# Preface

The Revision 5 6809 board was not designed by Bob Applegate. It is a continuation of his previous work and is intended to help the hobby community. With Bob's passing it left the only readily available boards for the SS50 Bus being the reproduction SWTPC boards that I make. Where possible I leave the original Corsham manuals intact only making the changes necessary for the upgraded boards. Prior to Bob's passing he had sent me some of his Eagle Design files and library files to help speed the development of my SWTPC replacement motherboard. With Bob's design files it was possible to quickly make reproductions of most of his boards. The continued availability of Corsham's boards gives the hobbyist more choice in building a retro SS50 system.

Frederic Brown

Peripheral Technology
Retired

**Corsham Technologies, LLC**
*www.corshamtech.com*
*617 Stokes Road, Suite 4-299*
*Medford, NJ 08055*

# 6809 CPU Board

## Introduction

Thank you for buying our 6809 CPU board!

This was a big project, and definitely the most complicated hardware project we've done so far. At the Vintage Computer Festival Midwest in 2014, I had our first SS-50 products on display and received a warm welcome. However, a lot of people said they really wanted a 6809 based board, so this is the result of all those requests.

Is this board vintage? Well, work started in 2014, so technically it is not. However, it uses a design very similar to the original SWTPC 6809 CPU board using parts available at that time. The large RAM and EPROMs are not vintage, particularly the 128K RAM chip. The board is vintage in that it uses the SS-50 bus and can plug into existing systems or work with other boards of that era.

Using older parts has been a problem because some of them have not been made in a long time, so prices are high, conditions of pulled chips are unknown, and we have to test a lot more components to verify they actually work as expected. Fortunately all the chips on this board are available from surplus inventories, but eventually they will be unavailable.

## Features

- 6809 running at 2 MHz.
- Baud rate generator provides all standard SS-50/SS-30 clocks.
- One baud rate line can be jumpered for higher speed options.
- 2K, 4K or 6K of EPROM. SBUG uses 2K, but the other 2K can be enabled for user extensions.
- Dynamic Address Translation that is fully compatible with SWTPC's scheme.
- 128K of RAM, fixed in banks 0 and 1. Each bank can be enabled/disabled.
- A16 to A19 available on the SS-50 bus, individually selectable.

## Reset and NMI

In the upper left hand corner of the board is a reset pushbutton switch along with jumpers JP1 (RESET) and JP9 (NMI). These two jumpers can be wired to external buttons on the chassis to provide reset and NMI signals to the processor. There is also a jumper block like SWTPC used for the RESET and NMI. This can connect to the existing wiring in a SWTPC cabinet.

## Baud Rate Selection

The SS-50 bus used five lines for baud rate clocks, while the SS-50C bus allowed those lines to be used for either those clocks or the extended addressing lines A16 to A19. Our board allows individual jumper selection for each pin using five jumpers located on the lower left hand corner of the board:

| Bus Pin | Jumper | SS-50 (6800) Use | SS-50C (6809) Use |
|---|---|---|---|
| 46 | JP2 | VAR baud | VAR baud or BUSRQ |
| 47 | JP13 | VAR baud | VAR baud or A19 |
| 48 | JP12 | 1200 baud | 1200 baud or A18 |
| 49 | JP11 | 600 baud | 600 baud or A17 |
| 50 | JP10 | 300 baud | 300 baud or A16 |

Note that the baud clocks are actually x16, that is, they are 16 times faster than the indicating baud.

Because we didn't want to tie up all those pins, our board is optimized to use pin 46, normally the 110 baud line, as a VARiable baud rate line. You'll see JP2 allows you to select either BUSRQ or VAR, and jumper J6 allows you to configure this line as 150, 2400, 4800, 9600, 19200 or 38400 baud.

Phew, that's a lot of options, and might not be very clear at all, so here is our recommendation on how to set up those six jumpers to give you a desired baud rate and also the full 20 bits of address space:

| Jumper | Suggested setting | Result |
|---|---|---|
| JP10 | A16 | Gives 17 bit addressing |
| JP11 | A17 | Gives 18 bit addressing |
| JP12 | A18 | Gives 19 bit addressing |
| JP13 | A19 | Gives 20 bit addressing |
| JP2 | VAR | Makes the 110 line one of six selectable baud rates |
| JP6 (VAR) | Your choice | Select this to provide your designed console baud rate setting. We use 9600. |

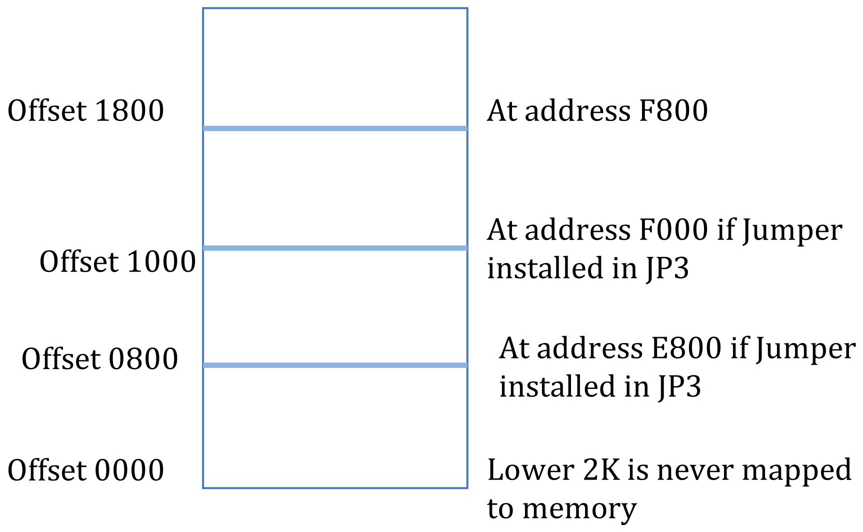
## EPROM

The original SWTPC 6809 board had a 2K EPROM with SBUG in the top of memory, from F800 to FFFF, along with 2 additional EPROM sockets.  Our board has a single 8K EPROM.  The top 2K of EPROM is always selected. Memory at F000-F7FF and E800-EFFF can be enabled by jumpers. V4 and earlier boards used a 2764 EPROM. V5 allows for a 2764, 27128 or 27256.  Only 8K is mapped to the processor at a time and the lower 2K is blocked from use.  A 27128 will allow  you to switch between 2 monitor programs such as SBUG and OS9.  With a 27256 you can switch between 4 monitor programs.  The selection of the monitor is controlled by Jumper JP7 and JP8.

