DESIGN & APPLICATION OF A TMS_320_20 STAND ALONE RUN BOARD

Designed by Ron Fox

Troubleshooted & Documented by Bob Smith

Supervisor: Dr R. Radzyner (UNSW)

Department of Electrical Engineering
School of Electrical Engineering & Computer Science
University of New South Wales

May 1987

ABSTRACT

As the project stands, it is very close to being finished. The limits have been discovered and now, ways of detouring around these limits can be developed.

- The board is running - as opposed to the beginning,
- test programs, included in the appendix, run everytime all the time,
- The next stage is to apply Ron Fox's program, but as most of the time, times seems to run out when there is more than one person playing a significant part in the development and troubleshooting.

Keywords to keep in mind are:

1. Digital-to-Analogue and Analogue-to-Digital Conversion,
2. Wait States,
3. Filters,
4. Ready and Reset Logic, and
5. Memory timing cycles.

ACKNOWLEDGEMENTS

I would like to acknowlege the following people for the progress of this project:

Dr. Radzyner - For supervising the project over the months of its life,

Ron Fox - For creating the original circuit,

Joe Yiu - For converting the circuit diagram into a printed circuit board layout and debugging some of the hardware,

Tom Millett - For being there to field my questions about the TMS_320_20,

Jeff Skebe - For aiding me in understanding the memory structure with the aid of the Logic State Analyser,

AIMS OF PROJECT

1. To construct a stand alone digital signal processing run board that will be execute a TMS_320_20 program.

2. Test each part of the system for functionality;

3. Apply a software program to generate the Lower Side-Band of an input signal, with carrier frequency of 16 kHz, from an input signal ranging up to 4 kHz.

4. Document the system, in the event of incompletion, for the next person to continue the train of thoughts - perhaps with a later version from the TMS_320 family.

ATTRIBUTES

1. Single board system,

2. On board D/A and A/D,

3. Maximum Input frequency of 19.5 kHz and maximum Output frequency of 19.5 kHz (or 100 kHz upper limit after hardware modification)**

4. Uses latest switched capacitor filters from Reticon that has flat group delay up to 2 times the cut-off frequency,

5. Function of the board can be altered by replacing EPROMs with their corresponding programs,

6. Uses +/-18 volt and +8 volt power supplies.

7. Program reset capability,

** depends on program length and the time between output instructions

CONVENTIONS USED IN REPORT

Throughout this report, the following conventions will be observed:

1. Signal lines will be in upper case, e.g, READY (Ready)

2. Logic levels, whether HIGH (+5 volts) or LOW (0 volts) in state,

3. Active LOW lines will have an asterisk following the signal, e.g, CE* (chip enable)

4. Integrated Circuit numbers will have an underscore ('_') placed at certain points in the number to emphasise the device, e.g, TMS_320_20 to stand for the TMS32020,

5. More siginificant lines will have the higher number associated with it. For instance, in the sequence D15..D0 --> D15 is the Most Significant Bit (MSB) of the data bus; similarly A15 is the MSB compared with A0 being the least of the address bus,

6. Appendices will have 6 as the prefixed number - for instance, Appendix 2 is chapter 6.2 in the contents page,

7. Figure numbers will have locations coded into the number, for example, Figure 3 in chapter 2 will be shown as Figure 2.3.

CONTENTS

1. INTRODUCTION.. 1

1.1 External Program Memory..... 2

1.2 Principle of Operation..... 3

1.3 Operation of Circuit..... 3

2. INPUT - OUTPUT..... 5

2.1 Data Collection..... 5

2.1.1 Input Protection..... 5

2.2 Data Output..... 7

2.3 Buffers and Filters..... 7

2.4 Anti-aliasing and Reconstruction Filters..... 9

2.5 Function of SCFs..... 9

3. CONTROL..... 11

3.1 Hardware..... 11

3.1.1 EPROMs..... 11

3.1.2 Wait States for the TMS_320_20..... 12

3.1.3 Reset Logic..... 14

3.1.4 SCF start up..... 16

3.1.4.1 Loading the Internal Register..... 17

3.2 Software..... 18

3.2.1 Test Programs..... 18

3.2.2 Cycle Timings..... 19

3.2.3 Specific Application..... 19

3.2.4 Eprom Burning..... 19

3.2.5 Using the SWDS..... 20

4. FINDINGS..... 21

4.1 Nuances..... 21

4.1.1 Hardware Reset..... 21
4.1.2 SCF kick off..... 21
4.1.3 Program Memory..... 22
4.2 Problems Encountered..... 22
4.3 Unresolved Questions..... 22
4.4 Modifications..... 23
5. Notes made by Reader..... 24
6. APPENDICES..... 27
6.1 Hardware Information..... 28
6.1.1 Circuit Diagram..... 28
6.1.2 TMS_320_20 pin outs..... 28
6.1.3 I.C. Layout..... 28
6.1.4 A wiring diagram of the system..... 28
6.1.5 SCF data sheet..... 28
6.2 Wait State Determination..... 33
6.3 Testing Programs..... 36
6.4 Current Drain from Power Supplies..... 38
6.5 Instruction Cycle Timings..... 39
6.6 Memory Map Adjustment..... 41
6.7 Burning EPROMs..... 42
6.7.1 PROM_BURN ala 313..... 42
6.7.2 PROM_BURN ala DSL..... 42
6.8 SSB program for the TMS_320_10..... 47

TABLE OF FIGURES

Figure 1.1 System Block Diagram 1

Figure 1.2 TMS_320_20 Memory Map 2

Figure 2.1 Basic A/D conversion stage 6

Figure 2.2 Input Protection Circutry 6

Figure 2.3 Basic D/A conversion stage 7

Figure 2.4 Input & Output Stages 8

Figure 3.1 Controlling Network 12

Figure 3.2 Wait State Realisation 14

Figure 3.3 Designed Reset Circuit 15

Figure 3.4 Diagram of Counters 17

Figure 3.5 Internal Register Input 18

Figure 6.1 System Wiring Diagram 28

Figure 6.2 Reading form EPROM 33

Figure 6.3 Execution of Normal Instructions 34

Figure 6.4 Execution of an Input Instruction 35

Figure 6.5 xflag.lst 36

Figure 6.6 inout2.lst 37

Figure 6.7 Instruction Cycle Timings 39

Figure 6.8 inout.lst cycle timings 40

Figure 6.9 xflag.lst cycle timings 40

Figure 6.10 Memory Map Adjusted 41

1. INTRODUCTION

As this is a stand alone printed circuit board, the heart of it is the Texas Instruments TMS_320_20 Digital Signal Processing (DSP) I.C. This I.C. has to control how the board reads the input signal, processes it and feeds it to the outside world.

Its set of instructions are stored in two EPROMs, side by side on the address bus. Figure 1.1, shows how the TMS_320_20 is connected to the rest of the system.

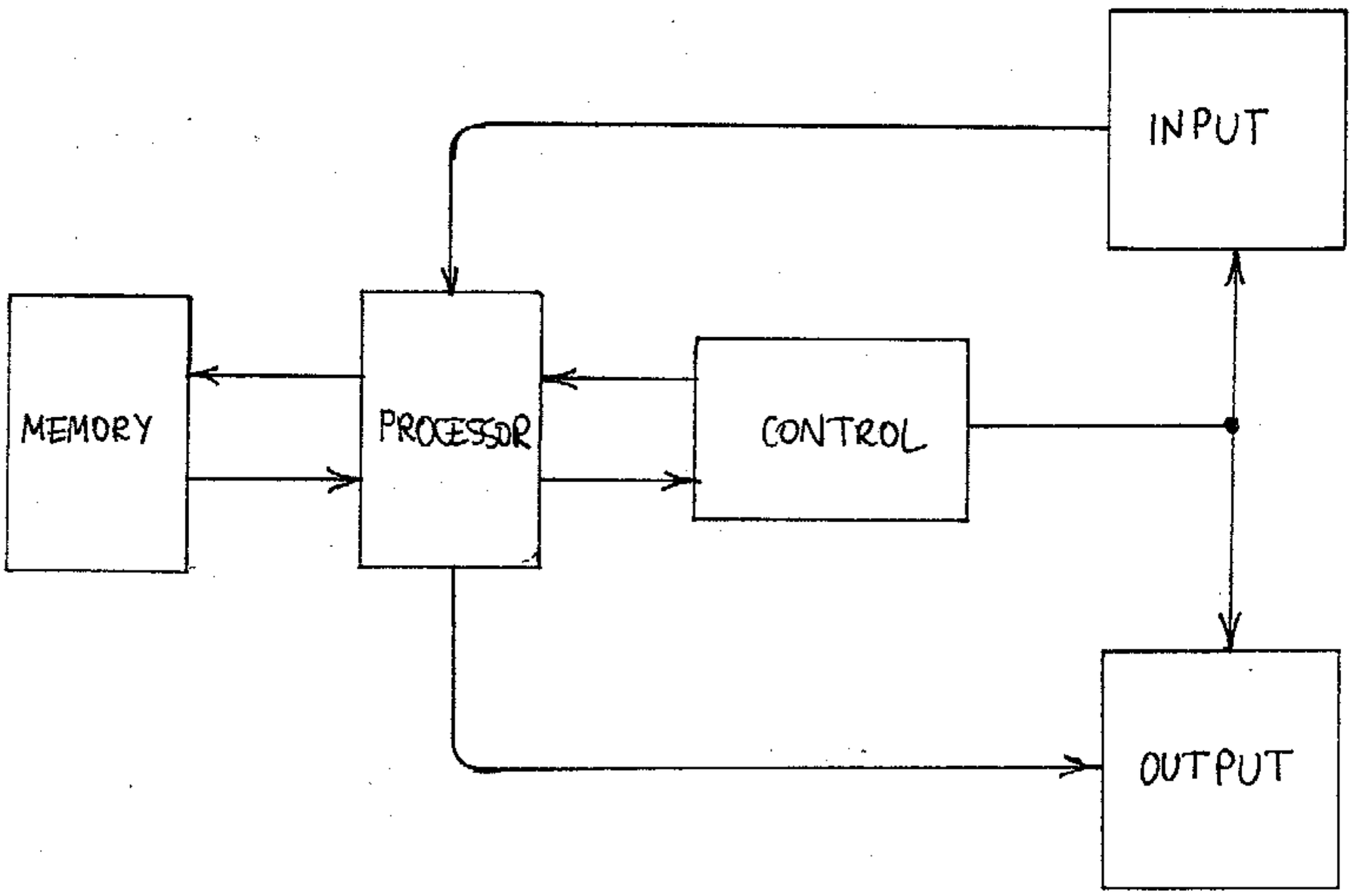


Figure 1.1. System Block Diagram

1.1 External Program Memory

This TMS_320 family member only access its program externally. Unlike the TMS_320_10 with 4 Kbytes of internal program memory, hardware has to be created to handshake with the associated EPROM. Figure 1.2 shows the separation of the Program and Data memories and the Input and Output space.

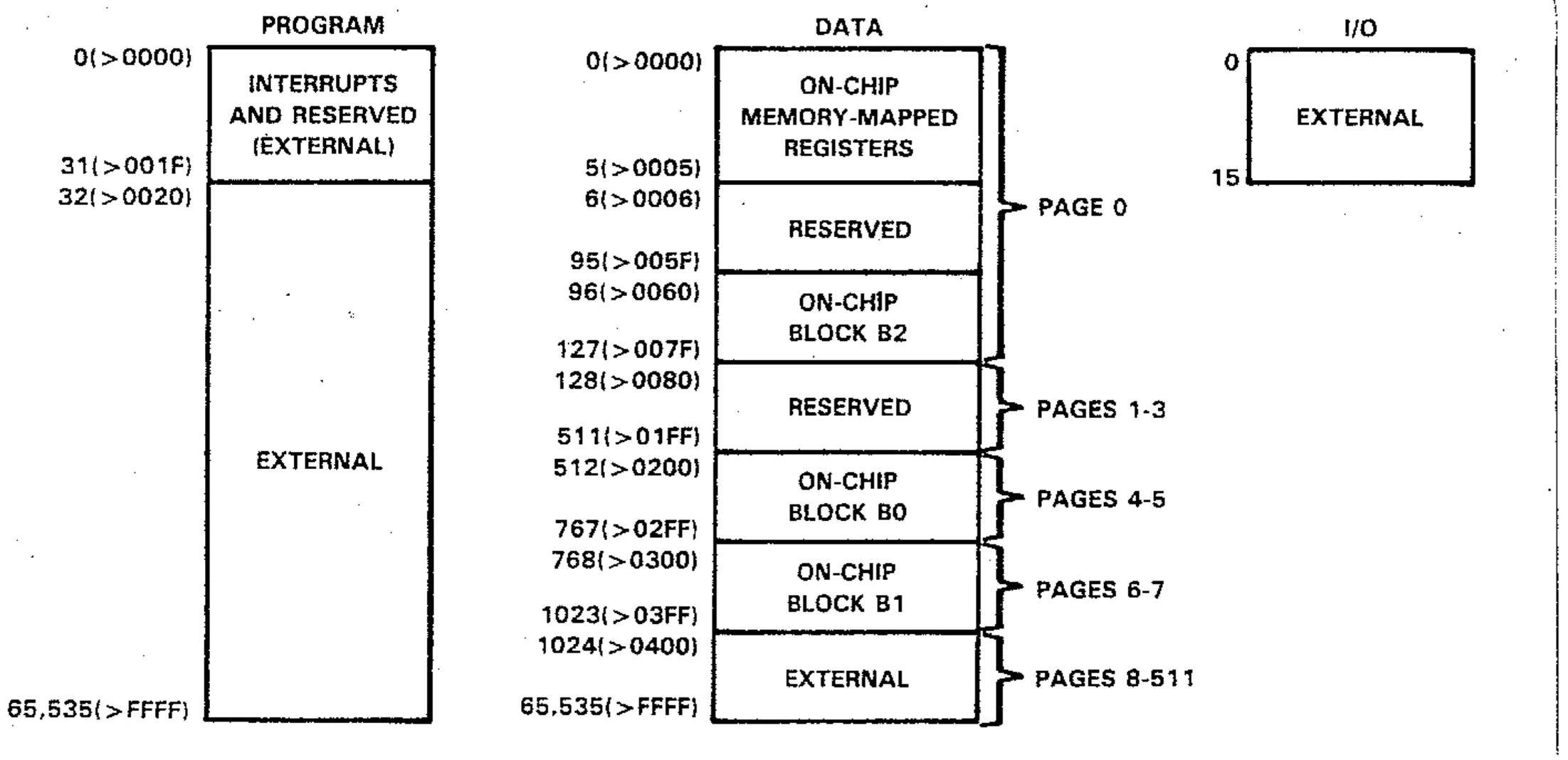


Figure 1.2. TMS_320_20 Memory Map

On the hardware side of the DSP I.C., there are signal lines that are activated in the correct time to access either the program bus or the data bus. Namely, the Program Strobe (PS*) and the Data Stobe (DS*). In addition, there is an Input/Output Strobe (IS*) line to indicate that its space is being accessed during the instruction.

One thing to consider, is the speed compatibility of the I.Cs. In this design, the fastest EPROMs, slower than the TMS_320_20 cycle time, were chosen. At this point, the use of the READY line to indicate to the TMS_320_20 that the information has not been completely accessed, prevents the DSP I.C. from overunning the EPROMs and catching every, say, third instruction.

1.2 Principle of Operation

Irrespective of program coded into the EPROMs, the input signal whether digital or analogue, is sampled, processed and then output. Here, the Switched Capacitor, Anti-aliasing and Reconstruction filters and the Sample-and-Hold circuit sample the input signal.

