MOTOROLA
Semiconductor Products Inc.

AN896A
Application Note

SERIAL I/O, TIMER, AND INTERFACE CAPABILITIES OF THE MC68901 MULTIFUNCTION PERIPHERAL

Prepared by
Geoff Brown
System Applications Engineer
Motorola, Inc.
Austin, Texas

INTRODUCTION

This application note illustrates a system which utilizes several functions of the MC68901 Multifunction Peripheral (MFP). The utilized functions include: 1) USART serial I/O, 2) utilization of internal timers to generate the serial I/O baud rate, 3) utilization of internal timers to generate external interrupts, and 4) use of general purpose I/O pins to provide a cassette interface.

Other general control signal connections are also illustrated. These include: system clock, R/W, IRQ, DTACK, IACK, DS, CS, and timer clock external connections (XTAL1 and XTAL2).

A schematic diagram of the actual hardware used in this application note is shown in Figure 1 at the end of this document. In addition to Figure 1, a listing of the software used with the application is also provided at the end of this document.

HARDWARE CONSIDERATIONS

The hardware shown in Figure 1 uses the MC68008 microprocessor unit (MPU) to control the system; that is, address, data, function codes, data and address strobes, etc. The MC68901 MFP then provides the interrupt and interrupt vectors for the MPU. Eight MCM6665 RAM devices are used to demonstrate the requirement for refresh timing (RAS and CAS). The ROM is implemented in EPROM. Miscellaneous glue parts then tie the system together.

ADDRESS DECODING

Because the addressing range of the MC68008 far exceeds the needs of this application, it is possible to use a simple address decoding scheme. An SN74LS138 3-to-8 demultiplexer (U19) is used to divide the address map into eight 128K segments. Three of these eight segments are assigned to RAM, the MC68901 MFP, and ROM respectively. RAM begins at $00000, MC68901 MFP begins at $20000, and ROM (EPROM) begins at $A0000. The other five segment select control lines are available for expansion.

One problem associated with placing system ROM at any segment other than the bottom of memory is that the MC68008 looks at location $00000 for its reset vector; however, it is impractical to place ROM at the bottom of the memory map because this would prohibit dynamic interrupt vector programming. This can be resolved by mapping the ROM to the lower portion of memory at reset. In this application, an SN74LS164 shift register (U18) is used to force selection of ROM for the first eight memory cycles after reset to allow the processor to fetch the reset vector and supervisor stack pointer from ROM. When QH of the SN74LS164 shift register is low, selection of ROM is automatic and selection of RAM is inhibited. Once QH goes high, selection proceeds in a normal fashion. U18 is reset whenever HALT and RESET are both active (the system reset condition). Once RESET or HALT become inactive, a logic one is shifted into U18 by the rising edge of AS. After eight memory cycles QH goes high and ROM returns to its normal location in the memory map.

RAM CONTROLS

A second SN74LS164 (U17) is used to generate the RAS, CAS, MUX, and DTACK signals. The RAS, CAS, and MUX signals provide control of the dynamic RAM, and DTACK is applied to the MPU to indicate access to the RAM and ROM. Shift register U17 is inhibited from shifting by IACK cycles and by memory cycles to the MC68901. For all other memory cycles, the shift register is allowed to shift and generate DTACK. Notice that DTACK is automatically generated for all areas of memory other than that assigned to the MC68901 and that only one $\overline{\text{DTACK}}$ time is generated (500 nanoseconds after $\overline{\text{AS}}$). System performance could be improved by optimizing dynamic RAM sequencing and $\overline{\text{DTACK}}$ generation. $\overline{\text{RAS}}$ is generated for all memory cycles while $\overline{\text{CAS}}$ is enabled by selection of RAM. By generating $\overline{\text{RAS}}$ for all memory cycles it is possible to refresh RAM by executing instructions out of ROM (software refresh). Address multiplexing for the dynamic RAM is accomplished with two SN74LS157 two-input multiplexers (U1 and U2).

MC68008/MC68901 INTERFACE

Interfacing the MC68901 is fairly simple. $\overline{\text{RESET}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, and D0-D7 on the MC68901 connect directly to the corresponding pins on the MC68008. RS1-RS5 on the MC68901 connect to the A1-A5 pins on the MC68008. Chip select ($\overline{\text{CS}}$) is generated by qualifying the memory segment signal from U19 with $\overline{\text{AS}}$. $\overline{\text{DTACK}}$ is gated with the QD output from U17 and passed to the MC68008. The preceeding signals are the only ones that are required for interfacing the MPU with the MFP. In addition, this application utilizes the interrupt capability of the MC68901. The $\overline{\text{IRQ}}$ line of the MC68901 is connected directly to both of the MC68008 $\overline{\text{IPL}}$ pins. This corresponds to a level seven interrupt (a nonmaskable interrupt; NMI). Because this application uses the MC68901 to time dynamic refresh intervals, it is imperative that the $\overline{\text{IRQ}}$ interrupt be of the highest priority. If the interrupt capabilities of the MC68901 are to be more fully exploited it is important that no interrupt level be implemented that is higher than the one used for software refresh. The user must never disable or mask the refresh interrupt as this will result in the loss of data. $\overline{\text{IACK}}$ for the MC68901 is generated when the three function codes (FC2-FC0) and A3, A2, and A1 are all high.