| EPROM | JP8 | JP7 | Address Selected |
|---|---|---|---|
| 2764 | 1 | 1 | 0000-1FFF |
| 27128 | 0 | 1 | 0000-1FFF |
| 27128 | 1 | 1 | 2000-3FFF |
| 27256 | 0 | 0 | 0000-1FFF |
| 27256 | 1 | 1 | 2000-3FFF |
| 27256 | 0 | 1 | 4000-5FFF |
| 27256 | 1 | 1 | 6000-7FFF |

If you wish to burn your own EPROM, this is where things are located  for a 27C64 or for each selected part of a 27128 or 27256.

The first 2K is completely unused and is not visible.  The upper 4K is mapped to F000-FFFF.


Offset 1800
At address F800
Offset 1000
At address F000 if Jumper installed in JP3
Offset 0800
At address E800 if Jumper installed in JP3
Offset 0000
Lower 2K is never mapped to memory

## RAM

The board has 128K of RAM available but must be configured via JP3 on the upper left hand side of the board.

Banks 0 and 1 can be enabled or disabled by jumpers. Typically only RAM at 0000-FFFF is needed. FLEX and OS9 Level 1 don't support more. Should you wish to experiment you can enable the second Bank. The SWTPC utility SBOX will report the total memory from both banks. (112K)

## Dynamic Address Translation

You don't really need to read this section unless you plan on writing software that uses the extended memory, in which case it's good to understand how SWTPC mapped 1 MB of address space into a processor with only 64K of address space. They did this with Dynamic Address Translation, or DAT. DAT uses 16 RAM locations to map a 16 bit address from the processor into a 20 bit address space.

The top four address lines (A12 to A15) are used as address select lines to 16 bytes of memory. The lower 4 bits of each address map to A12 through A15. The upper 4 bits are A16 to A19.

The top page of memory (FF00 to FFFF) is always mapped to the top 256 bytes of the EPROM. When SBUG starts, it loads up the DAT registers to map 56K of memory from 0000 to DFFF.

Addresses FFF0 to FFFF are the write-only DAT registers. If you read those locations you'll get the contents of EPROM, not the DAT registers. Each register maps one 4K block of memory:

| Address | Block | Default value |
|---|---|---|
| FFF0 | 0xxx | 0F |
| FFF1 | 1xxx | 0E |
| FFF2 | 2xxx | 0D |
| FFF3 | 3xxx | 0C |
| FFF4 | 4xxx | 0B |
| FFF5 | 5xxx | 0A |
| FFF6 | 6xxx | 09 |
| FFF7 | 7xxx | 08 |
| FFF8 | 8xxx | 07 |
| FFF9 | 9xxx | 06 |
| FFFA | Axxx | 05 |
| FFFB | Bxxx | 04 |
| FFFC | Cxxx | 03 |
| FFFD | Dxxx | 02 |
| FFFE | Exxx | 01 |
| FFFF | F000 | 00 |

That's as clear as mud, right? Okay, the value written into the registers is the inverse of the value for the lower 4 bits, and the true value for the upper 4 bits. Still not clear, I know, so let's take an example:

| | | |
|---|---|---|
| FFF0 | 0xxx | 0F |

The value 00001111 (binary) is written into the register. When the upper four bits of the address (A12 to A15) are 0000, the entry above is used. The inverse of the lower four bits of DAT register at FFF0 is 0000 (since it has 00001111). So the values for A12 to A15 put onto the bus will be 0000.

So how do we use that? Well, let's assume you want to load and use two programs that are both start at address 0000 hex. You can select bank 0's memory by writing 0F to FFF0 and load the first program.

Now there are multiple ways to put another block of memory at address 0xxx. You can map another block from bank zero, such as moving the memory currently at 8000 down to 0000 by writing 07 hex to FFF0. The inverse of 7 (0111) is 8 (1000), so now when any address with 0000 as the top four bits is selected, the top four bits put onto the address bus will be 1000.

Another way is to use bank 1 so that all of bank 0's memory remains in place. To do this, put the value 0001 in the top 4 bits by writing 1F to FFF0. Now bank 1 will be selected for all 0xxx addresses.

Load up your second program to 0000 and you're set! To select the initial program again, write 0F to FFF0.

## Summary of Jumpers and Switches

There are a number of jumpers and switches on the board that change the behavior. While many of them are discussed in other sections of the manual, here is a summary:

| Label | Use |
|---|---|
| JP1 | External RESET button connection. Short these two pins together to force a reset. |
| JP2 | Chooses the pin connected to SS-50C bus line 46. It can be set to either VAR to select the baud rate from JP6, or BUSRQ to put the BUSRQ signal onto the bus. |
| JP3 | Allows user to disable bank 0 and 1 of RAM and disable EPROM at E800-EFFF and F000-F7FF |
| JP6 | VAR – This jumper block should have no more than one jumper installed to select the desired baud rate for the VAR line. Available baud rates are 150, 2400, 4800, 9600, 19200, 38400. The actual speed of these lines is 16 times faster. |
| JP7-JP8 | EPROM Selection |
| JP9 | External NMI button connection. Short these two pins together to force a non maskable interrupt (NMI). |
| JP10 | Selects the signal present on SS-50C pin 50. Can be the 300 baud (x16) clock or A16. |
| JP11 | Selects the signal present on SS-50C pin 49. Can be the 600 baud (x16) clock or A17. |
| JP12 | Selects the signal present on SS-50C pin 48. Can be the 1200 baud (x16) clock or A18. |
| JP13 | Selects the signal present on SS-50C pin 47. Can be the output from the JP VAR jumper block or A19. |

## Initial Terminal Settings

SBUG for the SWTPC is set for 8N1. SBUG for the Corsham Shield will be changed to 8N1 at a later date.