It is then acted upon by the code in the TMS_320_20 whether to filter digitally, frequency translate or scramble the input sequence of samples. Note, a 16-bit number is the quantity to be manipulated in the TMS_320_20.

Finally, the number is fed out into the Digital-to-Analogue Converter and then through the Anti-aliasing, Switched Capacitor and Reconstruction filters.

1.3 Operation of Circuit

Looking at the schematic diagram in Appendix 1, the circuit operates basically as follows:

a. Reset pressed to initiate the program

b. Program begins

    1. dividers provide trigger signal for Switched Capacitor Filters (SCFs)

2. SCFs start sampling

3. sample is reconstructed by analogue filter after SCF

4. Analogue to Digital Converter (A/D) selected

5. Sample and Hold circuit passes on sample to A/D

6. program processes received sample

7. Digital to Analogue Converter (D/A) selected

8. output processed sample is converted

9. converted sample put through anti-aliasing filter for SCF input towards the outside world,

10. goto step 5, if circuit is collecting or fedding samples as determined by the program correctly because the input samples are continually flooding in at the input of the S/H and the next valid input instruction addresses the A/D,

c. If program locks up - doesn't input or output anymore & a self test pulse is supposedly sent to the TMS_320_20, then goto step b)

2. INPUT - OUTPUT

The first stage explaining the system block diagram (shown in the introduction as Figure 1.1) is by showing how the input signal is converted from an analogue signal into a suitable form for the TMS_320_20 data bus. Similarly, in the opposite direction, the conversion form will also be examined.

Obviously, Digital-to-Analogue (D/A) and Analogue-to-Digital Converters (A/D) must be used. In this system, these 12 bit devices are connected to the upper 12 bits of the TMS_320_20 data bus. Naturally, a Sample-and-Hold (S/H) circuit is the other unit in the pair with the A/D.

2.1 Data Collection

The heart of the input collection hardware is the A/D and the S/H. This pair will take a maximum time of 8 micro-seconds (us), (5us conversion + 3 us acquisition) to convert from analogue to digital form, giving a 125 kHz sampling frequency (1/8us).

As a precautionary measure, input protection is added in the event of too high a voltage being accidentally injected into the system. Figure 2.1 shows the basic components and the A/D system.

2.1.1 Input Protection

Referring to the circuit diagram, in Appendix 1 or Figure 2.2 here, the 1.8 kOhm resistor extracted from the input stage and put before the two diodes, D1 and D2, the gain of the amplifier is still -1. The purpose of this is to

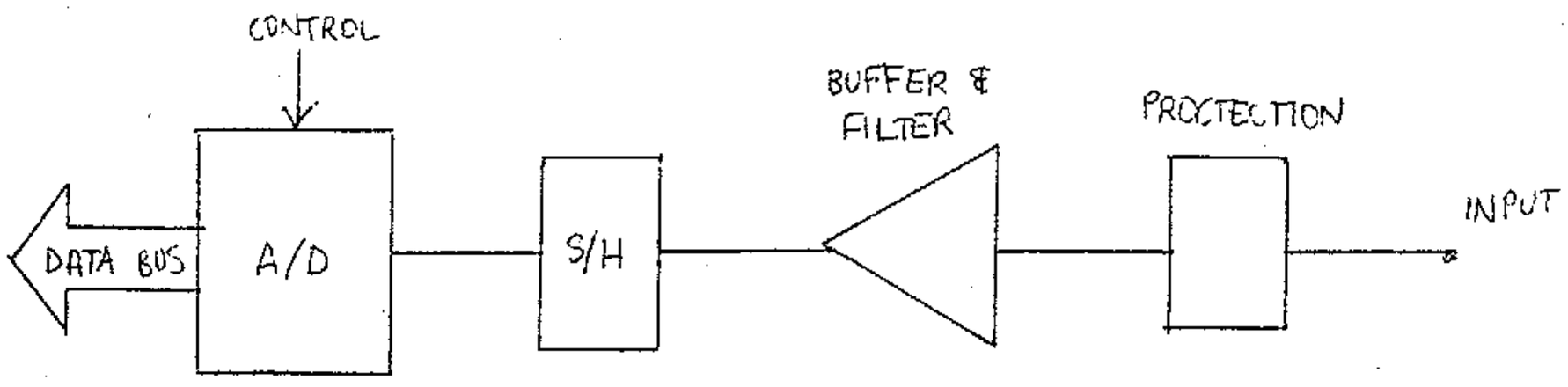


Figure 21. Basic A/D conversion stage

limit the current through either of the diodes.

In the event of a large signal, greater than 10 volts peak-to-peak, it is limited to the +/5 volts rails. Notice that there is a 100 kOhm resistor directly from the input. This makes the input impedance of the pcb, 100 kOhms.

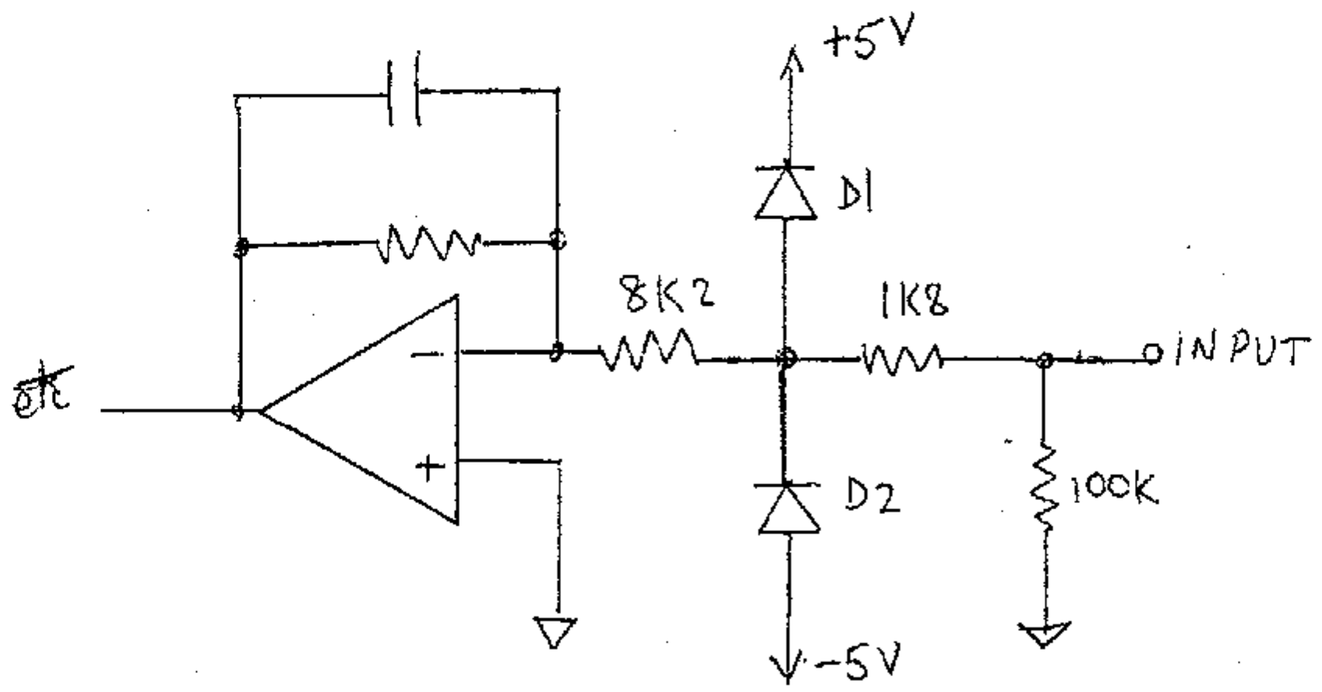


Figure 22. Input Protection Circutry

2.2 Data Output

Again, the heart, of the output stage is the D/A converter. Here, the maximum repsonse time is 8 us (4 us conversion + 4 us settling), giving 125,000 samples per second. But, as limited by the filter stage, shown later, this is limited to 19,500 samples per second. From the circuit diagram, in Appendix 1, output 2 has the potential of suppling this maximum sampling rate.

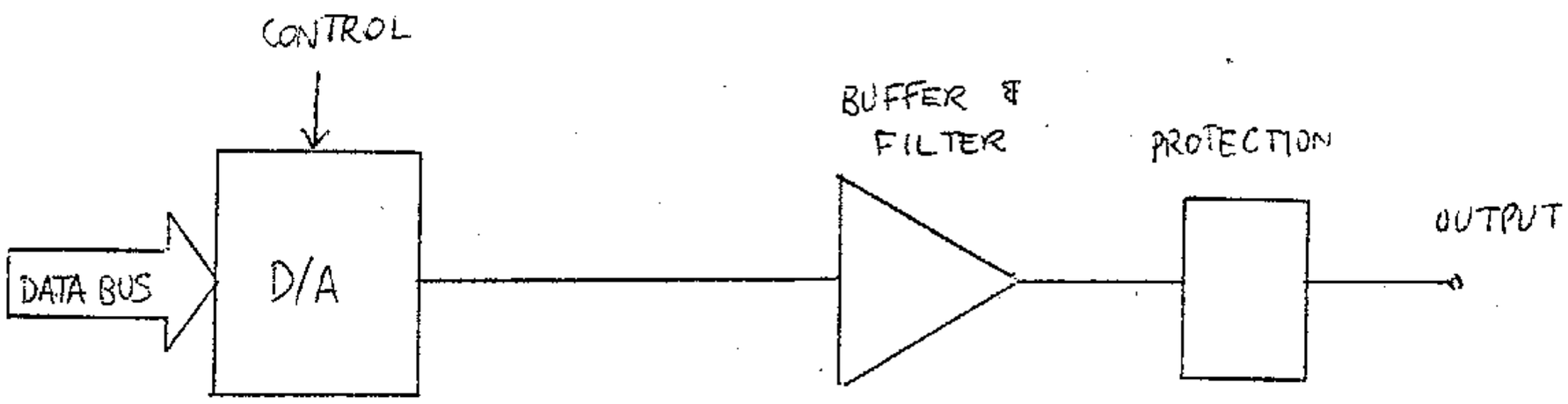


Figure 23. Basic D/A conversion stage

To guard against the output from short circuiting to ground, a resistor is connected in series with the output signal from the D/A. The second function of this resistor is to provide an approximate output impedance of 600 ohms.

2.3 Buffers and Filters

From the following diagram, the 'buffers & filters' block in the earlier conversion stage diagams seem to be a complicated arrangement. Essentially, the Low Pass Switched Capacitor Filter gives rise to the rest of the electronics. The purpose of the (LP)SCF was to be able to set the cut off frequency by having the program feed it the appropriate number on the data bus.

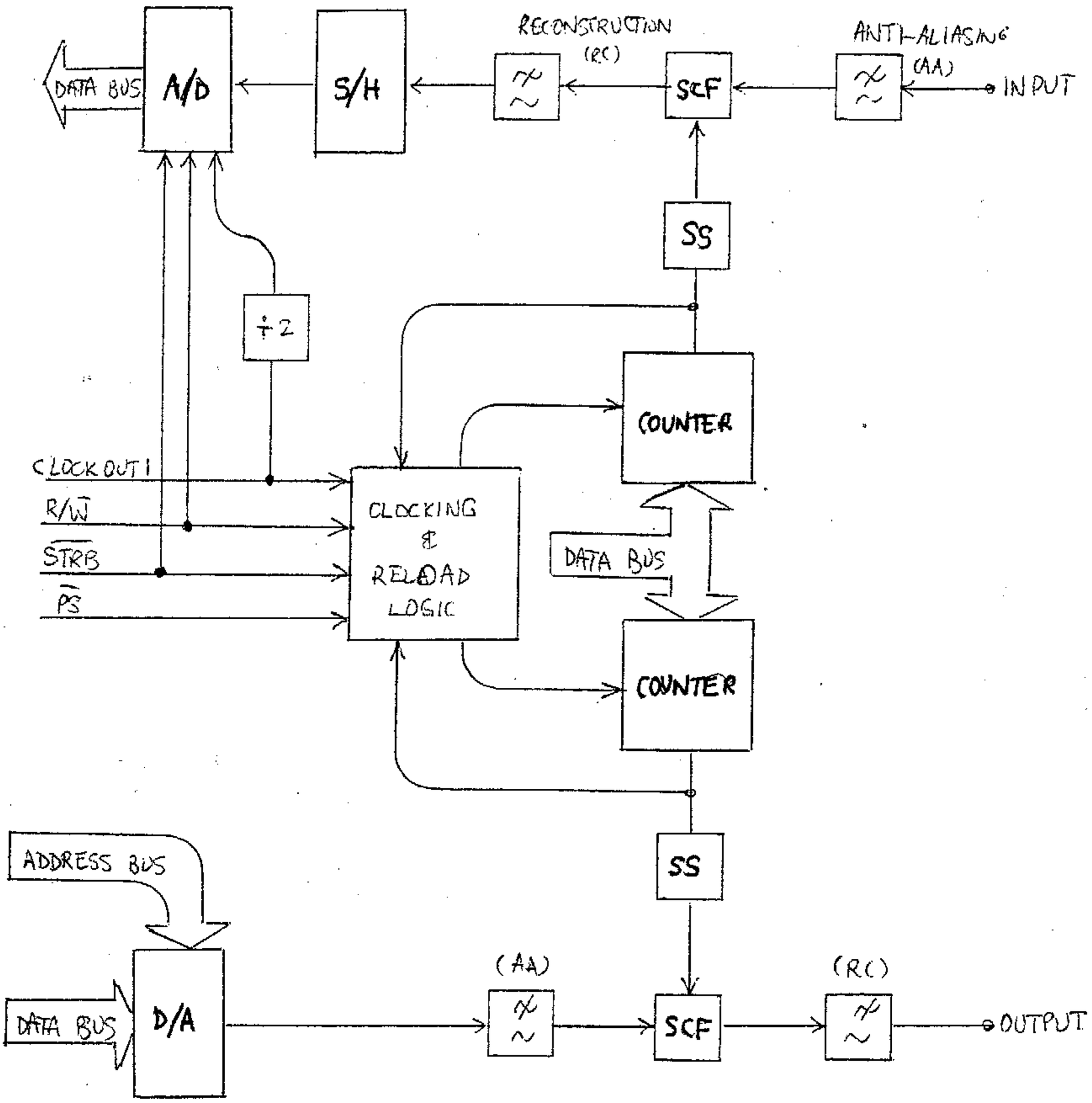


Figure 24. Input & Output Stages

In whatever direction the signal flows, an anti-aliasing filter must be encountered first to limit the higher frequency components, then the SCF itself and finally the filtered signal must be reconstructed. Hence, the reconstruction filter.

The anti-alising and reconstruction filters are simple low pass filters with unity gain where the input pair has a cut of frequency of 10.6 kHz (suitable for audio) and the output pair having 106 kHz cut off. The only restriction is the upper limit of the 19.5 kHz for the SCF as the trigger frequency limit is 2.5 MHz (2500/128 = 19.5). This means that the output (#1, circuit diagram in Appendix 1) is limited to the 19.5 kHz.

If this system is to cater for a 60 kHz output then it can be tapped off from the D/A via a 560 ohm resistor, thus, bypassing the anti-aliasing, switched capacitor and reconstruction filters. However, this maximum frequency of 19.5 kHz applies to the input stage as well - providing the cut-off frequencies for the anti-aliasing and reconstruction filters are greater than this.

2.4 Anti-aliasing and Reconstruction Filters

Feedforward compensation circuit was used for LM301 because it had a fast pulse response time, 1 us, allowing a maximum input frequency of 1 MHz. The two circuits in the LM301 data sheet appeared to have a slower rise time : in the order of 20 us.