For the purpose of baud rate generation, a 2.4576 MHz crystal is connected to the MC68901. Timer C (TCO) is externally connected to the receiver clock (RC) and timer D (TDO) is externally connected to the transmitter clock (TC). Although the software included with this application assumes that the receiver and transmitter clocks operate at the same frequency, the MFP allows for separate clocks.

RESET AND TIMING

The MC68008 requires that an external reset must be applied for at least 100 milliseconds to allow stabilization of the on-chip circuitry and system clock. In this application, system reset is caused at powerup by an MC1455 timer circuit output or it can be generated via a debounced switch. The outputs of the timer and the switch are buffered by open-collector drivers (U27) the outputs of which are connected to $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$.

System timing is provided by a 16 MHz oscillator (U20) which is divided by the two flip flops of U21 to provide 8 MHz (CLK8) and 4 MHz (CLK4) on-chip clocks. The 4 MHz clock is used only by the MC68901 which does not require that its clock be of the same frequency or phase as the system clock.

CASSETTE INTERFACE

Two general purpose I/O lines of the MC68901 (I5 and I6) are used for the cassette interface. Data is transmitted and received as square waves. The length of a single cycle of the square wave determines whether a "1" or a "0" is being transferred.

Data for the cassette interface is output at I6 of the MFP. This output drives a resistor network which divides the voltage by approximately 10. The cassette data output line is then connected to the microphone input of a cassette recorder.

Data to be received from the cassette tape player is shaped in a comparator, U30A. Two IN914 diodes limit the voltage swing to the input of the comparator. The second comparator (U30B) is used to invert the output of U30A. Inversion may or may not be needed depending on whether or not the cassette plays back an inverted signal. The software in this application note assumes that the signal is not inverted. Comparator U30A provides one level of inversion so if the cassette tape player does not provide a level of inversion then a second one must be provided by U30B. The output of comparator U30A is connected to I5 of the MFP (unless U30B is needed).

SOFTWARE

There are six basic software routines included with this application note: MC68901 initialization, software dynamic RAM refresh, transmit character to and receive character from the serial port, transmit character to and receive character from cassette tape. This software represents the basic core of hardware dependent routines necessary for this system.

MC68901 INITIALIZATION

Initialization of the MC68901 consists of starting the serial communication clocks, loading the USART control register, and enabling the refresh clock interrupt. Timers C and D are used for serial receiver and transmitter clocks. In this application both timers are programmed for 9600 baud operation. The 2.4576 MHz reference clock is divided by 16 by loading $02 into both data registers C and D and by starting timers C and D in the divide-by-4 mode. The USART control register is initialized to operate in the divide-by-16 mode (2.4576 MHz/16*16 = 9600 baud). In addition, the proper serial communications protocol must be loaded into the USART control register. In this case the USART is programmed for asynchronous communication with: 1 start bit, 1½ stop bits, and odd parity.

In order to facilitate software refresh of dynamic RAM, the MC68901 interrupt vector is initialized and the timer B interrupt enable and mask bits are set. The timer B output serves as the refresh clock.

SOFTWARE REFRESH

Software refresh consists of accessing 128 consecutive memory locations at regularly timed intervals. In this case, it is accomplished by executing 64 NOP instructions of which each requires two memory fetches. The software refresh program is written as a subroutine which may be called at any time to force a refresh. The refresh subroutine resets timer B (the refresh clock) and executes 64 NOP instructions. Timer B is programmed to generate interrupts every 2 milliseconds. The interrupt routine consists simply of a call to the refresh subroutine. One of the main concerns with software refresh is that programs that have critical timing loops (for example the cassette tape interface routines) could be interrupted for refresh if care were not taken. In order to avoid problems, the refresh routine is written so that an interrupt may be forced before a critical timing loop. The user may then be certain that an interrupt will not occur for at least 1.8 milliseconds. A call to the refresh subroutine should be included in any reset routine in order to preclude loss of data.

SERIAL I/O

Both the receive and transmit routines check for break by reading a bit in the receiver status register. If a break is received at any time during serial communications then a jump to a BREAK character handler routine is made. The exact nature of this subroutine is undefined in this application note but it could consist of transmitting a message and then returning to the user's monitor. The transmit routine also checks for a control-W character and halts if one is received. Transmission is then resumed if any character is received. For serial communications, the divide-by-16 mode (a USART control bit) should be used since it results in increased noise rejection. In order to operate the USART in the divide-by-1 mode the receiver clock must be synchronized externally to the received data.

CASSETTE TAPE INTERFACE SOFTWARE

Data is transmitted to the cassette through GPIP6 (bit 6 of the general purpose input/output port control register) and received through GPIP5 (bit 5 of the general purpose input/output port control register). Data is recorded as a sequence of single cycle square waves with a 500 microsecond period representing a logic one and a one millisecond period representing a logic zero. Before any critical timing loop is executed, in either the transmit or receive routine, a branch to the refresh software is made in order to guarantee that the timing loop will not be interrupted. Timer A of the MC68901 is used for period measurement in both routines. The transmit routine transmits a single byte with the most significant bit first. It is assumed that the first byte of any data stream to be transmitted will be a synchronizing character. In this case the receive routine assumes the synchronizing character to be an ASCII S. The receive routine measures the period length of all incoming square waves in order to generate a bit stream. A simple synchronization routine is included in the program which scans the bit stream for an S. After synchronization data, bytes are assembled from each successive 8-bit block.