8N2. Eight data bits, no parity and two stop bits.

## SBUG/EEPROM

The contents of the EPROM with any given board might change over time. The current source code can be found  on this web page: https://peripheraltech.com/Corsham-6809.htm

### TINY BASIC

*Tiny BASIC is not currently supplied for the reproduction board.  However you may build a copy from the source code if you wish and program your own EPROM.*

If your EPROM has a label that has "TINY BASIC" on it, then there are two commands for doing a cold start and warm start to BASIC:

- ! = Cold start BASIC.  Do this first.
- @ = Warm start BASIC.

Do a cold start first, then you'll be able to write simple programs using a Tiny BASIC dialect.  The BASIC was slightly reworked to make it fit into the bottom 2K of the 4K EPROM on the board.  It has one new command ("!") which exits back to SBUG.  You can re-enter BASIC with the "@" command, which keeps all variables and your BASIC programs intact.

### SD UTILS

More recently, we've been installing our low-level drivers for the SD Card System into the EPROM, along with adding a "B" (Boot) command in SBUG.  The B command will load the first sector from drive 0 into memory at C100, then jump to it.  We provide a version of 6809 FLEX that can be run directly from the B command.

## ¡Viva Fiesta!

All of our circuit boards have something unusual on them, and since SWTPC was in San Antonio, it seemed the city would make for some interesting additions.  Fortunately, I have a friend who is a native of San Antonio, so I asked her for some ideas or else I'd resort to Googling for something appropriate.  She said that ¡Viva Fiesta! is a big festival held in San Antonio each year, so that seemed like a good choice.  I was also excited about this board, so the exclamation points fit into my enthusiasm for this project.

## Revision History

| Version | Changes |
|---|---|
| 4 | Basis for REV5 |
| 5 | Enhanced Reproduction of V4 |