If the feedback components in this feedforward configuration are such that R = 10 kOhms and C = 150 pF then the 3 dB frequency is approximately 100 kHz.

2.5 Function of SCFs

To start the SCFs sampling, an initial software instruction has to load the internal register of the binary counter from the data lines going to it. As long as the system clock is changing state, the counter can increment the

number. When counting has finished, the output, fed through some logic, causes it to reload itself with the number latched into the register and proceeds to count again. See Figure 2.4 (earlier) for a frame of reference.

Everytime the counter times out (active low output), a single shot multivibrator is triggered on the negative edge to produce a standard trigger for the SCF. The pulse repetition frequency determines the sampling frequency, hence the 3 dB frequency.

As was highlighted in this chapter, the input and output stages interconnect together very simply. Since this system, at one end, has digital information and at the other end, analogue, both D/A and A/D integrated circuits play a significant role. The next chapter will look at the control of the whole system.

3. CONTROL

As in most micro-computer or micro-processor controlled equipment the hardware and software is totally dependent upon one another and either is of little use without the other. The first half of this chapter on control will deal with the hardware part. It will also deal with the reason behind the strange configurations.

Software, being the other side of the coin, we will expound on the purpose and the use of the programming cornerstones in order to have the system working in unison.

The following figure shows what type of control exists to the various stages. All the lines flowing into the blocks are software controlled from the TMS_320_20.

3.1 Hardware

Strobe & READY lines and wait states are so interwoven with each other in the function of this run board, the following sections will attempt to single out their main contributions.

3.1.1 EPROMs

As mentioned in the introductory chapter under section 1.1, the program memory for the TMS_320_20 has to be external as there is direct no provision for any internal program memory. An advantage of the TMS_320_20 over the TMS_320_10 was availability of handshake lines for access to external program, data and

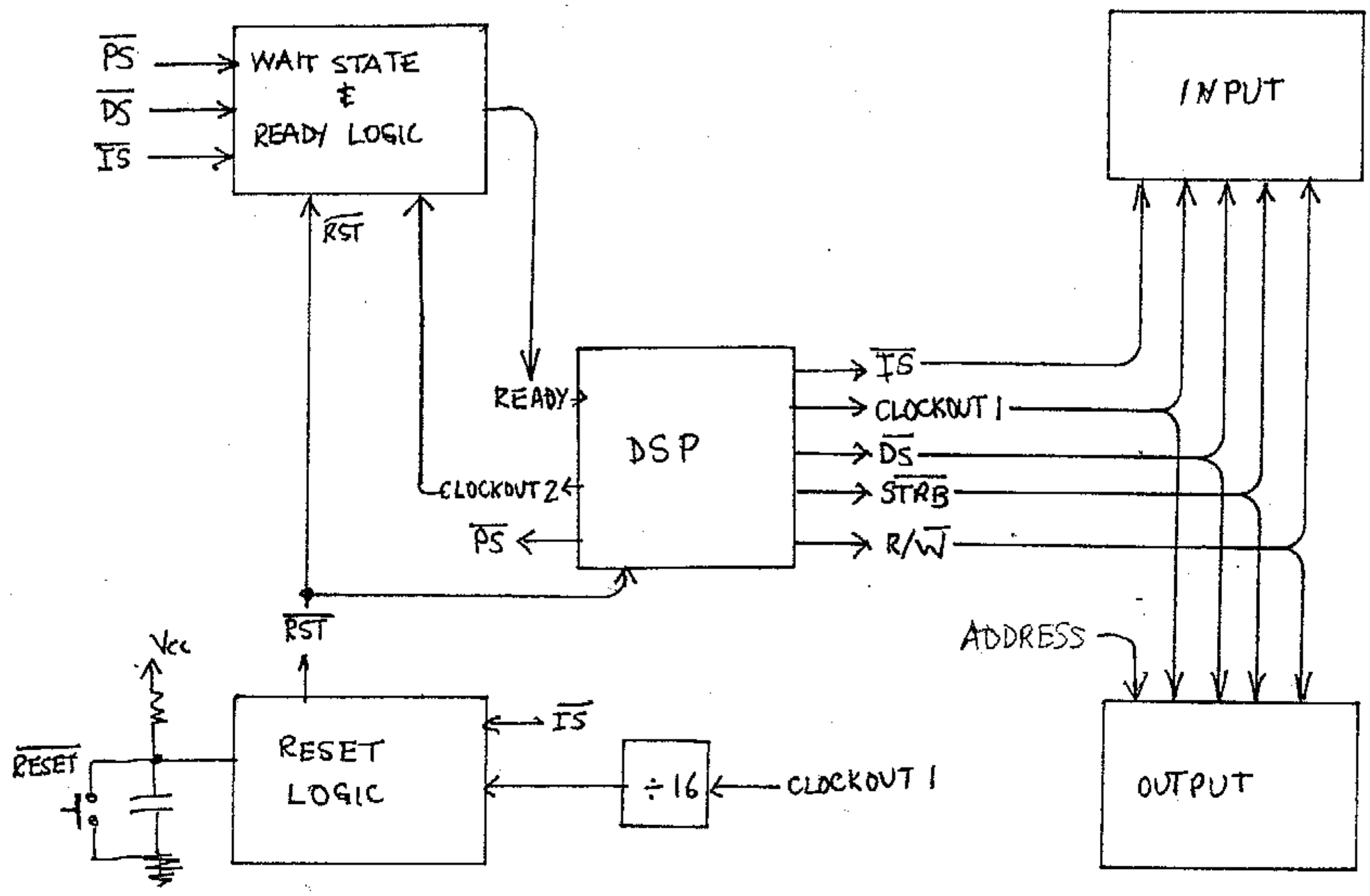


Figure 3.1. Controlling Network

input output memory.

On this board, using the TMS_320_20, the READY line going into the I.C. is checked for a HIGH state, so it can be allowed to fetch the next instruction from the EPROM.

3.1.2 Wait States for the TMS 320 20

The EPROM type chosen to store the program of this run board was the TMS_2764-25. This meant that it would take a maximum of 250 nanoseconds (ns) for data to be valid after the correct select lines were enabled on the EPROM. Hence, the suffix of "-25" on the EPROM number.

Following from this, it will take a significantly longer time to access this EPROM rather than its own internal program memory (256 words of reconfigured data memory). Here, the concept of a wait state was applied. This was possible because the TMS_320_20 had a 'READY' line where it could be used for handshaking with external memory devices.

Working through the timing diagrams of the TMS_320_20 and the TMS_2764-25, it was found that two wait states were needed. That is, a delay of two TMS_320_20 clock cycles (400 ns) before the data from the EPROMs had settled down and the READY line recognised.

With this limit of two wait states, EPROMs as slow as 400 nanoseconds could still be used in the system effectively. On the other hand, if 25 ns EPROMs arrive on the market, there will not be any need for program accessible wait states.

As the TMS_320_20 needed to communicate to other devices such as the D/A and the A/D , there was a further need to use a wait state. Once again working through the data sheets, it was found that one wait state was necessary both for the D/A and the A/D. See Appendix 2 for the relevant calculations and the timing diagrams between the TMS_320_20 and the 2764-25. (Timing for NOP & NOP and IN & NOP instructions).

These wait states were implemented with the use of two standard positive edge triggrered J-K flip flops and a NAND gate. The output of this logic was connected to the READY line of the TMS_320_20 I.C.

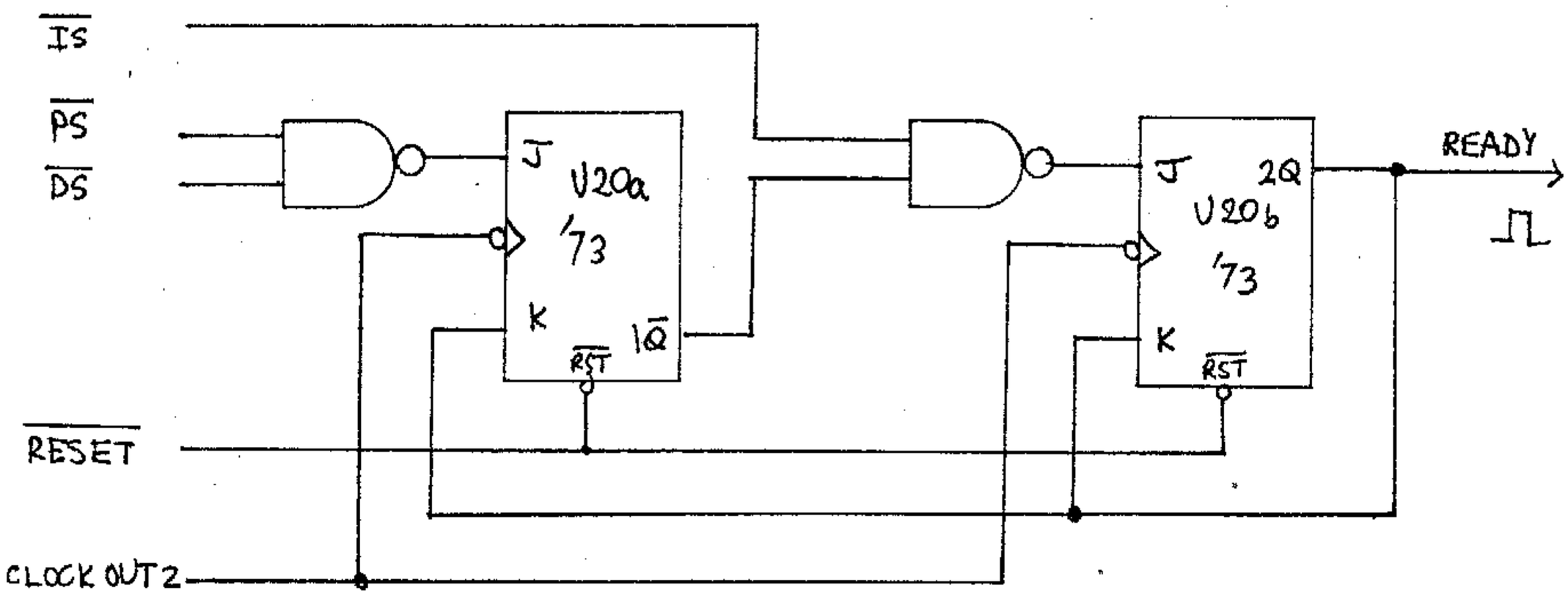


Figure 32. Wait State Realisation

The following figure shows that when the eproms are selected (PS* goes LOW), two CLOCK_OUT_2 cycles have to pass (clocking PS* through) before the READY lines is recognised. Similarly, only one CLOCK_OUT_2 cycle is needed to propagate the IS* line state.

3.1.3 Reset Logic

The aim behind this specially designed reset logic was to retrieve the TMS_320_20 out from a possibility of internal latch-up. The following figure shows how the signals flow. Apart from the standard manual reset, via the pushbutton, this circuit was supposed to reset the program counter to zero in the TMS_320_20 if after a hardware a hardware determined time interval had expired.

If output 2Q was connected to input 1A, then the following events should occur after pressing the reset button:

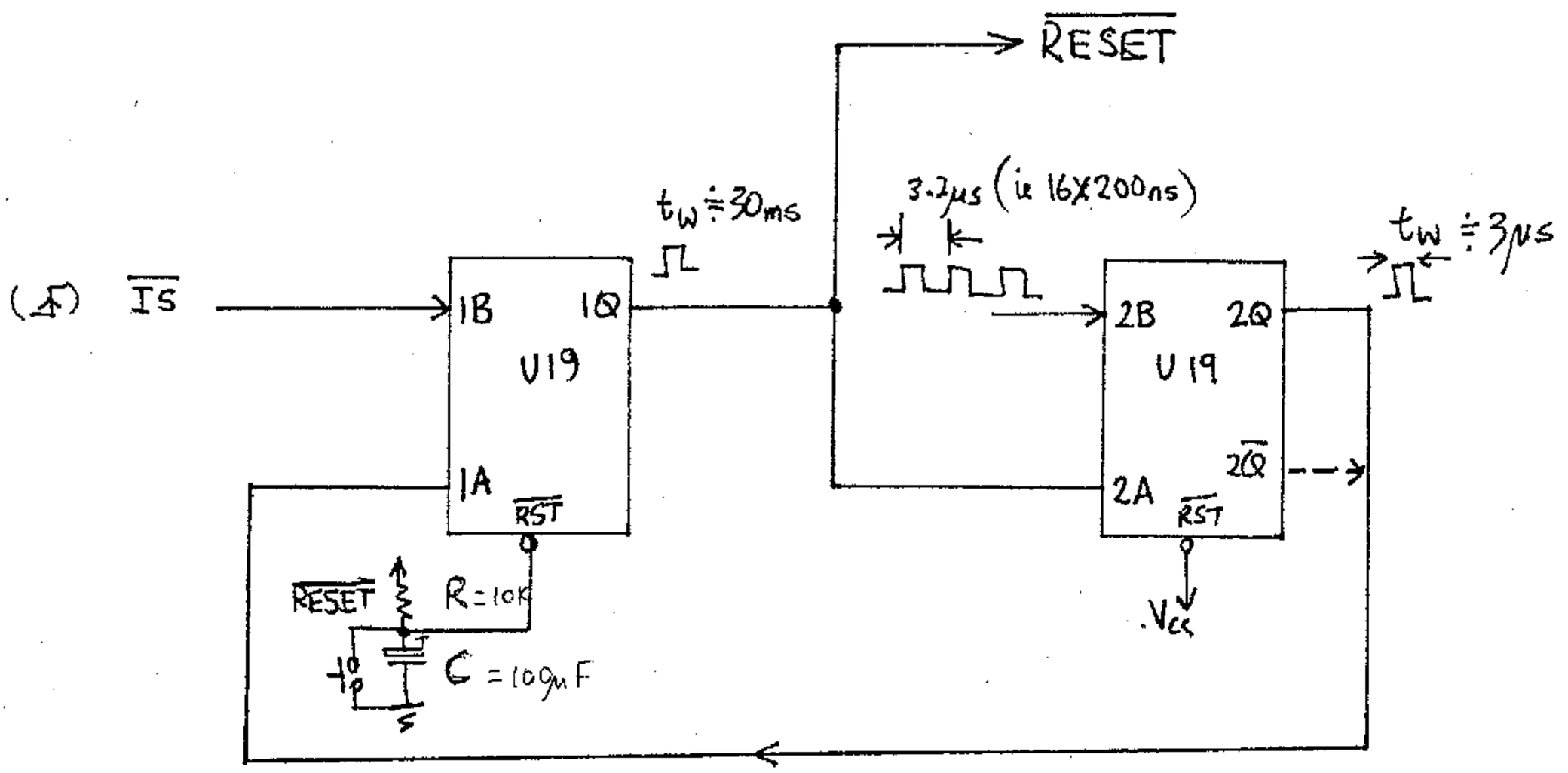


Figure 3.3. Designed Reset Circuit

1. during charging up of capacitor C, output 1Q = LOW (= 2A)

2. when the rising edge is detected (1 second time constant), 1Q goes HIGH for 30 ms and the program is started and a possible 50,000 intstructions can be executed (30 ms/ 600 ns).

3. As 2A is HIGH, then 2Q = LOW, so 1A = LOW, then the circuit is waiting for a rising edge from IS* on 1B. That is, an input or output instruction to the proper address.

   a. If there isn't a rising edge, then 1Q will time-out, setting 2A = LOW (resetting the program) and next rising edge on 2B will cause a time-out of approximately 3 us. On the falling edge of 2Q (= 1A), 1Q will go HIGH for another 30 ms period.

b. If there is an input or output instruction before the 30 ms time-out, then the rising edge on 1B causes 1Q to be set for another 30 ms countdown.