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

```
MOTOROLA M68000 ASM VERSION  1.30MFPS    .SA 08/23/84 09:25:40        PAGE   1

                      *
                      *
                      * 68901  I/O ROUTINES INCLUDING:
                      * TRANSMIT CHARACTER THROUGH SERIAL PORT,
                      * RECEIVE CHARACTER FROM SERIAL PORT,
                      * TRANSMIT CHARACTER TO TAPE,
                      * RECEIVE CHARACTER FROM TAPE,
                      * AND SOFTWARE REFRESH FOR RAM.
                      *
                      *
         00001000           ORG     $1000
         00020000     BASE   EQU     $20000    MFP BASE ADDRESS
         00020001     GPIP   EQU     BASE+$01  GENERAL PURPOSE I/O
         00020003     AER    EQU     BASE+$03  ACTIVE EDGE
         00020005     DDR    EQU     BASE+$05  DATA DIRECTION
         00020007     IERA   EQU     BASE+$07  INTERRUPT ENABLE  A
         0002000B     IPRA   EQU     BASE+$0B  INTERRUPT PENDING A
         00020013     IMRA   EQU     BASE+$13  INTERRUPT MASK A
         00020017     VR     EQU     BASE+$17  VECTOR
         00020019     TACR   EQU     BASE+$19  TIMER A CONTROL
         0002001B     TBCR   EQU     BASE+$1B  TIMER B CONTROL
         0002001D     TCDCR  EQU     BASE+$1D  TIMER C/D CONTROL
         0002001F     TADR   EQU     BASE+$1F  TIMER A DATA
         00020021     TBDR   EQU     BASE+$21  TIMER B DATA
         00020023     TCDR   EQU     BASE+$23  TIMER C DATA
         00020025     TDDR   EQU     BASE+$25  TIMER D DATA
         00020029     UCR    EQU     BASE+$29  USART CONTROL
         0002002B     RSR    EQU     BASE+$2B  RECEIVER STATUS
         0002002D     TSR    EQU     BASE+$2D  TRANSMITTER STATUS
         0002002F     UDR    EQU     BASE+$2F  USART DATA
         00000017     CTLW   EQU     $17
                      *
                      *   INITIALIZE 68901
                      *      START TRANSMITTER AND RECEIVER CLOCKS
                      *      FOR 9600 BAUD COMMUNICATION
                      *      LOAD USART CONTROL REGISTER
                      *      INITIALIZE REFRESH INTERRUPT VECTOR
                      *

00001000 13FC00020002 INIT   MOVE.B  #$02,TCDR  1/4 TRANSMITTER CLOCK
         0023
00001008 13FC00020002        MOVE.B  #$02,TDDR  1/4 RECEIVER CLOCK
         0025
00001010 13FC00110002        MOVE.B  #$11,TCDCR DIVIDE BY 4
         001D
00001018 13FC00940002        MOVE.B  #$94,UCR   ODD PARITY,1 1/2 STOP,
         0029
                      *                         1 START, ASYNC, 8 BITS
                      *                         1/16 FOR 9600 BAUD
00001020 13FC00010002        MOVE.B  #$01,RSR   START RECEIVER CLOCK
         002B
00001028 13FC00050002        MOVE.B  #$05,TSR   START TRANSMITTER CLOCK
         002D
```