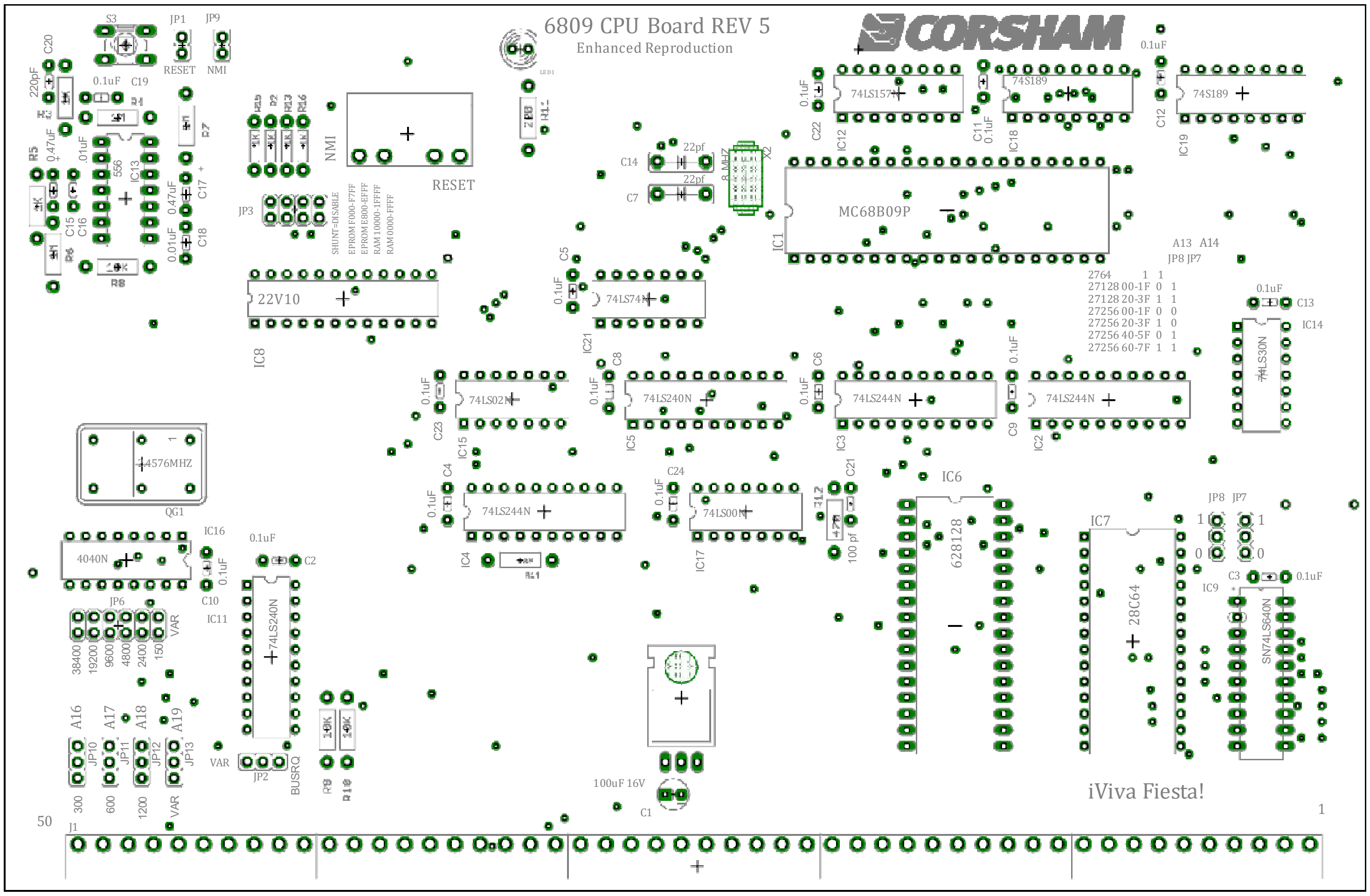
## Parts List

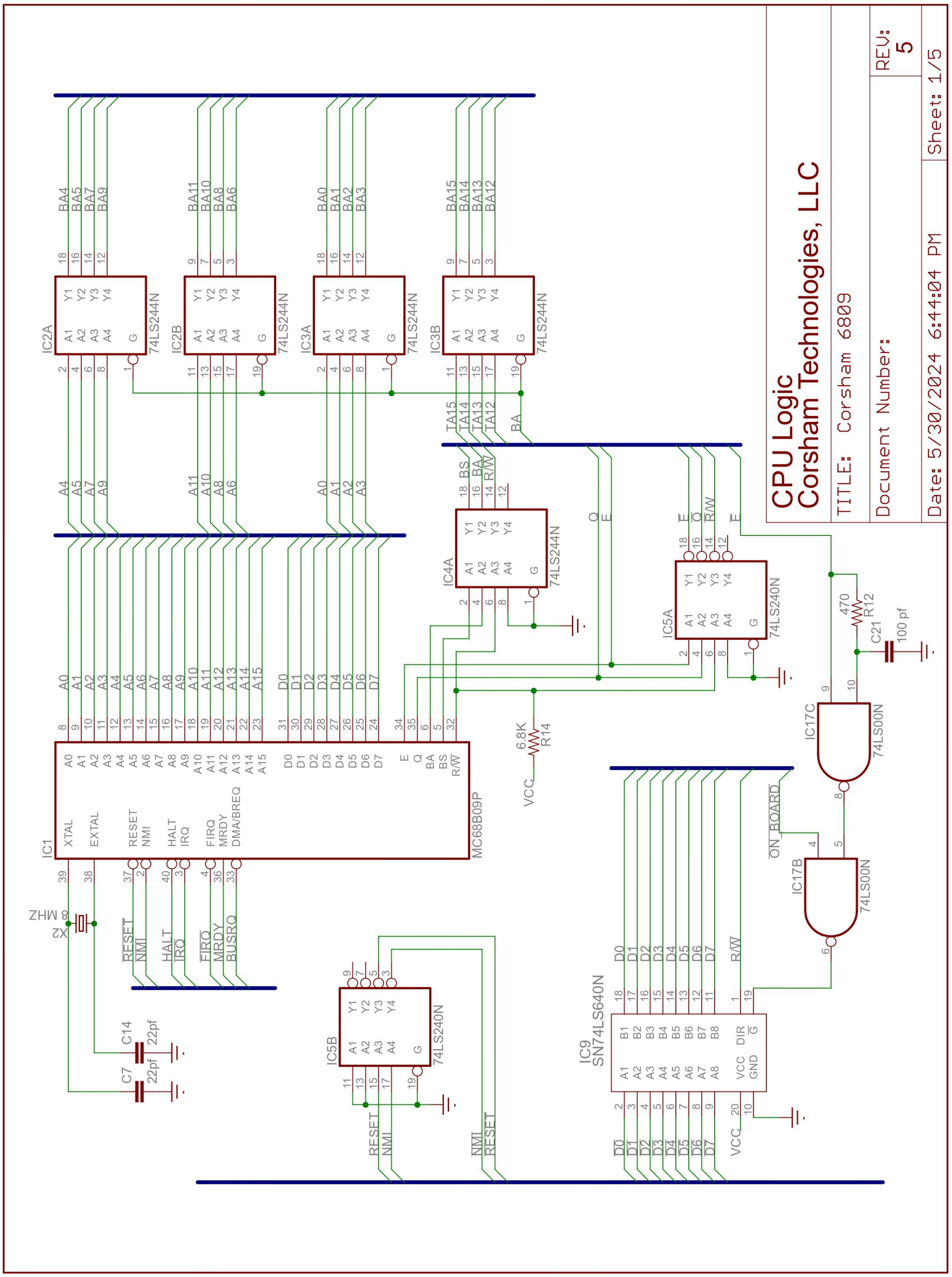
| Part | Number | Description |
|---|---|---|
| PCB | 1 | Printed Circuit Board |
| J1 | 5 | Molex 09-52-3101 |
| JP1, JP9 | 2 | 1x2 jumper block |
| JP2, JP10-13 J7,J8 | 7 | 1x3 jumper block |
| JP6 | 1 | 2x6 jumper block |
| S1 | 1 | 4 pin SPST pushbutton |
| C1 | 1 | 100uf, 16v electrolytic capacitor |
| C2-C6,C8-C13 C19,C22-C24 | 15 | .1 uf disc capacitor |
| C7, C14 | 1 | 22pf |
| C15, C17 | 2 | .47uf tantalum |
| C16, C18 | 2 | .01uf |
| C21 | 1 | 100pf |
| R1, R6, R7 | 3 | 1M ¼ watt |
| R3, R5 | 2 | 1K |
| R2,R8-10 R13,R15,R16 | 7 | 10K |
| R11 | 1 | 220 |
| R12 | 1 | 470 |
| R14 | 1 | 6.8K |
| QG1 | 1 | 2.4576 Oscillator (Full size or half) |
| X2 | 1 | 8 MHZ crystal |
| LED1 | 1 | 3mm LED (usually red, but does not matter) |
| VR1 | 1 | 7805 +5 VDC regulator, TO-220 case |
| IC1 | 1 | MC68B09 CPU |
| IC2, IC3, IC4 | 3 | 74LS244 |
| IC5, IC11 | 2 | 74LS240 |
| IC6 | 1 | 628128 128K SRAM |
| IC7 | 1 | 27C64, 27128 or 27256 EPROM |
| IC8 | 1 | F22V10C-15PU |
| IC9 | 1 | 74LS640 |
| IC12 | 1 | 74LS157 |
| IC13 | 1 | LM556 |
| IC14 | 1 | 74LS30 |
| IC15 | 1 | 74LS02 |
| IC16 | 1 | CD74HCT4040 |
| IC17 | 1 | 74LS00 |
| IC18, IC19 | 2 | 74LS189 |
| IC21 | 1 | 74LS74 |
|  | 5 | 14 pin IC socket |
|  | 4 | 16 pin IC socket |