3.1.4 SCF start up

To kick off the counters, the software has to cause the DS* line to go LOW. The way it was done here in the design was to write to an external data memory location. Figure 1.2 in the introductory chapter shows where this area is, and the way the instruction is executed is shown in Appendix 3 in program inout2.lst.

The following operational description refers to Figure 3.4 below or the relevant part is on the cicrcuit diagram (Appendix 1). The basic Operation is as follows:

1. rising edge of (STRB* and DS*) clocks binary number into register,
2. CLOCK_OUT_1 increments number already in the counter to FF (hexadecimal),
3. output RCO* goes LOW for approximately 20 ns,
4. falling edge of RCO*
   a. triggers single shot monostable multivibrator for approximately 0.3 RC seconds, (input = 140 ns; output = 1.4 us), and
   b. causes binary counter to reload itself from register
5. goto step 2)

This counting procedure literally continues forever. However, the starting number can be altered by the appropriate output to external data memory instruction of the TMS_320_20.

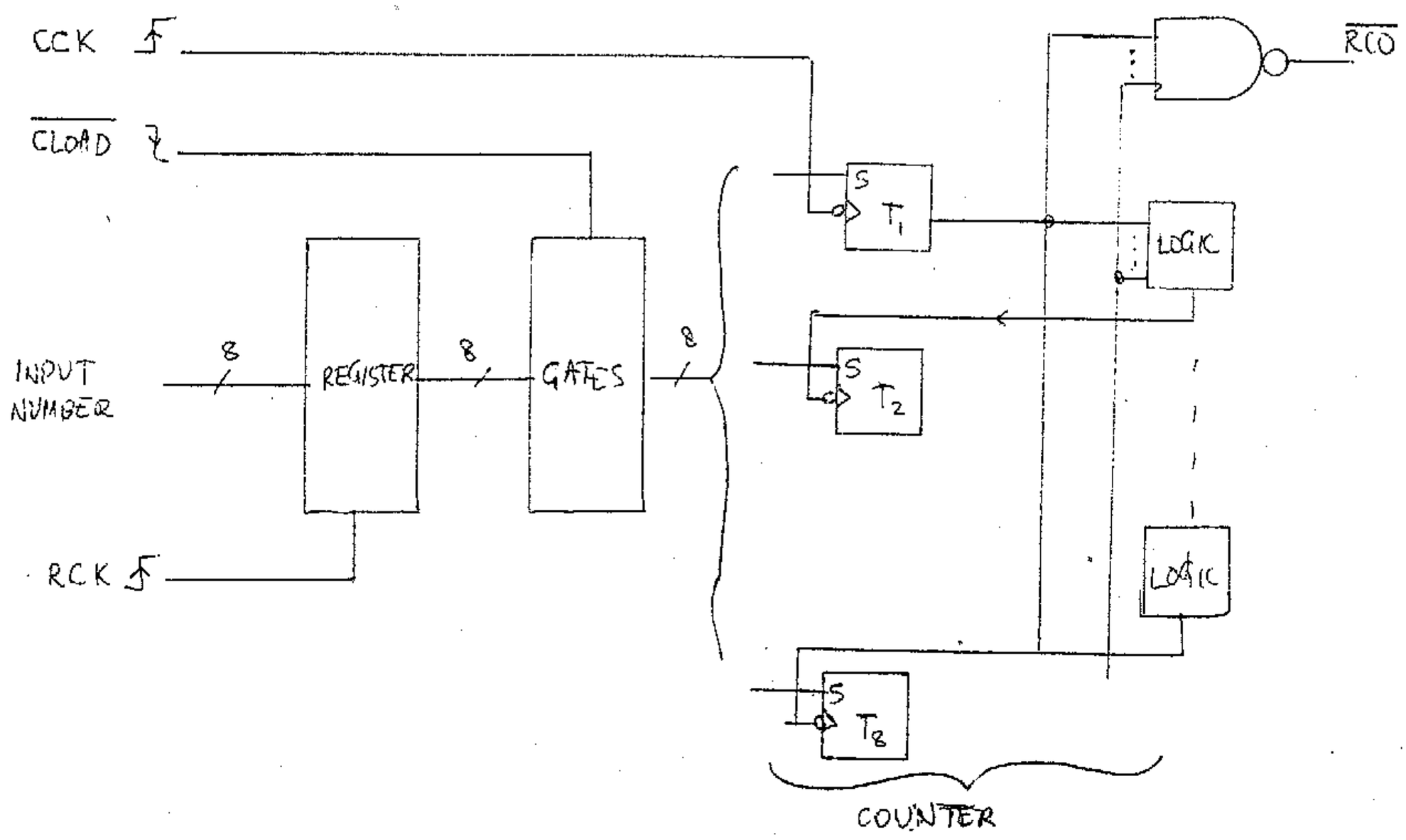


Figure 34. Diagram of Counters

Initially, RCO* is HIGH, so the number present in the counter at power up is incremented to FF (hex). Once this is reached, the RCO* line goes low for approximately 20 ns and the falling edge propagates through the logic gates to CLOAD* to load the counter. Thus, after the first timeout from power up, the correct number is loaded into the counter from the register.

3.1.4.1 Loading the Internal Register

From Figure 3.5 below, using lines B & C, allows a definite HIGH or LOW state to be set - mainly designed for the experimental stage. Ideally, the internal register would be hardwired to the data bus and the number would be put on the bus from the program. This now clarifies the purpose of point 'A' in the figure.

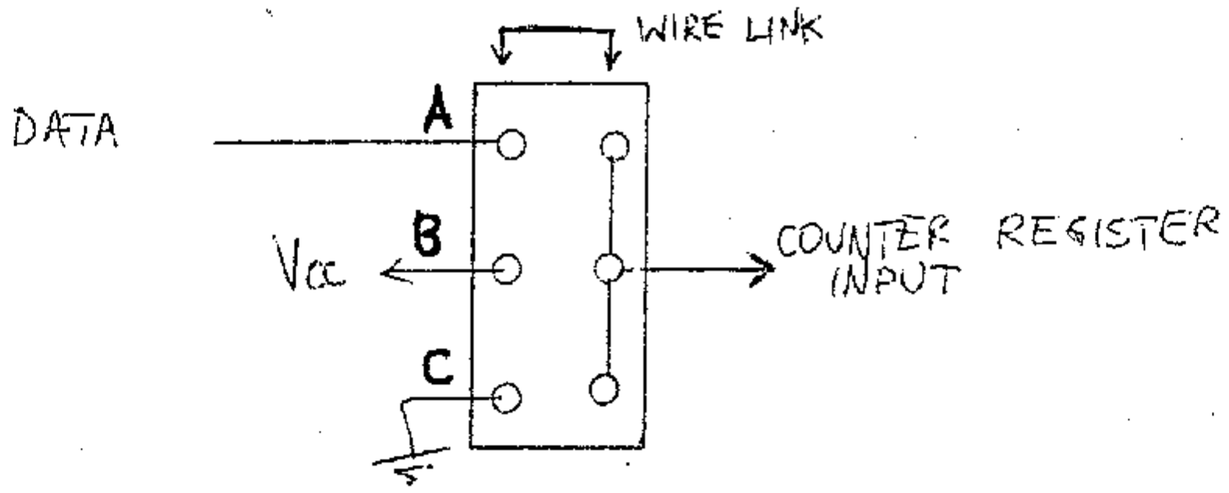


Figure 3.5. Internal Register Input

3.2 Software

In this section, the various aspects linked to the software will be highlighted. It will include the test programs used, the cycle timing, the specific application program and the burning of the EPROMs.

3.2.1 Test Programs

Appendix 3 has the basic test program to get the board running. Also, in the final version of software, the number can be loaded into the counters, supplying the trigger pulse to the SCFs, in the following way:

```
        LDPK    8               * page 8
        SACL    0               * external memory address now
        LDPK    0               * back to page 0
```

This assumes that the 16-bit number has already been calculated and left in the lower 16-bits of the 32-bit accumulator. This code was not used in the initial stages of troubleshooting as it would have been a time consuming task to burn the EPROM with the program and find out that the number was wrong. Hence, the usefulness of the wire link on the dual-in-line (DIL) sockets. See also the

circuit diagram in Appendix 1 or Figure 3.5 earlier.

3.2.2 Cycle Timings

See Appendix 5.

3.2.3 Specific Application

To apply the system, Ron Fox had written a single side band generator program for the TMS_320_10 system that worked according to his design. However, to modify it for the TMS_320_20, has consumed a large amount of his time and unfortunately it will not be possible to use it on this run board.

Before he embarked on the task, he mentioned that is was to load the time critical code into the reconfigured data memory (as program memory) and run it at the standard 200 ns cycle time instead of fetching the code from the EPROM at an average of 600 ns per cycle. Appendix 6 shows where the configured data memory (as program memory) is.

One limitation was the maximum of 256 words of now usable program memory because the time critical code was 344 words long. Perhaps condensing the code from 344 words to something less than 256 words using the additional instructions the TMS_320_20 offers was more involved than at first ? The listing of the TMS_320_10 program is included in Appendix 8.

3.2.4 Eprom Burning

Converting the code from the assembler into a suitable form to be fed into the two EPROMs (upper and lower 8-bits) was a headache for some time. Appendix 7 shows two ways of loading the information into the EPROMs.

3.2.5 Using the SWDS

In order to assemble the TMS_320_20 program, the software development system (swds) can either be used in Room 313 or in Room 423. The system up in 423 allows the user to run the program and debug it. This is a very powerful package from Texas Instruments, but the chances are that if you don't book the system, someone else will always be there when you want to use it. The setup allows you to print your assembled program. See Appendix 7 for the use of the system in either Room.

4. FINDINGS

This chapter will cover the up-to-now unexplained peculiarities of the circuit.

4.1 Nuances

4.1.1 Hardware Reset

For some strange reason, perhaps race conditons in the logic circuits and passive elements, the reset circuit does not work as outlined in section one of the third chapter on control. If it is connected the way stated, referring to Figure 3.3, the RESET line stays LOW permanently. If however, the feedback into input 1A is changed from 2Q to 2Q* then the cicrcuit can run its program in the EPROM (Modification courtesy of Joe Yiu).

According to my analysis, as soon as a rising edge is recognised on the RST* line, 1Q (=2A) should go HIGH for 30 ms; program starts, then 2Q* = HIGH (=1A), forcing 1Q LOW (ignores any change on 1B) after the next available time-out. Conclusion: it will stay LOW forever. It seems the opposite is the case in practice. Perhaps the reasoning is faulty?

4.1.2 SCF kick off

At the start of the program, the DS* must be forced LOW for a cycle or two so a rising edge can clock the 8-bit number for the counter into the register from where the counter is loaded. This can only happen when the program writes out to external data memory.

The number fed into the register must be subtracted from 256 and this will be the correct number the counter decrements to before an output pulse emerges.

4.1.3 Program Memory

With the reconfiguration of data memory, only 256 words are available for 200 ns cycle time execution. This puts a limit on the volume of code to be interpreted. See Figure 1.2 or Appendix 6 for block B0 on memory map diagram.

4.2 Problems Encountered

- Continuing someone else's train of thought,
- Making assumptions:
    1. component values in circuit diagram correspond to the ones on the printed circuit board,
    2. all relevant pins of integrated circuits were shown on the circuit diagram.
- The design not breadboarded and sections tested on the outset,
- Burning EPROMs : 8 most significant bits in one EPROM and the least 8 bits in the other - no 16 bit EPROMs on the market,
- Only having 256 words of reconfigurable TMS_320_20 data memory to program memory running at the 200 ns cycle time (no wait states needed).

4.3 Unresolved Questions

a. Why have the SCFs there in the first place when a simple buffer to adjust the output voltge level would suffice ? This eliminates the need to provide a trigger pulse to determine the 3 dB frequency and removes 7 I.C.s from the circuit.

that is,

I.C.	U
2 * 5613	3, 5
2 * LM_301	1, 6
2 * 74_LS_592	10, 13
1 * 74_LS123	8

b. The program into the EPROMs still has to be converted on the mainframe computer (commsl or karri). What happens if commsl is down ? You're out of action - which is why it's a good idea to have a compiled conversion program on the IBM in Room 313 for the EPROM burner.

c. THe whole aim is to have a portable system independent of any mainframe computer that can burn EPORMs without too much time consumed.

4.4 Modifications

These are possible changes that could be made towards achieving the final working design:

1. power supply flow indication LEDs,

2. removal of the 6-pin DIL sockets on the printed circuit board in order to decrease the amount of board real estate,

3. using faster EPROMs, thus removing the wait state logic and enhancing the program execution speed back to 5Mhz from the current speed of 1.67 MHz (600 ns effective cycle time),

4. possibly using fast Random Access Memory with battery backup instead of using slow EPROMs.

5. Notes made by Reader

These three pages are resevered for any comments the reader may wish make.

TMS_320_20 Run Board

Notes

TMS_320_20 Run Board

Notes

6. APPENDICES

6.1 Hardware Information

This Appendix should contain all the information relating to the hardware side of the project. These include:

6.1.1 Circuit Diagram

See next page.

6.1.2 TMS 320 20 pin outs

See after circuit diagram.

6.1.3 I.C. Layout

This section includes the integrated circuit layout on the printed circuit board with top view pin numbers shown. See after pin outs.

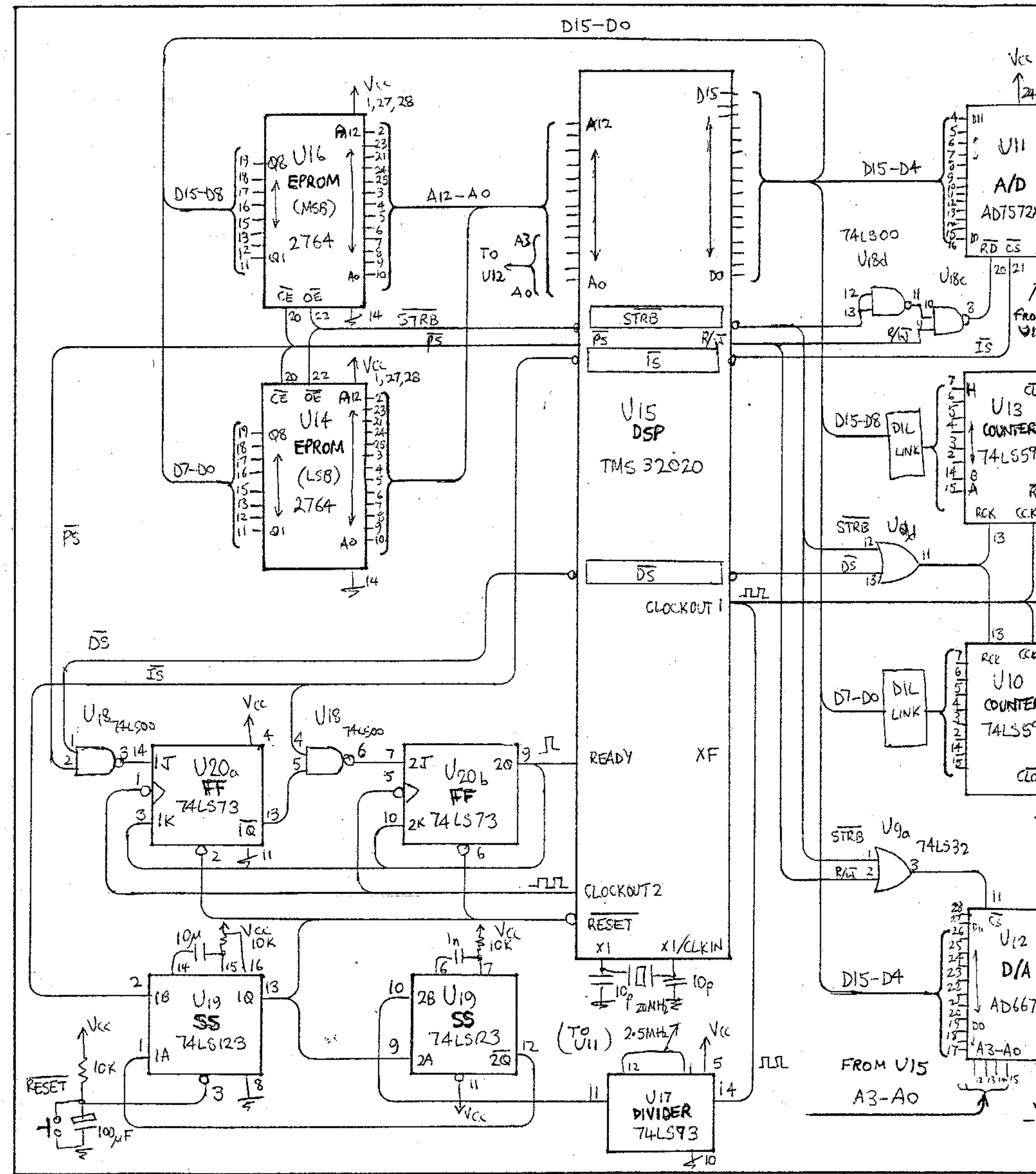
6.1.4 A wiring diagram of the system

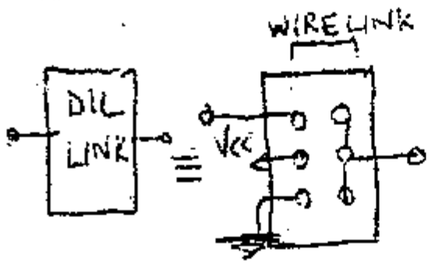
The following diagram shows where to connect the input, output and power supply leads as it wasn't etched onto the printed circuit board.

