* restores registers

RESTAXP0
         MEMORY8
         LDA   PSAVESEG
         PHA
         MX16
         LDX   XSAVESEG
         LDY   YSAVESEG
         LDA   ASAVESEG
         PLP

*** THIS RTS IS USED BY THE FINDSLOT ROUTINE ***
PUTSLOT  RTS
         MX    %11

*----------------------------------------
* Do a direct transfer to other segments

JUMPSEG0
         JSR   SAVEAXP0
         LDY   SLOTN0
         PLA               ;pull junk byte from dest. seg
         PLA               ;get destination segment
         STA   SEGMBASE,Y  ;the next inst' will be in new seg
         JSR   RESTAXP0    ;restore after xfer from other seg
         RTS               ;pull destination address from stack

* TRANSFER TO OTHER SEGMENTS
* Upper byte of X index is zeroed

TRANSFR0
         JSR   SAVEAXP0
         MEMORY16
         PLA               ;get return address from stack
         INC               ;inc to point at code byte & for RTS
         PHA
         MEMORY8
         LDA   #0          ;CURRENT SEG #
         PHA
```

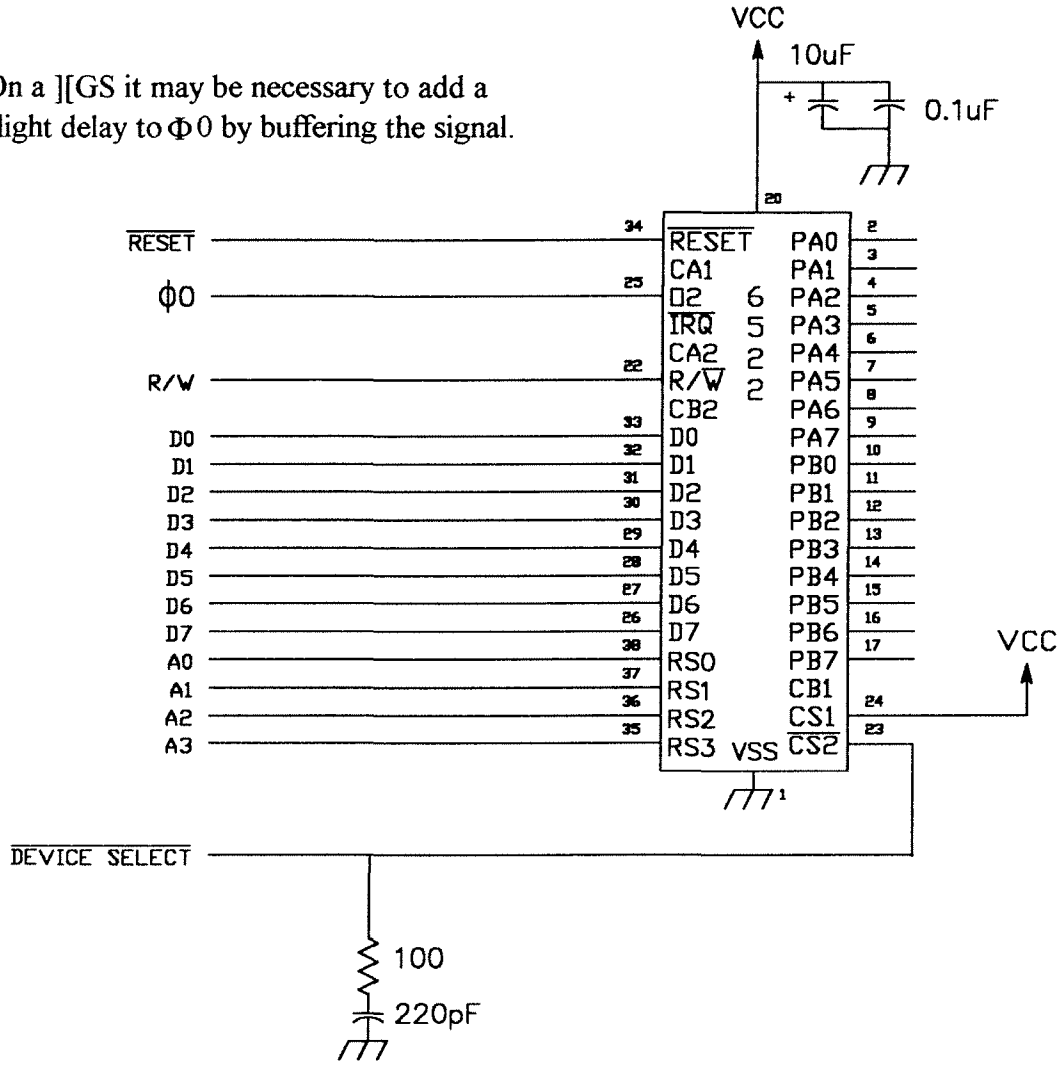
```
         LDY   #0
         LDA   (2,S),Y     ;GET CODE BYTE
         PHA               ;SAVE CODE
         AND   #$07        ;STRIP ALL BUT SEG #
         LDY   SLOTN0
         STA   SEGMBASE,Y ;NEXT INSTR. RUN FROM NEW SEGMENT
* NEW SEGMENT
         PLA               ;GET CODE
         PEA   RETURN0     ;where to return to
         AND   #$F8        ;STIP OFF SEG# LEAVING SUB #
         LSR
         LSR               ;LEAVE SUB# MULTIPLIED BY 2
* GET ADDRESS OF SUB FROM SUBTABL & PUSH ON STACK
         TAY
         MEMORY16
         LDA   SUBTABL0,Y
         PHA
         BRA   RESTAXP0    ;RESTORE REGISTERS, RTS TO SUBROUTINE
         MX    %11