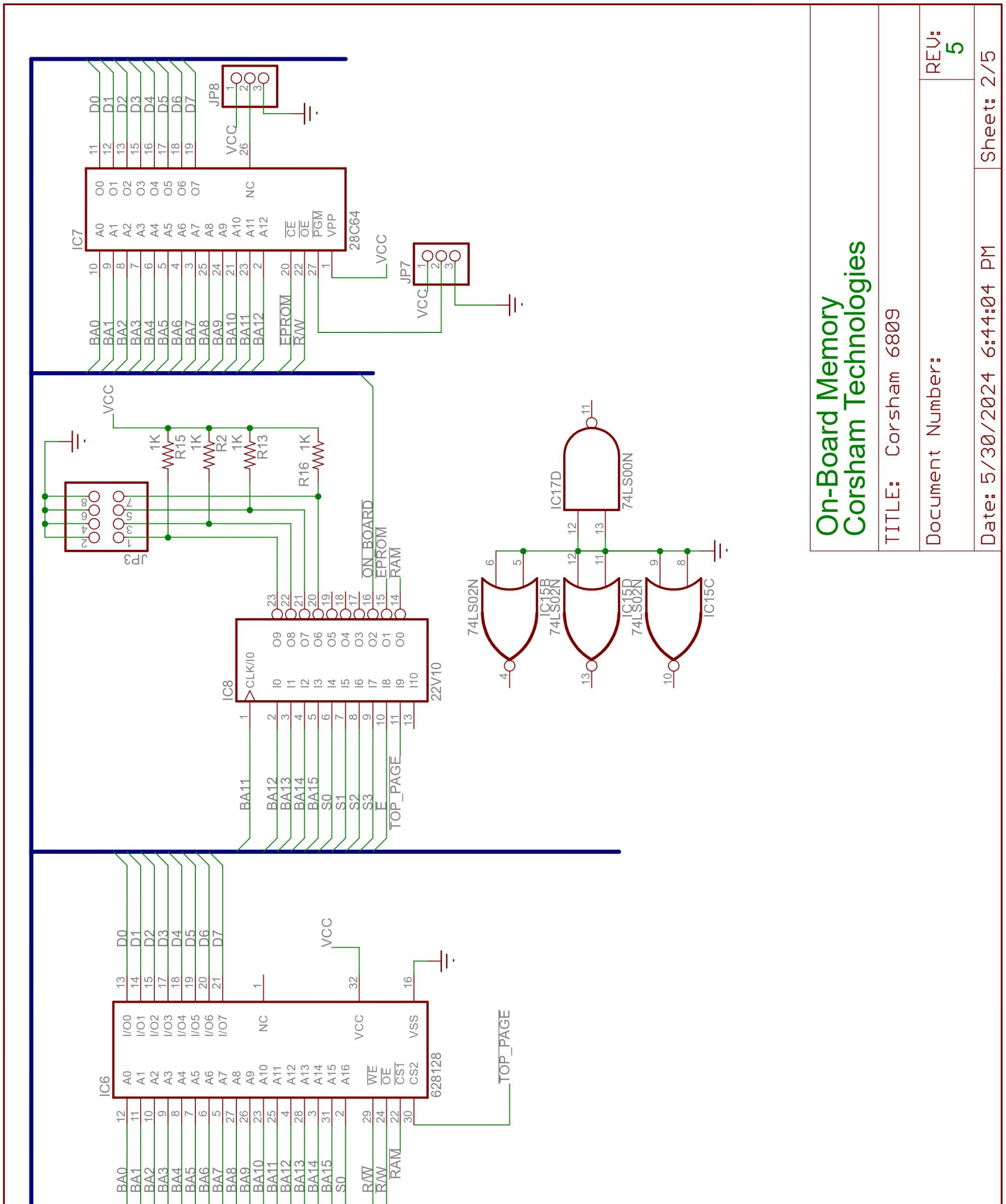
| | |
|---|---|
| 6 | 20 pin IC socket |
| 1 | 28 pin IC sockets for IC7 |
| 1 | 32 pin IC socket for IC6 |
| 1 | 40 pin socket for IC1 |