Figure 61. System Wiring Diagram

6.1.5 SCF data sheet

Note, this is the only data sheet included as the other data sheets (TTL and Analogue Devices) are readily accessible from the other Data Books.





V_{CC} = +5V LOGIC SUPPLY

±5V = ANALOGUE SUPPLY

POWER SUPPLY CONNECTIONS FOR U15 (TMS 32020

V_{CC}	GND
A6	B1
A10	K11
B10	L2
H2	
L6	

PULLED UP		GND'D	
HOLD	A7	CLKR	B9
SYNC	F2	CLKX	A9
INT Ø	G1	DR	J1
INT1	G2	FSR	J2
INT2	H1	FSX	F10

Ø9,18 V_{CC} = PIN 14 & GND = PIN 7

POWER SUPPLY COMPONENTS

V_{IN}	FIGURE	X1	C_{IN}	EARTH	C_{BYPASS} (n x)	U	V_{OUT}
+8V	A	LM309	0.47µF	LOGIC ⏚	11 x 10nF	8-11, 13, 14, 16-20	V_{CC}
+8V	A	78L05	o/c	SIGNAL ⏚	2 x 100nF	2, 5	+5V
-8V	B	79L05	o/c	⏚	2 x 100nF	2, 5	-5V
+15V	A	78L15	0.47µ	⏚	6 x 100nF	1, 3, 4, 6, 7, 12	+15V
-15V	B	79L15	0.47µ	⏚	7 x 100nF	1, 3, 4, 6, 7, 11, 12	-15V

ALL (C_{OUT} = 1µF & C_{IN}) = 35V TANTALUM ; C_{ADD} = 10µF for U11 & U15

U11 A/D AD7572XX05

U7 S/H AD585

U3 LM301

RECONSTRUCTION FILTER

U2 SCF 5613

U1 LM301

ANTI-ALIASING FILTER

FROM U17

2·5MHz

U13 COUNTER 74LS592

U9b 74LS32

U9c

U10 COUNTER 74LS592

U8a SS 74LS123

U8b SS 74LS123

OUTPUT 2

U12 D/A AD667JN

U4 LM301

ANTI-ALIASING FILTER

U5 SCF 5613

U6 LM301

RECONSTRUCTION FILTER

OUTPUT 1 (0–19·5kHz)

	V_{OUT}
16-20	V_{CC}
	+5V
	-5V
7, 12	+15V
11, 12	-15V

U15

FIGURE A:

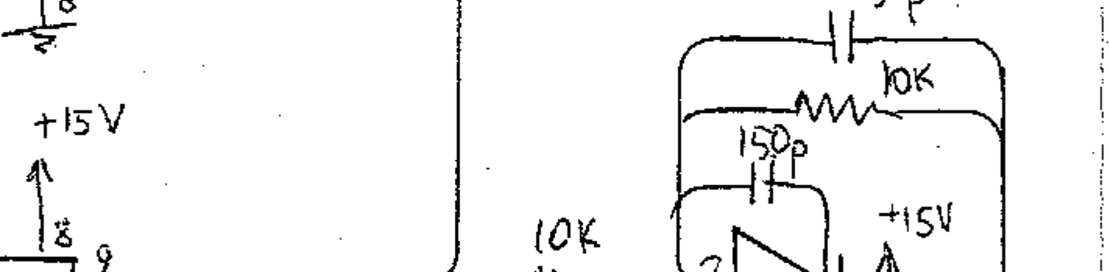
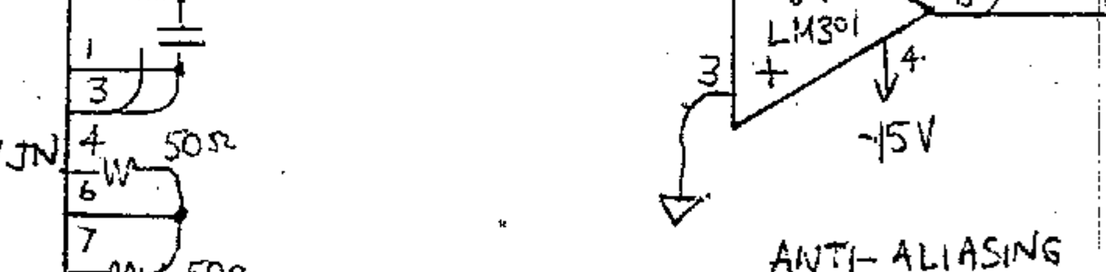


FIGURE B:

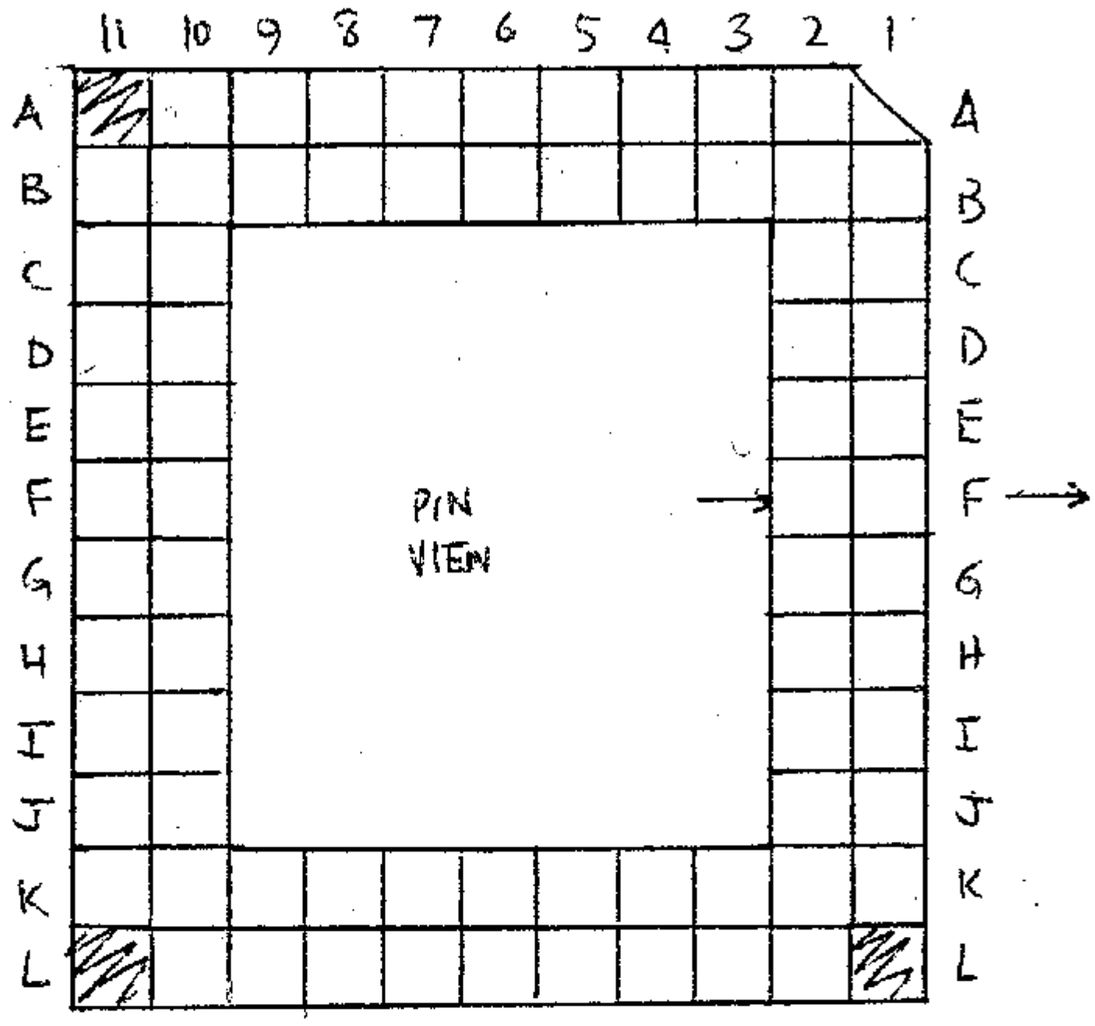


TITLE:	TMS32020 RUN BOARD
	DATE
DESIGNED BY	RON FOX — JUNE '86
DRAWN BY	BOB SMITH — APRIL '87

((PULL OUT))

TMS 32020 PIN OUTS (U15)

BOTTOM VIEW SHOWN**



LINE	PIN	LINE	PIN
LSB D0	F1	A0	K1
D1	E2	A1	K2
D2	E1	A2	L3
D3	D2	A3	K3
D4	D1	A4	L4
D5	C2	A5	K4
D6	C1	A6	L5
D7	B2	A7	K5.
D8	A2	A8	K6
D9	B3	A9	L7
D10	A3	A10	K7
D11	B4	A11	L8
D12	A4	A12	K8
D13	B5	A13	L9
D14	A5	A14	K9
MSB D15	B6	A15	L10

LINE	PIN
$\overline{DS}$	K10
$\overline{PS}$	J10
$\overline{IS}$	J11
READY	B8
R/$\overline{W}$	H11
$\overline{STRB}$	H10

$\overline{RS}$	A8
XF	D11

CLOCKOUT 1	C11
CLOCKOUT 2	D10

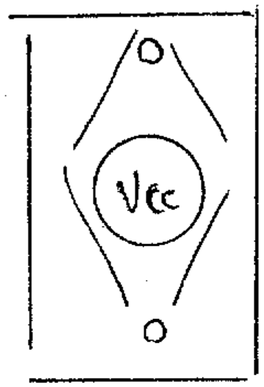
X1	G10
X1/CLKIN	F11

VCC	A6
"	A10
"	B10
"	H2

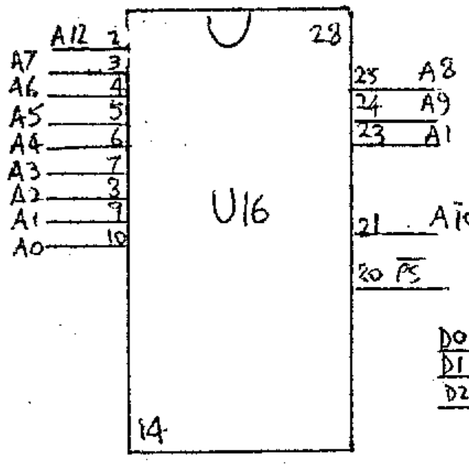
VSS	L6
"	B1
"	K11
"	L2

** TOP VIEW NONSENSICAL

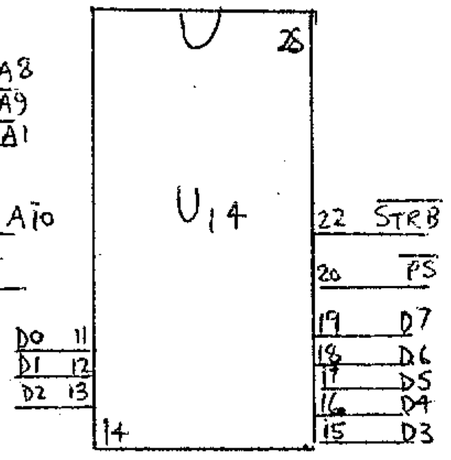
U 1	=	LM 301
U 2	=	R 5613
U 3	=	LM 301
U 4	=	LM 301
U 5	=	R 5613
U 6	=	LM 301
U 7	=	AD 585
U 8	=	74 LS 123
U 9	=	74 LS 32
U 10	=	74 LS 592
U 11	=	AD 7572
U 12	=	AD 667
U 13	=	74 LS 592
U 14	=	TMS 2764 (LSB)
U 15	=	TMS 32020
U 16	=	TMS 2764 (MSB)
U 17	=	74 LS 93
U 18	=	74 LS 00
U 19	=	74 LS 123
U 20	=	74 LS 73