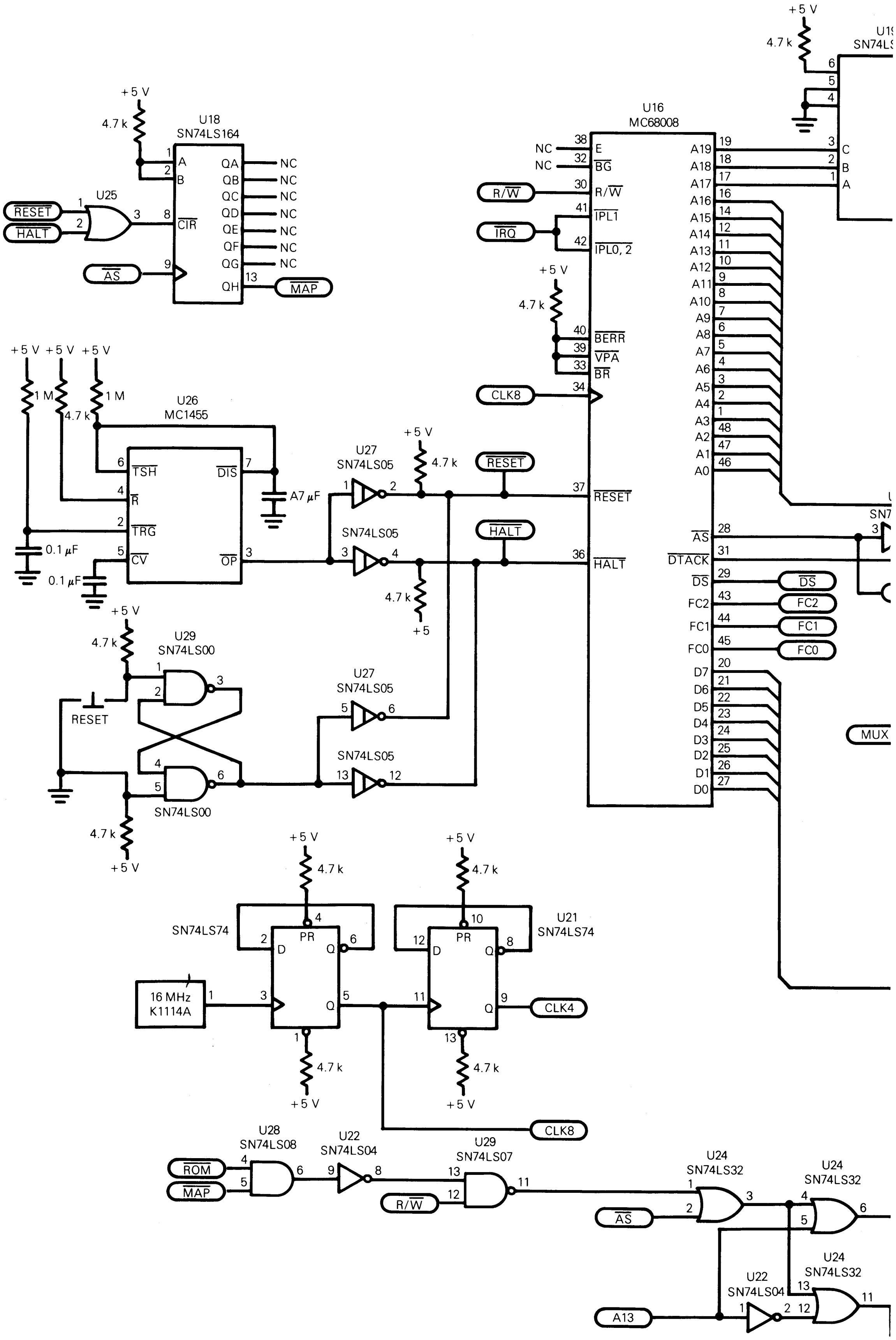
```
MOTOROLA M68000 ASM VERSION  1.30MFPS    .SA 08/23/84 09:25:40        PAGE   2

                      *
                      *      INITIALIZE REFRESH INTERRUPT
                      *
00001030 13FC00C00002        MOVE.B  #$C0,VR    LOAD MFP VECTOR REG
         0017
00001038 21FC000010E8        MOVE.L  #RFR2,$320 LOAD INT VECTOR
         0320
00001040 08F900000002        BSET.B  #0,IERA    ENABLE TIMER B INT
         0007
00001048 08F900000002        BSET.B  #0,IMRA    SET MASK BIT
         0013
                      *
                      *      REFRESH SUBROUTINE TO ALLOW SOFTWARE
                      *      TO FORCE AN EARLY REFRESH
                      *
00001050 423900020018 REFRESH CLR.B   TBCR      STOP TIMER B
00001056 13FC00310002        MOVE.B  #49,TBDR   LOAD TIMER B DATA REG
         0021
0000105E 13FC00060002        MOVE.B  #6,TBCR    START TIMER B 1/100
         001B
         00004E71     NOP    EQU     $4E71
00001066 00004E71     RF1    DCB.W   64,NOP     64 NOPs
000010E6 4E75                RTS
000010E8 6100FF7C     RFR2   BSR RF1            INTERRUPT HANDLER
000010EC 4E73                RTE                FOR REFRESH