## PAL Equations

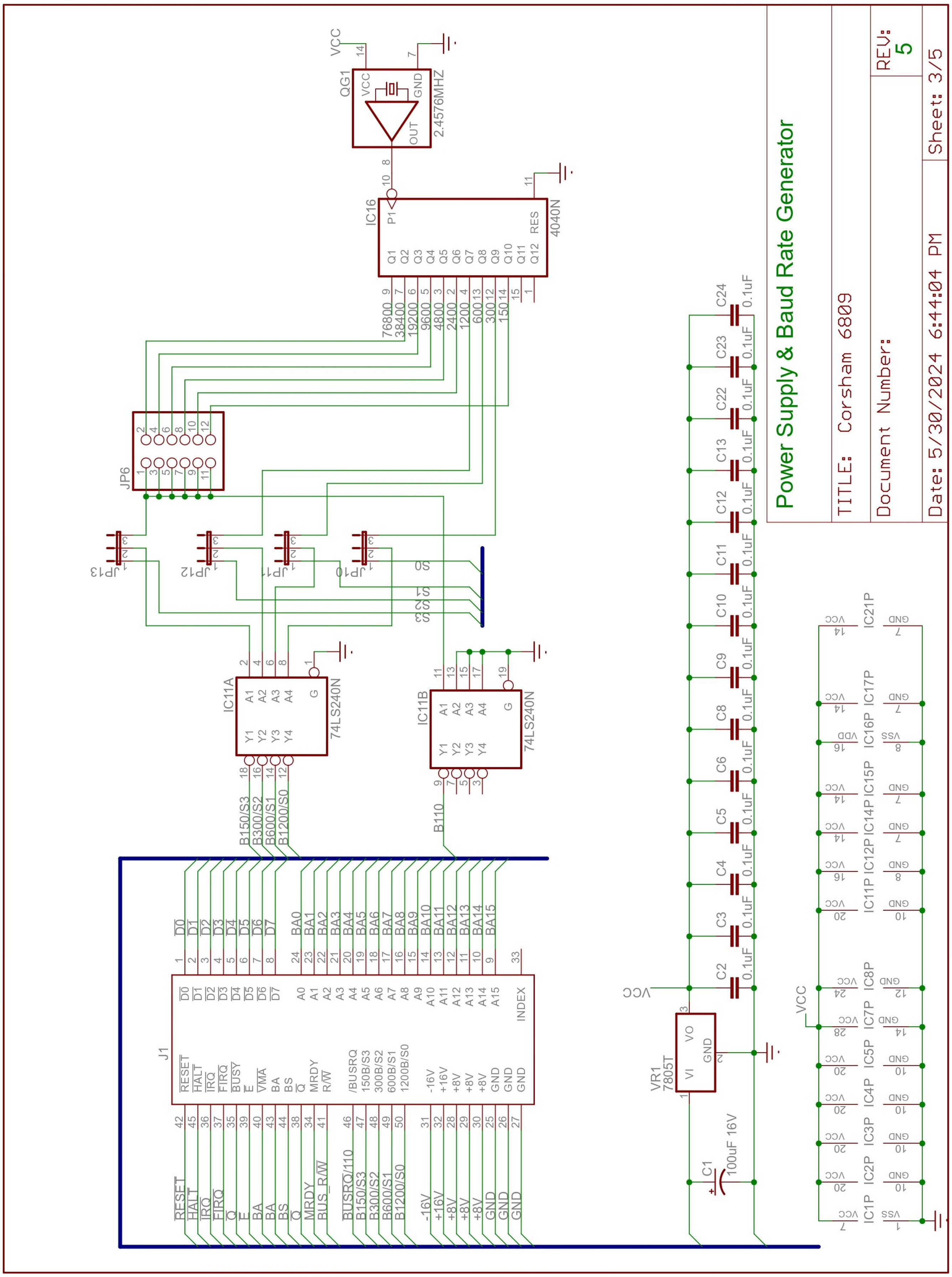
```
Name     Corsham 6809 ;
PartNo   IC8 ;
Date     03/24/24 ;
Revision 0 ;
Designer Frederic C Brown ;
Company  Peripheral Technology ;
Assembly None ;
Location  ;
Device   p22v10 ;

/* *************** INPUT PINS *********************/
PIN   1  = BA11                      ; /*                                 */
PIN   2  = BA12                      ; /*                                 */
PIN   3  = BA13                      ; /*                                 */
PIN   4  = BA14                      ; /*                                 */
PIN   5  = BA15                      ; /*                                 */
PIN   6  = S0                        ; /*  A16                            */
PIN   7  = S1                        ; /*  A17                            */
PIN   8  = S2                        ; /*  A18                            */
PIN   9  = S3                        ; /*  A19                            */
PIN  10  = E                         ; /*                                 */
PIN  11  = TPAGE                     ;
PIN  12  = GND                       ; /*                                 */
PIN  13  = NC0                       ; /*                                 */
PIN  17 =  NC1                       ;
PIN  18 =  NC2                       ;
PIN  19 =  NC3                       ;
PIN  20  = RAM0                      ; /* 0 - DISABLE RAM 0000-FFFF       */
PIN  21  = RAM1                      ; /* 0 - DISABLE RAM 10000-1FFFF     */
PIN  22  = ROME8                     ; /* 0 - DISABLE EPROM E800-EFFF     */
PIN  23  = ROMF0                     ; /* 0 - DISABLE EPROM F000-F7FF     */
PIN  24  = VCC                       ; /*                                 */

/* *************** OUTPUT PINS *********************/

PIN  16  = ONBRD                   ; /*  RAM OR EPROM ACCESS-ACTIVE 0     */
PIN  14  = RAM                     ; /*  RAM SELECT-ACTIVE LOW            */
PIN  15  = CSROM                   ; /*  EPROM SELECT - ACTIVE 0          */

       /* BUS S0-S3 IS INVERTED ON INTERNAL S0-S3 BOARD SIGNALS */
!RAM =  (  S3 &  S2 &  S1 &  S0 & !BA15 &                 RAM0 & !E ) /*Enable 0000-7FFF*/
      # (  S3 &  S2 &  S1 &  S0 &  BA15 & !BA14 &         RAM0 & !E ) /*Enable 8000-BFFF*/
      # (  S3 &  S2 &  S1 &  S0 &  BA15 &  BA14 & !BA13 & RAM0 & !E ) /*Enable C000-DFFF*/

      # (  S3 &  S2 &  S1 & !S0 & !BA15 &                 RAM1 & !E ) /*Enable 10000-17FFF*/
      # (  S3 &  S2 &  S1 & !S0 &  BA15 & !BA14 &         RAM1 & !E ) /*Enable 18000-1BFFF*/
      # (  S3 &  S2 &  S1 & !S0 &  BA15 &  BA14 & !BA13 & RAM1 & !E ) /*Enable 1C000-1DFFF*/
  ;

!CSROM = ( BA15 & BA14 & BA13 &  BA12 &  BA11 & !E )               /* Enable F800-FFFF */
       # ( BA15 & BA14 & BA13 &  BA12 & !BA11 & !E & ROMF0 )       /* Enable F000-FFFF */
       # ( BA15 & BA14 & BA13 & !BA12 &  BA11 & !E & ROME8 )       /* Enable E800-EFFF */
       # (!TPAGE & !E) ;


!ONBRD = ( !CSROM # !RAM ) ;                         /* Active low on EPROM or RAM access */
```