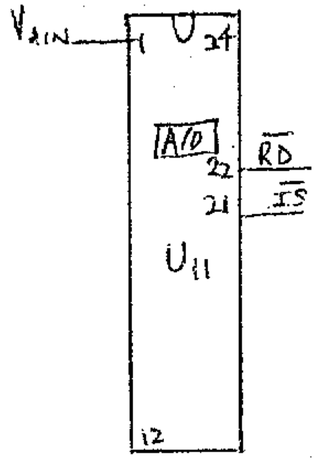


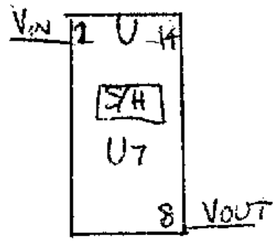
MSB

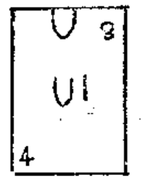


LSB

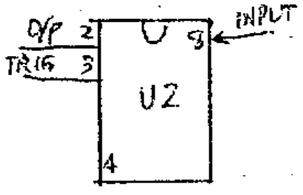




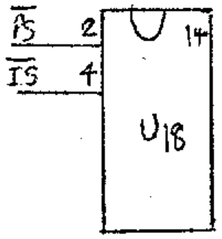


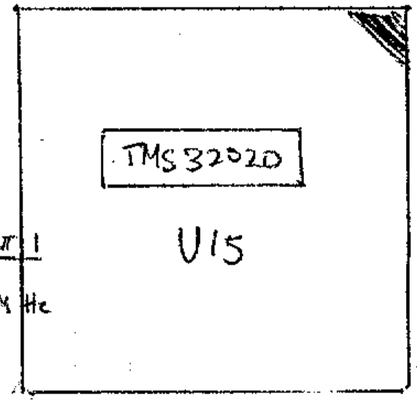


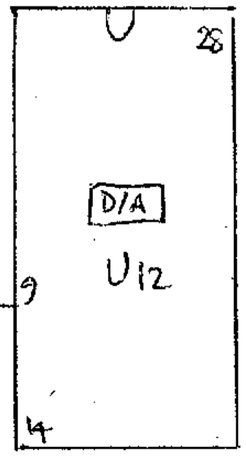
INPUT WIRE

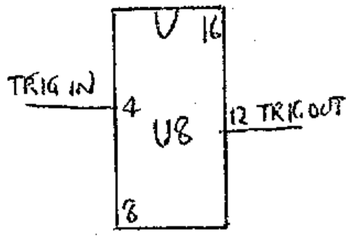


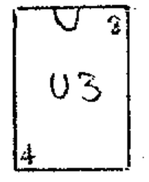
MARKING ON PCB

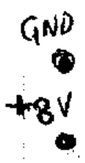


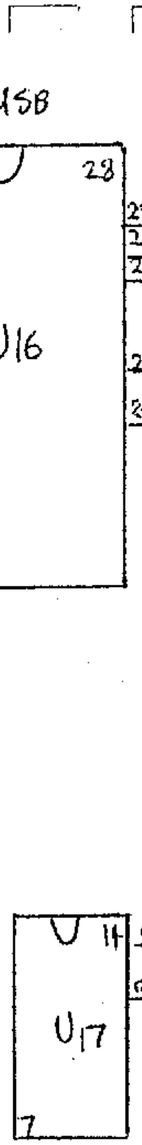


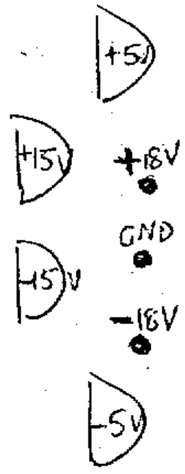


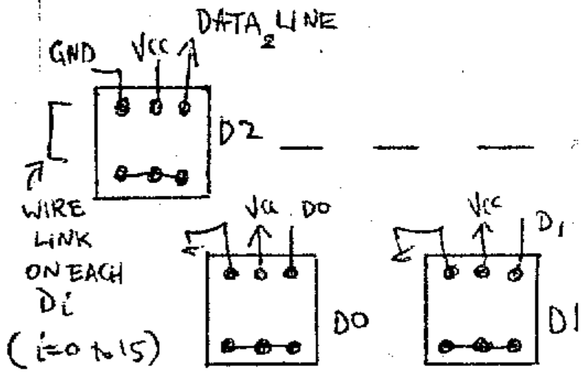


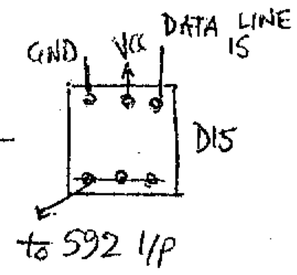


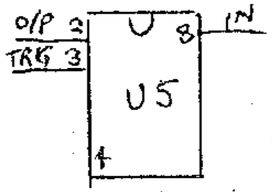


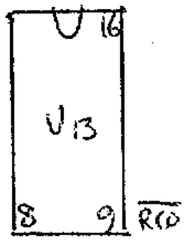


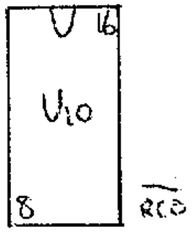


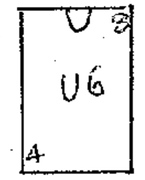












OUTPUT WIRE

IC LOCATION & TEST POINTS

SCF DATA SHEET

EG&G RETICON R5609/R5613 LOW-PASS FILTERS

Key Features

- Easy to use
- No external components required
- Small size: 8 pin mini-DIP
- Wide power supply range: ±5V (or 10V) to ±10 (or 20V)
- Dynamic Range: up to 75 dB
- Corner Frequency Range: 10 Hz to 25 KHz
- Insertion loss: 0 dB

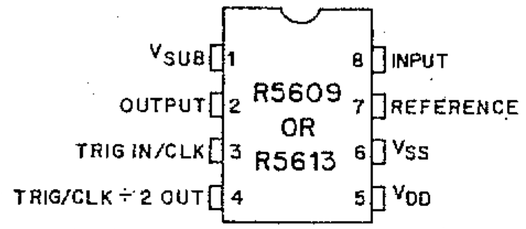


Figure 1. Pinout

Description

The Reticon R5609 and R5613 are monolithic switched-capacitor low-pass filters fabricated in a double-poly NMOS process.

The R5609 is a seven-pole, six-zero elliptic low-pass filter with over 75 dB out-of-band rejection and less than ±0.5 dB of passband ripple. The Reticon R5613 is a linear-phase low-pass filter with over 60 dB out-of-band rejection, it has an elliptic stop band response for faster rolloff.

The stability of the switched-capacitor filters eliminates alignment problems and the need for tight tolerance components and trimpots.

Typical Applications

- Antialias filters
- Reconstruction filters
- Tracking filters
- Audio analysis
- Telecommunications
- Portable instrumentation
- Biomedical/Geophysical instrumentation
- Speech processing

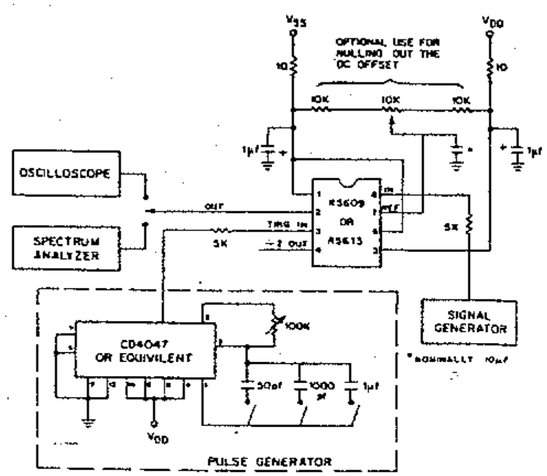


Figure 2. Test circuit

Note: The following should be applied in test and production circuits as applicable. (See Table I)

1) Power supply resistors may be required for transient protection.

2) Input and trigger resistors are required if signals or trigger may be applied with power off, and there is not a resistor already in series with the input or trigger.

Device Operation

The R5609 and R5613 are self-contained and require only an external clock trigger, either TTL or CMOS, and power supply. The device characteristic and operating parameters were obtained using the test configuration shown in Figure 2.

In applications where DC information must be passed through the filter, the output offset may be nulled out by varying the reference voltage, which will change the input trigger level and may require adjustment of clock voltage values. The reference input requires less than 100 µA of current and must always be well-filtered. A circuit that may be used to adjust out the output offset is shown as optional resistors in Figure 2.

A divide-by-two output is also available. This output contains a square wave at the sample rate and may be used for triggering, summing out the sample rate residue, or driving additional filters especially when filtering requirements are spaced by an octave. Gain and phase tracking from device to device and over the temperature range is typically better than .5%. This measurement excludes the fixed offset of fc/fo tolerance at room temperature.

EG&G RETICON • 345 POTRERO AVENUE • SUNNYVALE, CALIFORNIA 94086-9930
(408) 738-4266 • TWX: 910-339-9343

EG&G RETICON R5609/R5613 LOW-PASS FILTERS

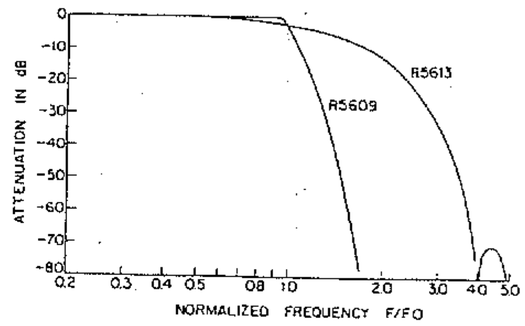


Figure 3. Magnitude response

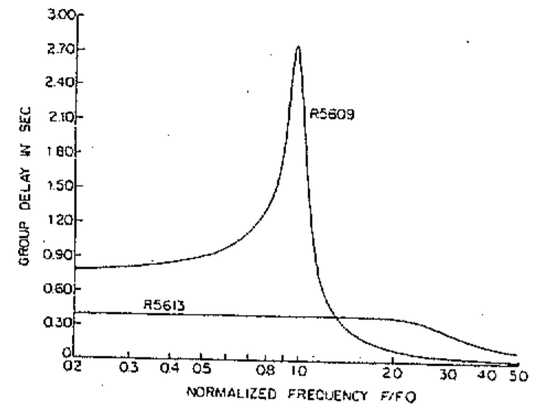


Figure 4. Group delay

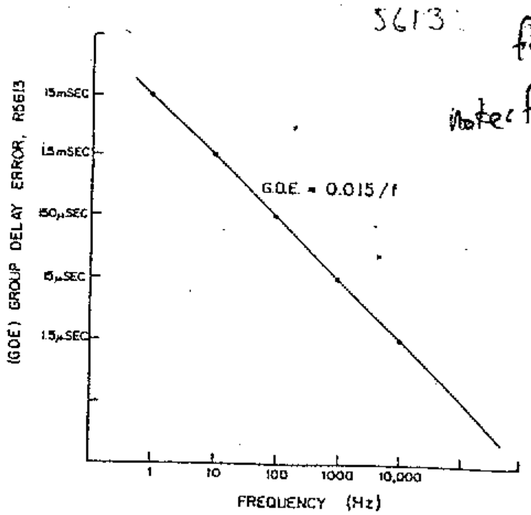


Figure 5. Group delay error
(Second order effects of switched capacitor filter)

Pre/Post Filtering Considerations

The typical sampling rate on the R5609 is 50 times the corner frequency and for the R5613 it is 64. (Note: Sampling rate = ½ input clock trigger rate.) Because these sample rates will be far from the frequencies of interest in most cases, antialiasing filtering will usually not be required. However, as with all sampling systems, frequencies or noise above half the sample rate will be aliased and may appear in the band of interest. If this is the case, an external antialiasing filter will be required on the input. A one or two pole Butterworth low-pass filter will usually suffice. An unstable clock frequency can also produce the effect of an aliased signal. In applications where sampling residue may affect system performance, a single pole RC filter may be added to the output.

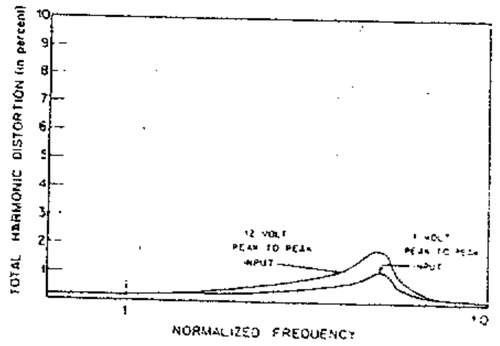


Figure 6. Typical total harmonic distortion

Table I
Absolute Maximum/Minimum Ratings * R5609/5613

	Min	Max	Units
Input Voltage–any terminal with respect to substrate[1]	-.4	21	V
Output short circuit duration–any terminal	—Indefinite—		
Input/Output current–any terminal externally forced[2]		10	mA
Power Dissipation[3]	500		mW
Storage Temperature	-55	125	°C
Operating Temperature–plastic	0	70	°C
–ceramic	-25	85	°C
Junction Temperature (chip)		175	°C
Lead Temperature (Soldering, 10 sec)		300	°C

***Caution:** Observe MOS Handling & Operating Procedures*
**Operation at these limits may result in permanent damage.*

(1) Although devices are internally gate protected to minimize the possibility of static damage, MOS handling precautions should be observed. Do not apply instantaneous supply voltages to the device or insert or remove device from socket while under power. Use decoupling networks to suppress power supply turn-on/off switching transients and ripple. Applying AC signals or clock to device with power off may exceed negative limit.

(2) It is possible to exceed the maximum voltage limit when forcing current, especially on the inputs.

(3) Under worse case loads and temperature, the package limitations prevail. In normal modes of operation, total power dissipated is a combination of Quiescent Power (e.g., $I_{QSS} V_{SS} + I_{QDD} V_{DD}$) plus a percentage of output load power which reflects the efficiency of the output driver. The power dissipation of the substrate will exceed this plastic package limit.

Table II: Device Characteristics & Operation Range Limits[1] R5609/R5613

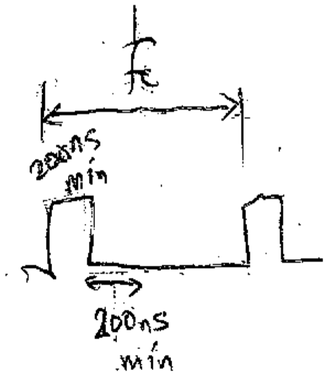
Parameter	Conditions & Comments	Symbol	Min	Typ	Max	Units
Supply Voltages		V_{DD}, V_{SS}	±5		±10	V
Quiescent Current[2]	No load	I_Q	9	12	16	mA
Clock Frequency	$f_C = 2 \times f_S$	f_C	1		2500	KHz
Clock Pulse Width	Ext. drive	t_C	200		$10^6/f_C$-200	nsec
Input Clock Threshold Level		V_{th}	0.8	2.2	3.0	V
Output Signal[3]	V_{in} = 14-20V p-p $R_L \geq 10K\Omega$	V_O	12	13.3		V_{pp}
	$R_L = 0\Omega$	I_O		4		mA
Clock to Corner Frequency Ratio Range	R5609	f_C/f_0	97	100	109	
	R5613		124	128	132	
Corner Frequency Range[3]	Max	f_0	16	20	25	KHz
	Min		10	50	100	Hz
Input Impedance		Ri		≥10		MΩ
		Ci			≤15	pF
Load Impedance(s)		R_L	≥10			KΩ
		C_L			≤50	pF
Dynamic Output Impedance	Small signal	R_O		10	250	Ω

1. $V_{DD}^+ = +10V$, $V_{SS}^- = -10V$, f_C = 500 KHz, T = 25°C
2. Increase 15% for operation to 0°C.
3. Performance degrades at temperatures above 25°C.

Table III: Performance Standards[1] R5609/R5613

Parameter	Comments	Symbol	Min	Typ	Max	Units
Output Noise[1]	BW = fs/2	e_n			2.5	mV_{rms}
Dynamic Range[1]	V_{Op-p}/e_n	D.R.	70	75		dB
Total Harmonic Distortion		THD		(See Fig. 6)	3	%
Insertion Loss			-.4	0	+.4	dB
Clock Feedthrough	$Bw = f_C$			30	60	mV p-p
Ripple[2]			-.2		+.2	dB
DC Offset Voltage			-0.6	0.1	+0.6	Vdc

1. Measured with ±10V supply at 25°C, f_C = 500 KHz, R_L = 500KΩ
2. R5609 only. Not applicable for R5613.



$f_c = f_{3dB\ pt.}$

055-0032
38353

6.2 Wait State Determination

Figure 6.2 shows the clock timing diagram for the TMS_320_20 and the relevant program strobe (PS*) and STRB* lines. When they are connected to the Chip Enable (CE*) and the Output Enable (OE*) pins of the EPROM, the following two 'equations' must be satisfied:

1. CE* (i.e. PS*) must be LOW for atleast 250 ns before the data from the EPROMs will be valid, that is, 295ns from point 0, valid data will be guaranteed by the manufacturer, and

2. OE* (i.e. STRB) must be LOW for atleast 100 ns to have valid data emerging from the EPROM. Again, that is, 165 ns from the point 0 on the timing diagram.