```

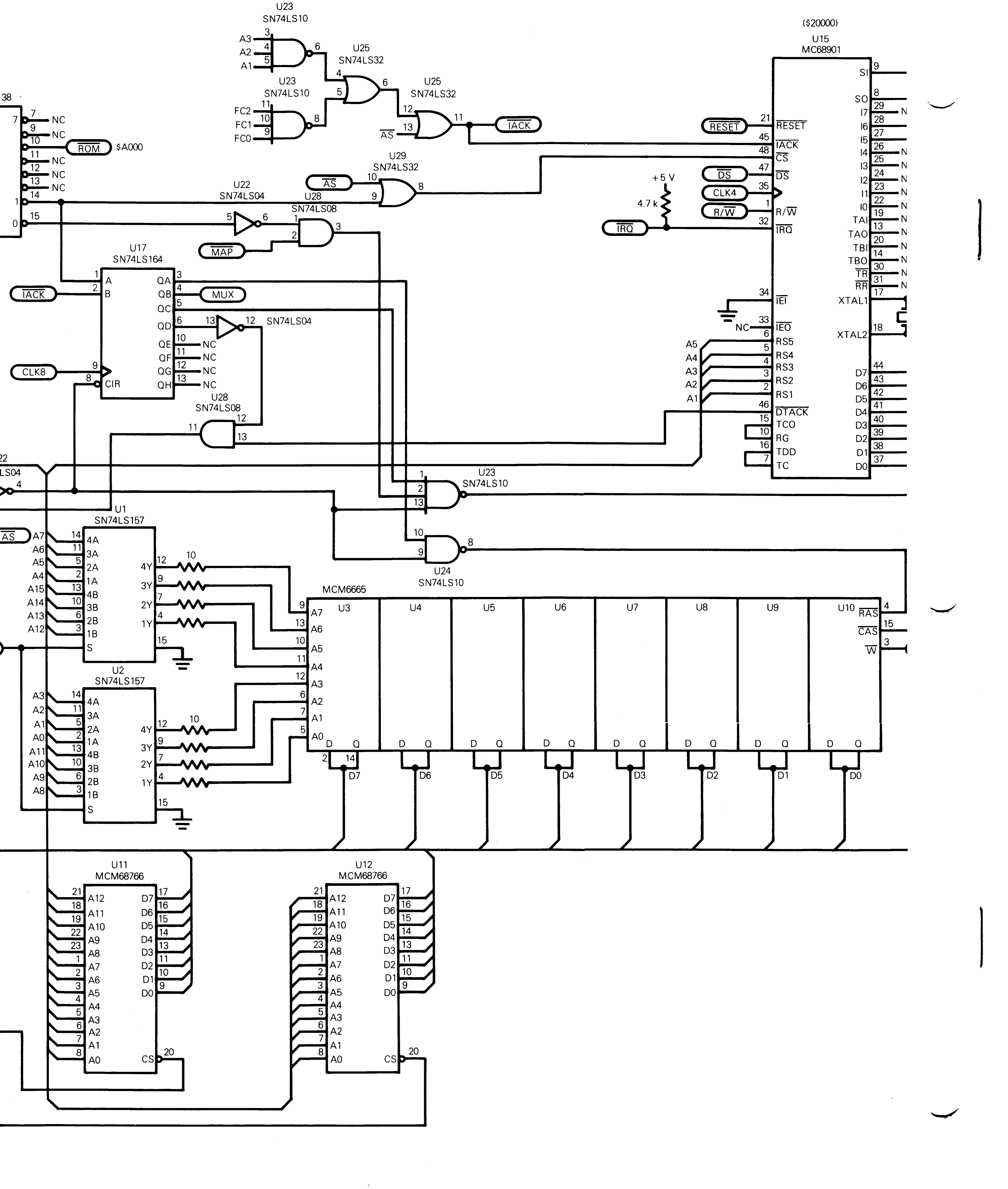
```
 72                             *
 73                             *      INPUT CHARACTER FROM SERIAL PORT INTO D0
 74                             *
 75   000010EE 083900030002 INCHNE BTST.B  #3,RSR      (INCH NO ECHO)
               002B
 76                             *                      CHECK FOR BREAK
 77   000010F6 6654                    BNE.S   BREAK       GO PROCESS IT
 78   000010F8 083900070002            BTST.B  #7,RSR      CHECK FOR CHARACTER
               002B
 79   00001100 67EC                    BEQ.S   INCHNE      IF NOT READY
 80   00001102 10390002002F            MOVE.B  UDR,D0      READ DATA SIDE
 81   00001108 4E75                    RTS
 82                             *
 83                             *      SEND CHARACTER IN D0.B TO SERIAL PORT
 84                             *
 85   0000110A 6134             OUTCH  BSR.S   CHKBRK      CHECK FOR BREAK
 86   0000110C 083900070002            BTST.B  #7,TSR      BUFFER EMPTY
               002D
 87   00001114 67F4                    BEQ.S   OUTCH       STILL NOT READY
 88   00001116 13C00002002F            MOVE.B  D0,UDR      SEND CHARACTER
 89                             *
 90                             *      CHECK FOR CONTROL W
 91                             *
 92   0000111C 083900070002            BTST.B  #7,RSR      READ STATUS
               002B
 93   00001124 6718                    BEQ.S   CTLW9       CHAR NOT READY
 94   00001126 12390002002F            MOVE.B  UDR,D1      READ CHARACTER
 95   0000112C 0C010017                CMP.B   #CTLW,D1
 96   00001130 660C                    BNE.S   CTLW9       NOT CNTL/W
 97   00001132 610C             CTLWH  BSR.S   CHKBRK      CHECK FOR BREAK
 98   00001134 083900070002            BTST.B  #7,RSR      READ STATUS
               002B
 99   0000113C 67F4                    BEQ     CTLWH       WAIT FOR ANY CHAR
100                             *                          TO CONTINUE
101   0000113E 4E75             CTLW9  RTS
102                             *
103                             *      CHECK FOR BREAK ON SERIAL PORT
104                             *
105   00001140 083900030002 CHKBRK BTST.B  #3,RSR      READ STATUS
               002B
106   00001148 6602                    BNE.S   BREAK
107   0000114A 4E75                    RTS
108                             *
109                             *      WHAT TO DO WHEN THE BREAK IS PRESSED
110                             *
111                             *
112   0000114C 083900070002 BREAK  BTST.B  #7,TSR      CHECK "TRANSMIT READY"
               002D
113   00001154 67F6                    BEQ.S   BREAK       WAIT FOR READY
114   00001156 10390002002F            MOVE.B  UDR,D0      READ CHARACTER
115   0000115C 083900030002            BTST.B  #3,RSR      BREAK BUTTON RELEASED?
               002B
116   00001164 66E6                    BNE     BREAK       NO... KEEP LOOPING
117                             *
118                             *      USER SHOULD INSERT BREAK HANDLER HERE
119                             *
120   00001166 4E75                    RTS
```