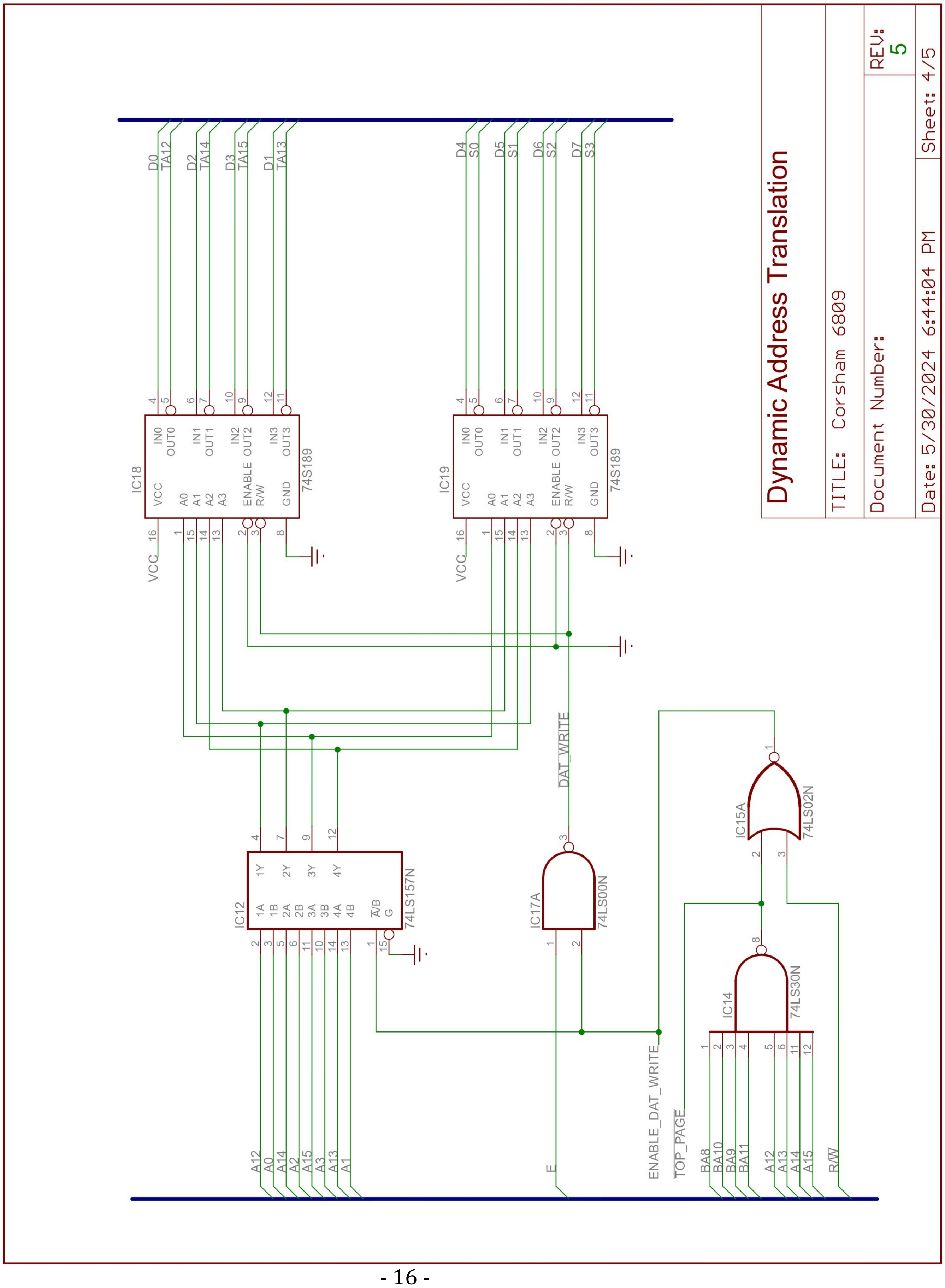

6809 CPU Board REV 5
Enhanced Reproduction
CORSHAM
MC68B09P
IC1
22V10
IC8
74LS157N
74S189
74S189
74LS74N
74LS02N
74LS240N
74LS244N
74LS244N
74LS30N
74LS244N
74LS00N
628128
IC6
28C64
IC7
SN74LS640N
IC9
4040N
74LS240N
556
1.84576MHZ
QG1
8 MHZ
X2
RESET
NMI
SHUNT=DISABLE
EPROM F000-F7FF
EPROM E800-EFFF
RAM 10000-1FFFF
RAM 0000-FFFF
JP3
JP6
38400
19200
9600
4800
2400
150
VAR
A16
A17
A18
A19
300
600
1200
VAR
BUSRQ
A13 A14
JP8 JP7
2764 1 1
27128 00-1F 0 1
27128 20-3F 1 1
27256 00-1F 0 0
27256 20-3F 1 0
27256 40-5F 0 1
27256 60-7F 1 1
100uF 16V
¡Viva Fiesta!
50
1
J1