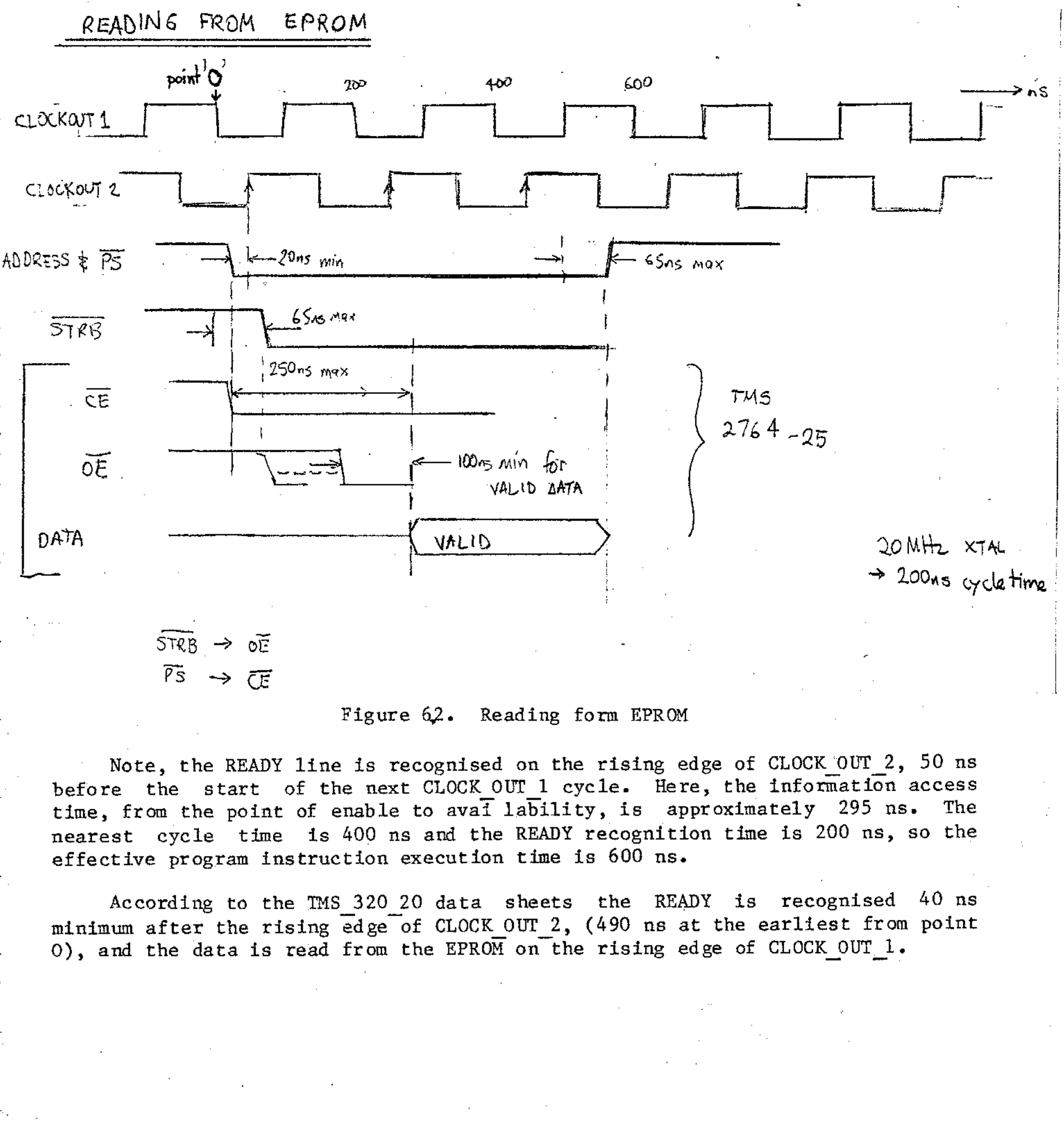


Figure 6.2. Reading form EPROM

Note, the READY line is recognised on the rising edge of CLOCK_OUT_2, 50 ns before the start of the next CLOCK_OUT_1 cycle. Here, the information access time, from the point of enable to avai lability, is approximately 295 ns. The nearest cycle time is 400 ns and the READY recognition time is 200 ns, so the effective program instruction execution time is 600 ns.

According to the TMS_320_20 data sheets the READY is recognised 40 ns minimum after the rising edge of CLOCK_OUT_2, (490 ns at the earliest from point 0), and the data is read from the EPROM on the rising edge of CLOCK_OUT_1.

For the determination input wait states, the data will be available from the ADC 7572 atleast (CHECK THIS !!) 110 ns after the RD* line has been selected LOW on the A/D. Similarly, for the output wait state, the CS* line must be LOW for atleast (CHECK THIS !!) 100 ns to have the correct data to be converted into the analogue form.

The following two figures show the PS* and the READY line waveforms for 2 NOP and an IN + NOP instructions being executed from the EPROMs.

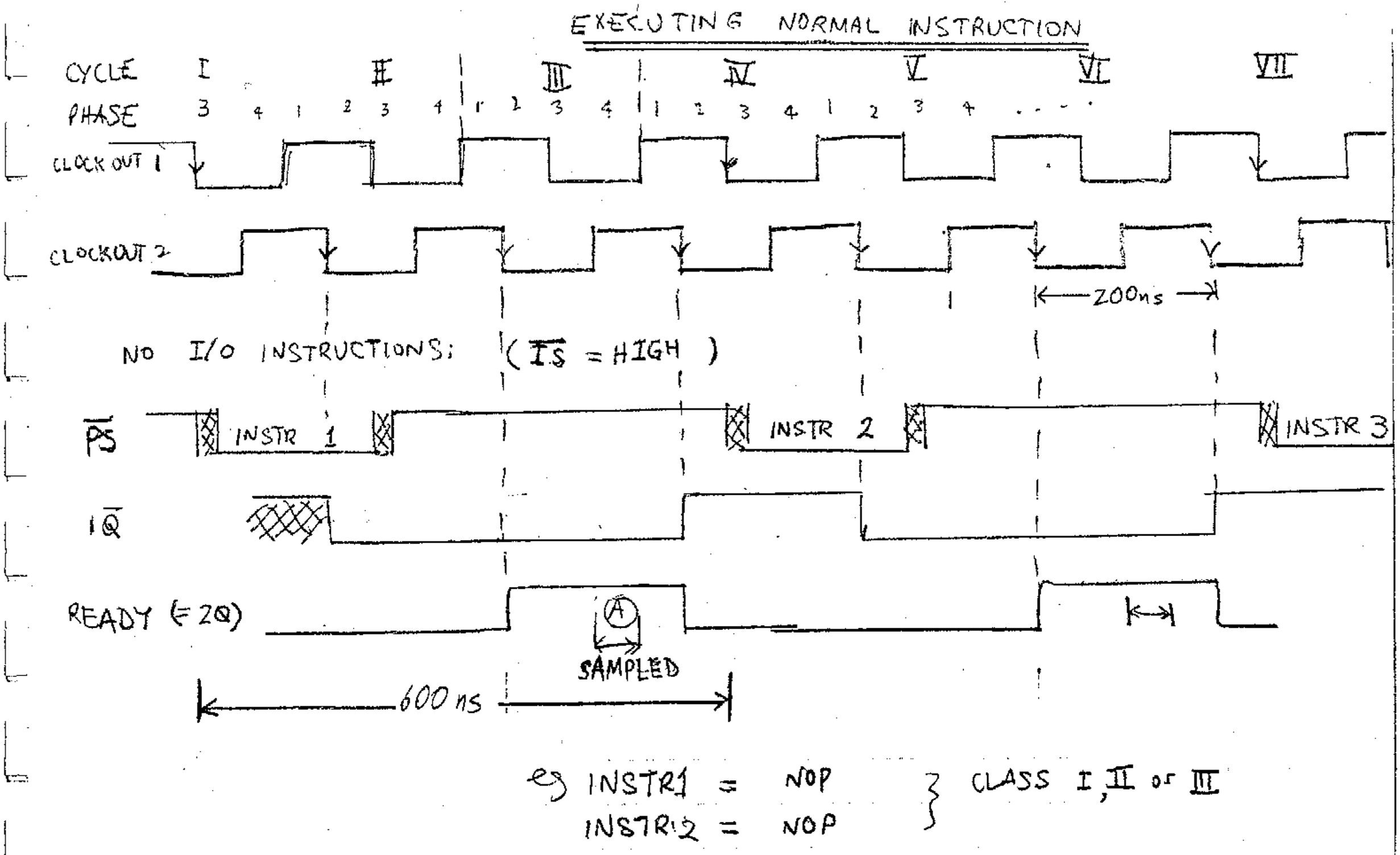


Figure 63. Execution of Normal Instructions

This means that the TNMS_320_20 has to wait for one clock cycle (200 ns), the next shortest time span, before it can proceed with interpreting its instructions. (P.T.O)

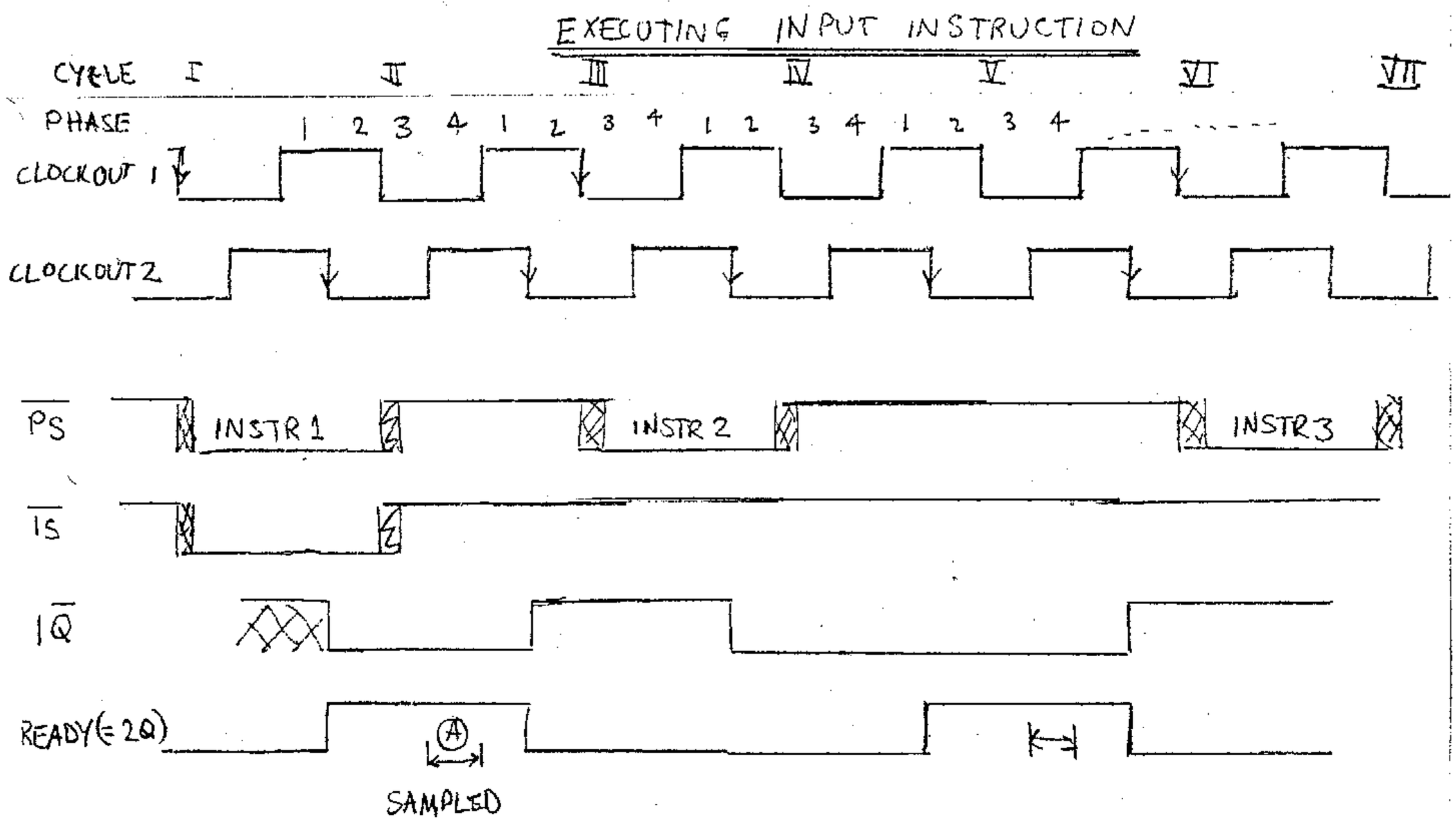


Figure 64. Execution of an Input Instruction

6.3 Testing Programs

In this Appendix are two programs written for the TMS_320_20 to test the run board. The first, "xflag.lst", is an assembled and listed program that simply toggles the logic level of the XF pin on the TMS_320_20.

Its purpose was to monitor the execution of the program on the oscilloscope. It was designed to test the communicating ability between the TMS_320_20, the EPROMs and the READY signal generation logic.

```
NO$IDT      32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036     14:38:57  04-01-87
                                                                    PAGE 0001

0001 0000                   AORG    0
0002 0000 FF80   RST        B       START
     0001 0032
0003 0032                   AORG    50      * FIRST 32 locations reserved
0004 0032 CE01   START      DINT
0005 0033 CE0D   LOOP       SXF                     * SET EXTERNAL FLAG
0006 0034 5500              NOP
0007 0035 5500              NOP
0008 0036 5500              NOP
0009 0037 5500              NOP
0010 0038 5500              NOP
0011 0039 5500              NOP
0012 003A 5500              NOP
0013 003B 5500              NOP
0014 003C 5500              NOP
0015 003D CE0C              RXF                   * RESET IT
0016 003E FF80              B       LOOP
     003F 0033
NO ERRORS, NO WARNINGS
```

Figure 6.5. xflag.lst

The second program, "inout2.lst", was designed to make use of the A/D & S/H pair and the D/A. Its main function was to input a sample and feed it straight back out through the D/A.

```
NO$IDT      32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036     19:40:52  04-15-87
                                                                PAGE 0001

0001                *
0002                *       input output program
0003                *
0004                *                                          7 Apr '87
0005                *
0006      0060      DMA96   EQU 96           * data memory
0007      0000      PMA0    EQU 0
0008      0007      PMA7    EQU 7            * program memory
0009      0008      PMA8    EQU 8
0010      0000      AUXR0   EQU 0          * Auxiliary reg. 0
0011                *
0012 0000                     AORG    0
0013 0000 FF80    RST         B       STRT
     0001 0032
0014 0032                     AORG    50
0015 0032 CE01    STRT        DINT
0016 0033 C808                LDPK    8       * make DS* go low to trigger filter
0017 0034 2000                LAC     0
0018 0035 CE04                CNFD
0019 0036 C800                LDPK    0
0020                *
0021 0037 8060    SAMPL       IN      DMA96,PMA0
0022 0038 5588                LARP    0       * Delay for about 8 us (next available
0023 0039 C005                LARK    AUXR0,5  * times
0024 003A FB90    LOOP        BANZ    LOOP
     003B 003A
0025                *                               Double buffering for the D/A (667)
0026 003C E860              OUT     DMA96,PMA8  * load the number into register
0027 003D E760              OUT     DMA96,PMA7  * convert and put straight out
0028 003E FF80              B       SAMPL
     003F 0037
0029                        END
NO ERRORS, NO WARNINGS
```

Figure 66. inout2.lst

6.4 Current Drain from Power Supplies

The following is a table showing the current consumed by the particular integrated circuit in its running state.