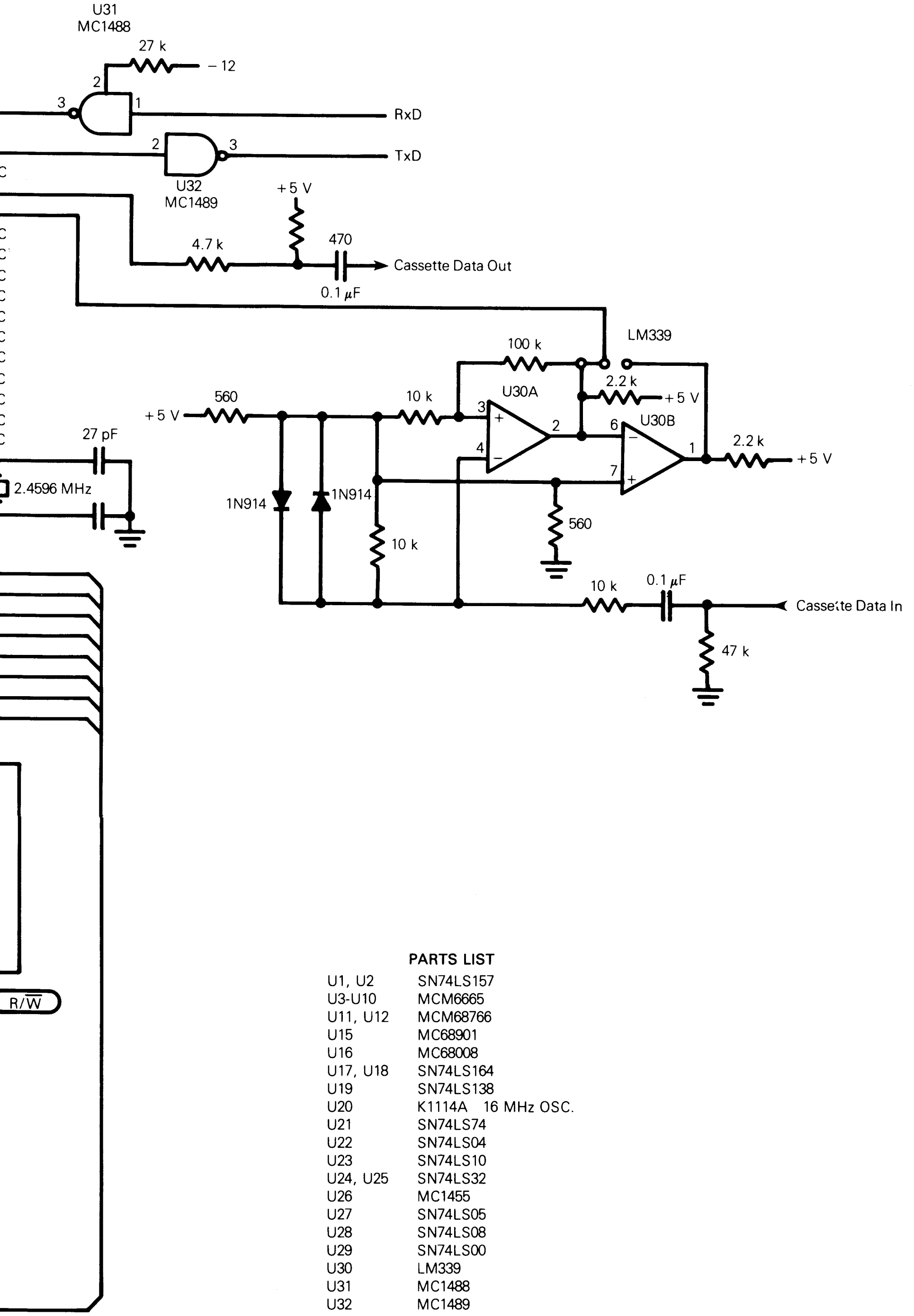
```
122                             *
123                             *      TRANSMIT CHARACTER IN D2 TO TAPE
124                             *
125                             *      A LOGIC '0' IS RECORDED AS ONE SQUARE WAVE
126                             *      PERIOD OF 1 MILLISECOND DURATION.  A LOGIC
127                             *      '1' IS RECORDED AS ONE SQUARE WAVE PERIOD
128                             *      OF 500 MICROSECOND DURATION.
129                             *
130   00001168 08F900060002 TAPE0  BSET.B  #6,DDR      SET GPIP6 AS OUTPUT
               0005
131   00001170 08F900050002        BSET.B  #5,IERA     ENABLE TIMER A INTERRUPT
               0007
132   00001178 103C0001            MOVE.B  #1,D0       STOP BIT INTO D0
133   0000117C E31A         TAPE01 ROL.B   #1,D2       DATA BIT INTO D2
134   0000117E 6100FED0            BSR     REFRESH     FORCE REFRESH
135   00001182 614C                BSR.S   TTST        WAIT UNTIL PULSE DONE
136   00001184 13FC00000002        MOVE.B  #$00,TACR   HALT TIMER A
               0019
137   0000118C 123C000A            MOVE.B  #10,D1      TIMER COUNT FOR 1
138
139   00001190 08020000            BTST.L  #0,D2       SENDING 1?
140   00001194 6606                BNE.S   TAPE02      YES
141   00001196 06810000000A        ADDI.L  #10,D1      NO. TIMER COUNT FOR 0
142   0000119C 13C10002001F TAPE02 MOVE.B  D1,TADR     SET TIMER PRELOAD
143   000011A2 08F900060002        BSET.B  #6,GPIP     SEND 1 TO TAPE
               0001
144   000011AA 13FC00050002        MOVE.B  #$05,TACR   START TIMER A 1/64
               0019
145   000011B2 611C                BSR.S   TTST        WAIT UNTIL PULSE DONE
146   000011B4 423900020019        CLR.B   TACR        HALT TIMER
147   000011BA 08B900060002        BCLR.B  #6,GPIP     SEND 0 TO TAPE
               0001
148   000011C2 13FC00050002        MOVE.B  #$05,TACR   START TIMER A 1/64
               0019
149   000011CA E300                ASL.B   #1,D0       SENT 8 BITS?
150   000011CC 66AE                BNE     TAPE01      NO, CONTINUE
151   000011CE 4E75                RTS
152                         *
153                         *      TIMER TEST
154                         *
155   000011D0 0C3900000002 TTST   CMP.B   #0,TACR     TIMER RUNNING?
               0019
156   000011D8 6712                BEQ.S   TTST1       NO,RETURN
157   000011DA 083900050002        BTST.B  #5,IPRA     TIME DELAY ELAPSED?
               000B
158   000011E2 67EC                BEQ.S   TTST        NO. WAIT
159   000011E4 08B900050002        BCLR.B  #5,IPRA     CLEAR INTERRUPT
               000B
160   000011EC 4E75         TTST1  RTS
```

```
MOTOROLA M68000 ASM VERSION  1.30MFPS    .SA 08/23/84 09:25:40         PAGE   5

                                 *
                                 *
                                 *     RECEIVE CHARACTER FROM TAPE INTO D0.B
                                 *
00011EE 423900020019 TAPEIN CLR.B   TACR         STOP TIMER A
000011F4 4201                 CLR.B   D1           CLEAR D1 FOR DATA
000011F6 083900050002 T10     BTST.B  #5,GPIP      WAIT FOR LOW
         0001
000011FE 66F6                 BNE     T10
00001200 083900050002 T20     BTST.B  #5,GPIP      WAIT FOR HIGH
         0001
00001208 67F6                 BEQ     T20
                       *
                       *      SYNCHRONIZE ON S CHARACTER
                       *
                       *      THIS ROUTINE LOOKS FOR AN ASCII 'S'
                       *      TO SYNCHRONIZE THE TAPE DATA
                       *
0000120A E301          TS     ASL.B   #1,D1
0000120C 6114                 BSR.S   T30.         GET BIT FROM TAPE
0000120E 0C010053             CMP.B   #'S',D1      S?
00001212 66F6                 BNE.S   TS           NO, CONTINUE
00001214 1CC1                 MOVE.B  D1,(A6)+
                       *
                       *      GET CHARACTER FROM TAPE
                       *
00001216 7202          GC     MOVEQ   #2,D1         SET STOP BIT
00001218 6108          GC10   BSR.S   T30           GET BIT FROM TAPE
0000121A E301                 ASL.B   #1,D1         STOP IN CARRY?
0000121C 64FA                 BCC.S   GC10          NO
0000121E 6102                 BSR.S   T30           GET LAST BIT
00001220 4E75                 RTS
00001222 13FC003B0002 T30     MOVE.B  #$3B,TADR     LOAD TIMER PRELOAD
         001F
0000122A 13FC00050002         MOVE.B  #5,TACR       START TIMER IN
         0019
                       *                            DIVIDE BY 64 MODE
00001232 6100FE1C             BSR     REFRESH       FORCE REFRESH
00001236 083900050002 T40     BTST.B  #5,GPIP       WAIT FOR LOW
         0001
0000123E 66F6                 BNE.S   T40
00001240 083900050002 T50     BTST.B  #5,GPIP       WAIT FOR HIGH
         0001
00001248 67F6                 BEQ.S   T50
0000124A 423900020019         CLR.B   TACR          STOP TIMER
                       *
00001250 16390002001F         MOVE.B  TADR,D3       STORE MEASUREMENT
00001256 0C03001F             CMPI.B  #$1F,D3       LOGIC 1?
0000125A 6D02                 BLT.S   T60           NO
0000125C 5201                 ADDQ.B  #1,D1         STORE 1
0000125E 4E75          T60    RTS
                       *
                       *
                       *
                              END

****** TOTAL ERRORS      0--
****** TOTAL WARNINGS    0--
```

```
MOTOROLA M68000 ASM VERSION  1.30MFPS    .SA 08/23/84 09:25:40         PAGE   6

SYMBOL TABLE LISTING

SYMBOL NAME     SECT   VALUE       SYMBOL NAME     SECT   VALUE


AER                    00020003    T20                    00001200
BASE                   00020000    T30                    00001222
BREAK                  0000114C    T40                    00001236
CHKBRK                 00001140    T50                    00001240
CTLW                   00000017    T60                    0000125E
CTLW9                  0000113E    TACR                   00020019
CTLWH                  00001132    TADR                   0002001F
DDR                    00020005    TAPEO                  00001168
GC                     00001216    TAPEO1                 0000117C
GC10                   00001218    TAPEO2                 0000119C
GPIP                   00020001    TAPEIN                 000011EE
IERA                   00020007    TBCR                   0002001B
IMRA                   00020013    TBDR                   00020021
INCHNE                 000010EE    TCDCR                  0002001D
INIT                   00001000    TCDR                   00020023
IPRA                   0002000B    TDDR                   00020025
NOP                    00004E71    TS                     0000120A
OUTCH                  0000110A    TSR                    0002002D
REFRESH                00001050    TTST                   000011D0
RF1                    00001066    TTST1                  000011EC
RFR2                   000010E8    UCR                    00020029
RSR                    0002002B    UDR                    0002002F
T10                    000011F6    VR                     00020017
```