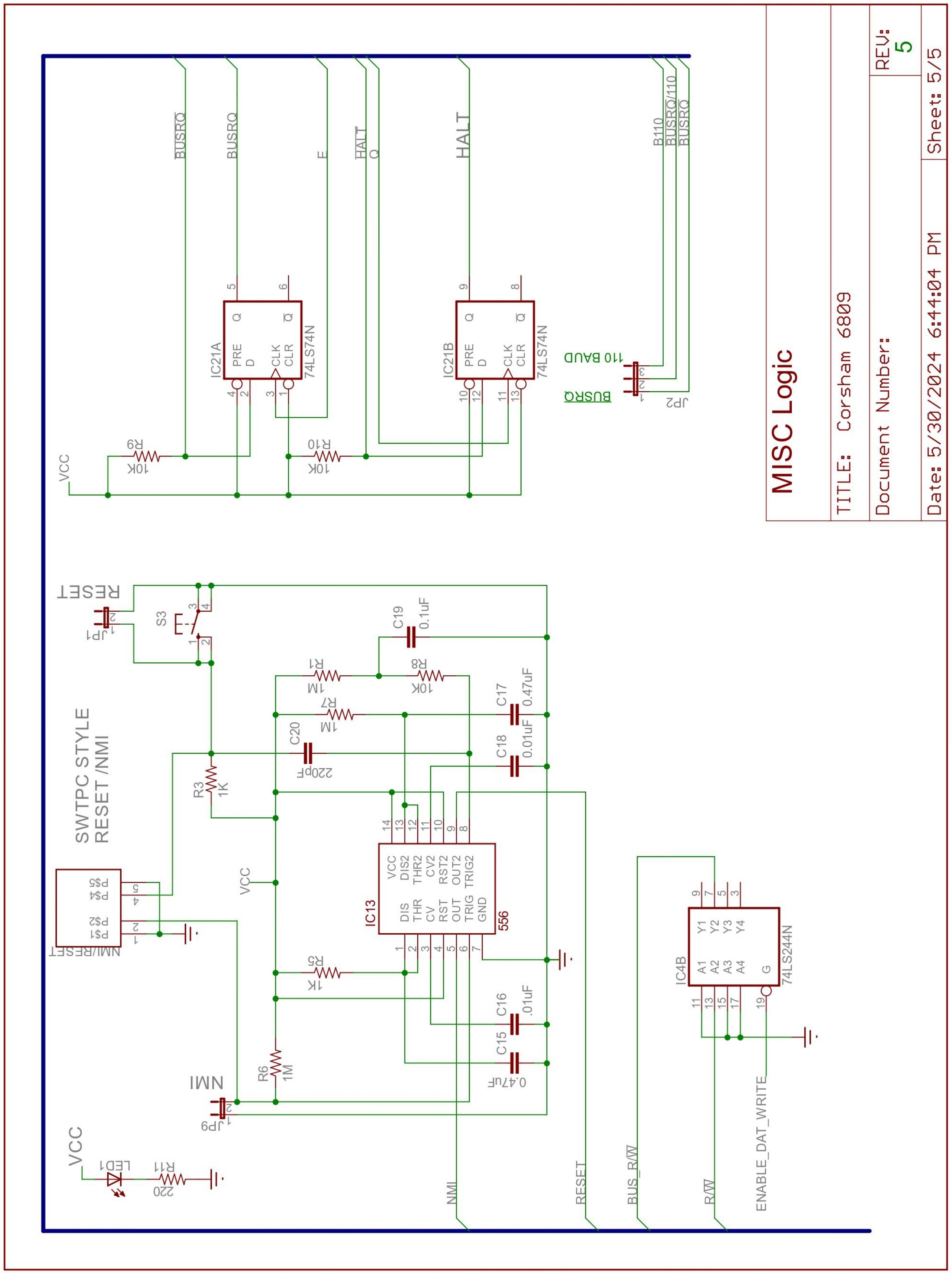

CPU Logic
Corsham Technologies, LLC
TITLE: Corsham 6809
Document Number:
REV: 5
Date: 5/30/2024 6:44:04 PM
Sheet: 1/5
IC1 MC68B09P
IC2A 74LS244N
IC2B 74LS244N
IC3A 74LS244N
IC3B 74LS244N
IC4A 74LS244N
IC5A 74LS240N
IC5B 74LS240N
IC9 SN74LS640N
IC17B 74LS00N
IC17C 74LS00N
X2 8 MHZ
C7 22pf
C14 22pf
C21 100 pf
R12 470
R14 6.8K
ON_BOARD
VCC



On-Board Memory
Corsham Technologies
TITLE: Corsham 6809
Document Number:
REV: 5
Date: 5/30/2024 6:44:04 PM
Sheet: 2/5
IC7 28C64
IC6 628128
IC8 22V10
JP3
JP7
JP8
R15 1K
R2 1K
R13 1K
R16 1K
IC15B 74LS02N
IC15D 74LS02N
IC15C 74LS02N
IC17D 74LS00N
VCC
ON_BOARD
EPROM
RAM
TOP_PAGE
R/W


Power Supply & Baud Rate Generator
TITLE: Corsham 6809
Document Number:
REV: 5
Date: 5/30/2024 6:44:04 PM
Sheet: 3/5
QG1
2.4576MHZ
IC16
4040N
JP6
JP10
JP11
JP12
JP13
S0
S1
S2
S3
IC11A
IC11B
74LS240N
B150/S3
B300/S2
B600/S1
B1200/S0
B110
76800
38400
19200
9600
4800
2400
1200
600
300
150
J1
VR1
7805T
C1
100uF 16V
C2
C3
C4
C5
C6
C8
C9
C10
C11
C12
C13
C22
C23
C24
0.1uF
VCC
GND
IC1P
IC2P
IC3P
IC4P
IC5P
IC7P
IC8P
IC11P
IC12P
IC14P
IC15P
IC16P
IC17P
IC21P
RESET
HALT
IRQ
FIRQ
BUSY
E
VMA
BA
BS
Q
MRDY
BUS_R/W
BUSRQ/110
-16V
+16V
+8V
INDEX
D0
D1
D2
D3
D4
D5
D6
D7
BA0
BA1
BA2
BA3
BA4
BA5
BA6
BA7
BA8
BA9
BA10
BA11
BA12
BA13
BA14
BA15


Dynamic Address Translation
TITLE: Corsham 6809
Document Number:
REV: 5
Date: 5/30/2024 6:44:04 PM
Sheet: 4/5
IC18
74S189
IC19
74S189
IC12
74LS157N
IC17A
74LS00N
IC15A
74LS02N
IC14
74LS30N
D0 TA12 D2 TA14 D3 TA15 D1 TA13
D4 S0 D5 S1 D6 S2 D7 S3
VCC
DAT_WRITE
ENABLE_DAT_WRITE
TOP_PAGE
A12 A0 A14 A2 A15 A3 A13 A1
E
BA8 BA10 BA9 BA11 A12 A13 A14 A15 R/W


MISC Logic
TITLE: Corsham 6809
Document Number:
REV: 5
Date: 5/30/2024 6:44:04 PM
Sheet: 5/5
SWTPC STYLE RESET /NMI
RESET
NMI
VCC
BUSRQ
E
HALT
Q
B110
BUSRQ/110
110 BAUD
JP2
IC21A
IC21B
74LS74N
IC13
556
IC4B
74LS244N
R9 10K
R10 10K
R1 1M
R8 10K
R7 1M
R3 1K
R5 1K
R6 1M
R11 220
C20 220pF
C19 0.1uF
C17 0.47uF
C18 0.01uF
C16 .01uF
C15 0.47uF
LED1
S3
JP1
JP9
NMI/RESET
ENABLE_DAT_WRITE
BUS_R/W
R/W