* RETURN HERE FROM SUBROUTINE

RETURN0  EQU   *-1
         JSR   SAVEAXP0
         PLA               ;SEG # TO RETURN TO
         LDY   SLOTN0
         STA   SEGMBASE,Y ;RETURN TO SEGMENT
         BRA   RESTAXP0
         DS    \,$FF       ;PUT OBJECT AT NEXT PAGE
```

16 channel I/O card

On a][GS it may be necessary to add a slight delay to Φ0 by buffering the signal.



Drawn by: Chuck Kelly

Programming for the 16 channel I/O card.

Each pin on port A and B can be configured to be either an input or an output. This is done by writing a 1 to the corresponding bit in the data direction register for each pin that is to be an output and writing a 0 for each pin that is to be a input.

The following sample code assumes the I/O card is located in slot #6.

A simple program to read the 8 inputs from port A and echo them to port B.

```
DDRA  EQU   $C0E3          ; data direction register for port A
DDRB  EQU   $C0E2          ; data direction register for port B
PORTA EQU   $C0E1
PORTB EQU   $C0E0

      LDA   #00
      STA   DDRA           ; make inputs of port A
      LDA   #$FF
      STA   DDRB           ; make ouputs of port B

LOOP  LDA   PORTA
      STA   PORTB
      BRA   LOOP
```

The registers of the 6522 would be located at the following addresses if the card were plugged into slot #6.

Address	Write	Read
$C0E0	Output Register B	Input Register B
$C0E1	Output Register A	Input Register A
$C0E2	Data Direction Register B	
$C0E3	Data Direction Register A	
$C0E4	T1 Low-Order Latches	T1 Low-Order Counter
$C0E5	T1 High-Order Counter	
$C0E6	T1 Low-Order Latches	
$C0E7	T1 High-Order Latches	
$C0E8	T2 Low-Order Latches	T2 Low-Order Counter
$C0E9	T2 High-Order Counter	
$C0EA	Shift Register	
$C0EB	Auxiliary Control Register	
$C0EC	Peripheral Control Register	
$C0ED	Interrupt Flag Register	
$C0EE	Interrupt Enable Register	
$C0EF	Same as Reg 1 Except No "Handshake"	

For more information on the 6522 and other hardware I recommend the following books.

Apple II Assembly Language by: Marvin L. De Jong
Howard W. Sams & Co

Microcomputer Electronics by: Daniel L. Metzger
Prentice Hall