+5 V
4.7 k
U1
SN74L
6
5
4
3
2
1
C
B
A
U16
MC68008
+5 V
4.7 k
U18
SN74LS164
1
2
A
B
QA
QB
QC
QD
QE
QF
QG
QH
NC
NC
NC
NC
NC
NC
NC
U25
RESET
HALT
1
2
3
8
CIR
AS
9
13
MAP
38
32
E
BG
NC
NC
R/W
30
R/W
41
IPL1
IRQ
42
IPL0, 2
+5 V
4.7 k
40
BERR
39
VPA
33
BR
CLK8
34
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
19
18
17
16
14
12
11
10
9
8
7
6
5
4
3
2
1
48
47
46
+5 V
+5 V
+5 V
1 M
4.7 k
1 M
U26
MC1455
6
TSH
4
R
2
TRG
5
CV
7
DIS
3
OP
A7 µF
0.1 µF
0.1 µF
U27
SN74LS05
1
2
+5 V
4.7 k
RESET
37
RESET
SN74LS05
3
4
HALT
36
HALT
4.7 k
+5
28
AS
31
DTACK
29
DS
DS
43
FC2
FC2
44
FC1
FC1
45
FC0
FC0
20
D7
21
D6
22
D5
23
D4
24
D3
25
D2
26
D1
27
D0
MUX
U
SN7
3
+5 V
4.7 k
U29
SN74LS00
1
2
3
RESET
4
5
6
SN74LS00
4.7 k
+5 V
U27
SN74LS05
5
6
SN74LS05
13
12
+5 V
4.7 k
+5 V
4.7 k
4
PR
SN74LS74
2
D
Q
6
10
PR
U21
SN74LS74
12
D
Q
8
16 MHz
K1114A
1
3
Q
5
11
Q
9
CLK4
1
13
4.7 k
4.7 k
+5 V
+5 V
CLK8
U28
SN74LS08
ROM
MAP
4
5
6
U22
SN74LS04
9
8
U29
SN74LS07
13
12
11
R/W
U24
SN74LS32
1
2
3
AS
U24
SN74LS32
4
5
6
U24
SN74LS32
13
12
11
U22
SN74LS04
1
2
A13


U23
SN74LS10
A3
A2
A1
FC2
FC1
FC0
U25
SN74LS32
AS
IACK
U29
SN74LS32
U22
SN74LS04
U28
SN74LS08
MAP
ROM
$A000
NC
U17
SN74LS164
A
B
QA
QB
QC
QD
QE
QF
QG
QH
CIR
MUX
CLK8
SN74LS04
U28
SN74LS08
($20000)
U15
MC68901
RESET
IACK
CS
DS
CLK4
R/W
IRQ
+5 V
4.7 k
IEI
IEO
RS5
RS4
RS3
RS2
RS1
DTACK
TCO
RG
TDD
TC
SI
SO
I7
I6
I5
I4
I3
I2
I1
I0
TAI
TAO
TBI
TBO
TR
RR
XTAL1
XTAL2
D7
D6
D5
D4
D3
D2
D1
D0
A5
A4
A3
A2
A1
U23
SN74LS10
U24
SN74LS10
U1
SN74LS157
A7
A6
A5
A4
A15
A14
A13
A12
4A
3A
2A
1A
4B
3B
2B
1B
S
4Y
3Y
2Y
1Y
10
U2
SN74LS157
A3
A2
A1
A0
A11
A10
A9
A8
MCM6665
U3
U4
U5
U6
U7
U8
U9
U10
A7
A6
A5
A4
A3
A2
A1
A0
RAS
CAS
W
D
Q
D7
D6
D5
D4
D3
D2
D1
D0
U11
MCM68766
U12
MCM68766
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
CS
20



PARTS LIST

U1, U2	SN74LS157
U3-U10	MCM6665
U11, U12	MCM68766
U15	MC68901
U16	MC68008
U17, U18	SN74LS164
U19	SN74LS138
U20	K1114A 16 MHz OSC.
U21	SN74LS74
U22	SN74LS04
U23	SN74LS10
U24, U25	SN74LS32
U26	MC1455
U27	SN74LS05
U28	SN74LS08
U29	SN74LS00
U30	LM339
U31	MC1488
U32	MC1489

▷ = Open Collector Outputs

FIGURE 1 — Schematic Diagram of Hardware Used in This Application Note

MOTOROLA ***Semiconductor Products Inc.***

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

A17711-1 PRINTED IN USA 10-84 IMPERIAL LITHO C25889 18,000

AN896A