Voltage				I.C.	Current Drawn (mA MAX)
	+5 Logic	+/-5 Analogue	+/-15		
	X			TMS_320_20	360
	X			74_LS_123	2 * 66 = 132
	X			74_LS_32	9.8
	X			74_LS_592	2 * 60 = 120
	X			74_LS_93	15
	X			74_LS_00	4.4
	X			74_LS_73	8
	X			TMS_2764-25	2 * 100 = 200
	X		X	AD_7572_XX05	12
			X	AD_585	10
			X	AD_667_JN	25
			X	LM_301	4 * 1.2 = 4.8
	X	X		R_5613	14 + 2 = 16
Total: 861, 16, 47 mA maximums for their respective voltage supplies.					

Concluding from these figures, it will not be unreasonable to state a power supply current deliverance of:

- 1.5 Amperes for the Logic (+5 V) rail,
- 100 milli-amperes for the +/- 15 V rails, and
- 50 milli-amperes for the Analogue (+/- 5V) rails.

This means that the voltage regulator power dissipation for the:

- 5 V rail will have to be atleast 4.5 Watts (3 V drop * 1.5 A),
- 15 V rails needing atleast a 0.3 Watt regulator, and
- the Analogue supply regulator 150 milli-Watts.

6.5 Instruction Cycle Timings

The following is a table reproduced from the TMS_320_20 User's Guide (Appendix D p2) and is valid for this circuit where the variables i and p are equal to 1 and 2 respectively.

CLASS	WHEN NOT IN REPEAT MODE				WHEN IN REPEAT MODE			
	PI/DI	PI/DE	PE/DI	PE/DE	PI/DI	PI/DE	PE/DI	PE/DE
I	1	2+d	1+p	2+d+p	n	2n+nd	n+p	2n+nd+p
II	1	2+d	1+p	3+d+p	n	2n+nd	n+p	3n+nd+p
III	1		1+p		n		n+p	
IV	2		2+2p		not repeatable			
V	3	N/A	3+2p	N/A	2+n	N/A	2+n+2p	N/A
VI	2 (br int-to-int)		2+p (int-to-ext)		not repeatable			
	2+p (ext-to-int)		2+2p (ext-to-ext)		not repeatable			
VII	2		2+p		2n		2n+p	
VIII	1+i	2+i+d	2+i+p	3+i+d+p	n+ni	2n+ni+nd	2n+ni+p	3n+ni+nd+p
IX	Table in internal program memory:				Table in internal program memory:			
	3	3+d	3+p	3+d+p	2+n	2+n+nd	2+n+p	2+n+nd+p
	Table in external program memory:				Table in external program memory:			
	3+p	4+d+p	3+2p	4+d+2p	2+n+np	2+2n+nd+np	2+n+np+p	2+2n+nd+ np+p
X	Data source internal: †				Data source internal: †			
	3	3+d	3+2p	3+2p+d	2+n	2+n+nd	2+n+2p	2+n+nd+2p
	Data source external: †				Data source external: †			
	3+d	4+2d	3+d+2p	4+2d+2p	2+n+nd	2+2n+2nd	2+n+nd+2p	2+2n+ 2nd+2p
XI	Program source internal: †				Program source internal: †			
	3	3+d	3+2p	3+2p+d	2+n	2+n+nd	2+n+2p	2+n+2p+nd
	Program source external: †				Program source external: †			
	3+p	4+p+d	3+3p	4+3p+d	2+n+np	2+2n+np+nd	2+n+2p+np	2+2p+2n+ np+nd
XII	2	2+d	2+p	2+d+p	2n	2n+nd	2n+p	2n+nd+p
XIII	1 (minimum, waits for INT)		1+p (minimum, waits for INT)		not repeatable			

† Column headings (DI/DE) refer to data destination.

PI - The instruction executes from internal program memory.
PE - The instruction executes from external program memory.
DI - The instruction executes using internal data memory.
DE - The instruction executes using external data memory.
br - Branch from...
int - Internal program memory.
INT - Interrupt.
ext - External program memory.
n - The number of times an instruction is executed when using the RPT or RPTK instruction.

Figure 6.7. Instruction Cycle Timings

For instance, the input instruction:

IN data_memory_address, port_address

will take (2+1+2) * 200 ns to complete. Note, the program is external (PE) and the data memory is internal (DI) leading to the column 'PE/DI'. Similarly, the

branch instruction:

B program_memory_address

will take (2+2*2) cycles (6 * 200 ns) to complete.

If we examine the program " inout.lst", in appendix 3, from the label "SAMPL" to the end of the program, thefollowing figure shows the number of cycles each instruction takes.

code			cycles
SAMPL	IN		5
	LARP		3
	LARK		6
LOOP	BANZ	LOOP	6 * 6 i.e (5,4,3,2,1,0)*6
	OUT		5
	OUT		5
	B	SAMPL	6

Figure 68. inout.lst cycle timings

This shows, the program samples every 66 cycles (* 200 ns gives 13.2 us). On the Oscilloscope, 12 us was measured, indicating the correct order of magnitude, on the line between the A/D busy* and the S/H hold*.

Similarly for the external flag testing program: xflag.lst, the cycles were:

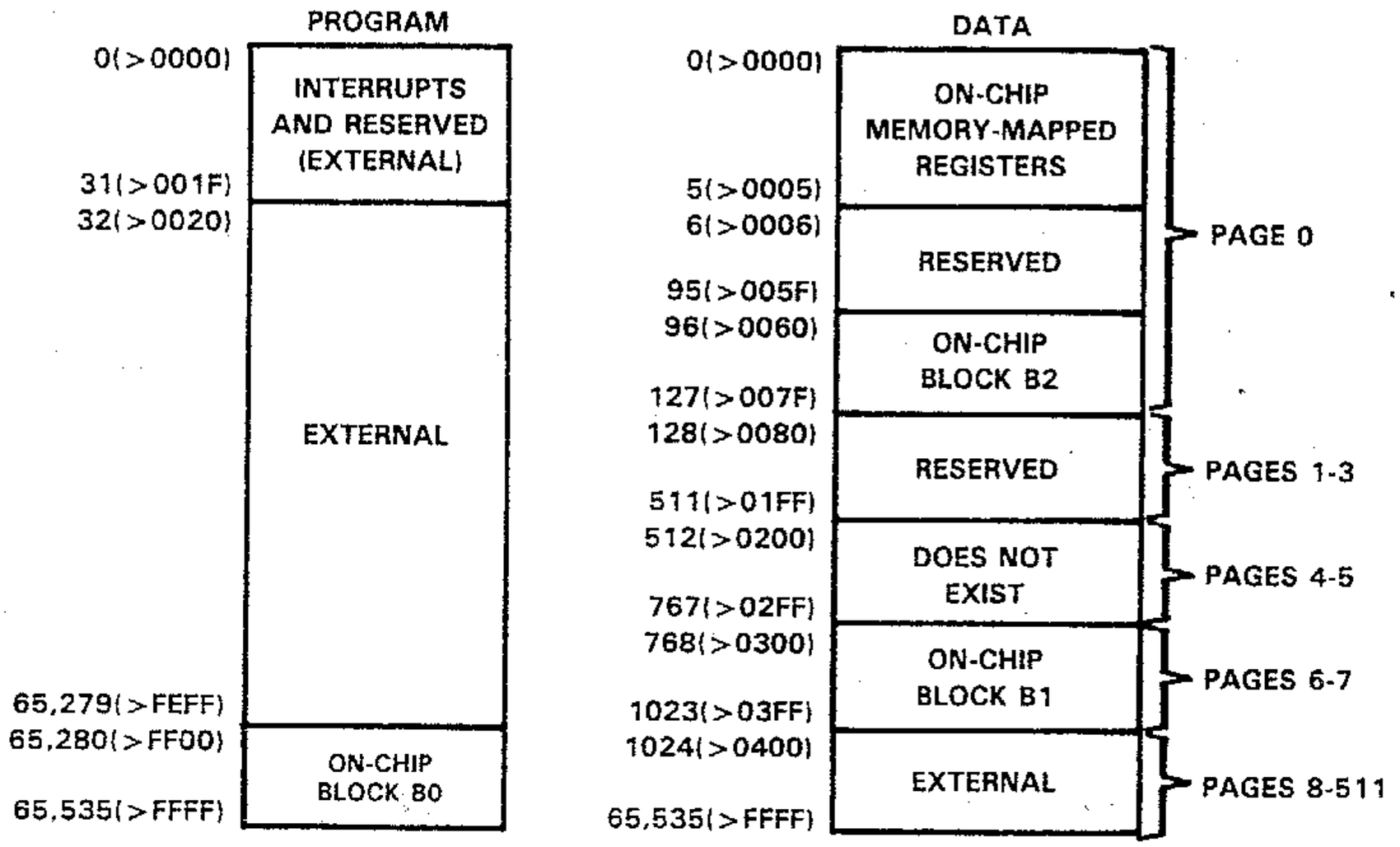
code			cycles	XF state
LOOP	SXF		3	HIGH
	9 * NOPS		9 * 3	HIGH
	RXF		3	LOW for 600 ns
	B	LOOP	6	LOW for 1200 ns

totalling 39 cycles.

Figure 69. xflag.lst cycle timings

6.6 Memory Map Adjustment

For the software written, the memory map of the TMS_320_20 has to be reconfigured so the 'fast' code can executed. The following figure shows how the memory map is altered by the CNFD instruction. Having no permanent internal program memory, only 256 words of the its data memory can be reconfigured.



(b) ADDRESS MAPS AFTER A CNFP INSTRUCTION

Figure 6.10. Memory Map Adjusted

6.7 Burning EPROMs

In this Appendix are two procedures used in burning the TMS_320_20 programs into the 2764, 8k x 8, EPROMs. Note, because of the 16-bit wide address bus, the program op-code was separated into the "upper" and "lower" significant bytes of 8-bits in width.

6.7.1 PROM BURN ala 313

6.7.2 PROM BURN ala DSL

AIM

TO BURN A TMS_320_20 PROGRAM INTO AN EPROM VIA EPROM BURNER IN '313

pre conditions:

Using a PC in ROOM 313

MSKERMIT pgm on floppy in Drive A

cmds → cd swds320 CR
↳ tms CR
] sets up correct paths.

CR = carriage return

COMMENTS	TYPE IN
1. assemble 32020 pgm ie (file.asm)	XASM CR file CR CR file
2. file transfer pgm connect to mainframe login onto system	MSKERMIT CR c CR login: _____
3. invoke KERMIT to receive (-r) option press control] then c to talk to pc	kermit -r CR ^]c
4. Send object file across and connect	send file.mpo CR c CR
5. use object program to split file.mpo into upper & lower 8 bit files (file.mpo.1 & file.mpo.0 respectively	intlx320 file.mpo CR
6. move into recognisable form (MSDOS doesn't recognise 2nd dot in filename	mv file.mpo.0 fi'e.0 CR mv file.mpo.1 file.1 CR
7. send both files from mainframe down to PC (-s option = send) and within 15 seconds after CR press control] then connect across.	kermit -s file.0 file.1 CR ^] c
8. type in receive, wait ..., then type in quit to exit mskermit	receive CR quit CR.

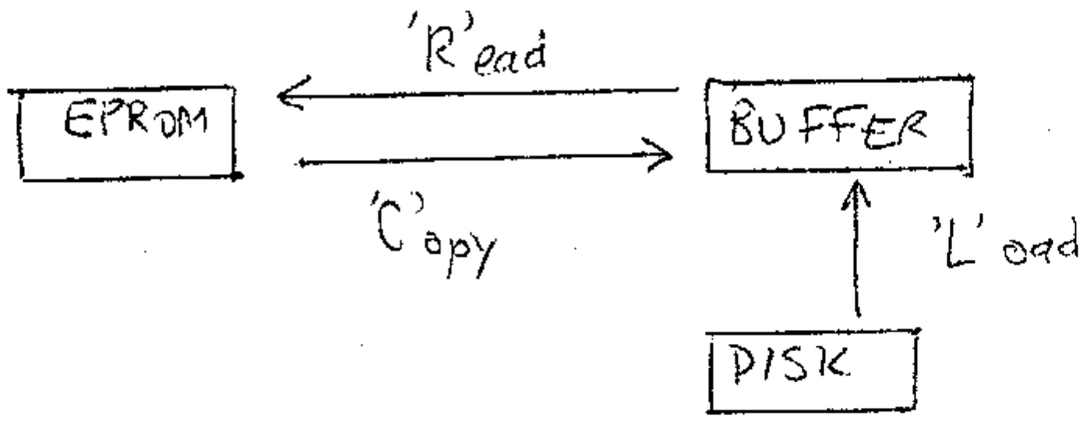
Now have 2 files in INTEL FORMAT to be converted to object code for EPROM burning program (upp512) on the PC.

9. change to EPROM directory on the Hard Disk	cd \eprom CR
10. execute object creating program I/p file = a:file.ø O/p file = say a:file.øout	hexObj CR
11. repeat for 2nd file (file.1)	hex obj CR

/PTO

note !

EPROM SET UP



12. Execute eprom burning program

upp512 . can check if eprom is clean by typing the B option (1FFF = 8 K Bytes)

13. a) load in object file (say LSB)
    b) start address = ØØØØ . . .
    c) can check contents of buffer by display cmd & then quit
    d) copy from buffer into eprom

```
L   a:file.Øout CR
ØØØØ            CR

d   CR
d ØØØØ.ØØFF CR (256 bytes)
Q   CR
C
```

14. as for step 13 for MSB eprom

Now have 2 EPROMs with TMS32020 assembled program in it.

overview

EPROM

MSDOS

MAINFAME

TMS32020 source (file.asm)

ASSEMBLE ↓

file.mpo → intlx 320

creates

file.mpo.Ø (LSB)

file.mpo.1 (MSB)

mv to file.Ø

file.1

hexobj

LSB

MSB

AIM — TO BURN A TMS320-20 PROGRAM INTO AN EPROM VIA EPROM BURNER IN THE DSL ('233)

COMMENTS	TYPE IN
1) 4) Same as steps 1–4 for burning via RM313	
5) execute to create 2 files in Motorola S1S9 format ie creates file.mpo.0 file.mpo.1	tms2020 file.mpo CR
6) send both files to DSL	send jeffs:dsl file.mpo.0 file.mpo.1 CR
7) a) login as tomm – no password required b) collect files at dsl (via jeffs office)	tomm CR qrec file.mpo.0 file.mpo.1 CR
8) edit EACH file & remove the 'L' on the first line of each file.	e file.mpo.0 CR 1d CR • CR w CR q CR
9) use hex to binary conversion program	hexbin file_mpo.0 file.0 CR hexbin file.mpo.1 file.1 CR
Now have 2 binary files for EPROM BURNING program promburn.	
10) execute program to burn code into a 2764 eprom • load file into buffer • start address = 0000 • finish address ≒ last byte of program • examine limits (S & F) • burn into EPROM • verify buffer = EPROM CONTENTS	promburn 2764 CR L file.0 CR S 0000 CR F xxxx CR E CR B V CR
11) repeat step 10) for 2nd file (file.1)	
Now have 2 EPROMS with TMS 32020 assembled program in it.	

USING THE SOFTWARE DEVELOPMENT SYSTEM (SWDS)

RM 313		RM 423
1. cd \swds320	✓	✓
2. tms	✓	✓
3 a) swds	can asm	✓
	can edit	✓
	✗ — can execute file	
	x — debug — load file, step through pgm, quits, etc	
b) xasm	✓	✓
4. turbo-87 to edit file		✓

* NEED MS DOS 3.1 OR HIGHER IN DRIVE A
NEED TMS320_20 PROGRAM IN DRIVE B

✓ = POSSIBLE
x = NOT ✓

TO PRINT FILE, ^P TOGGLES PRINTER ON/OFF.

6.8 SSB program for the TMS 320 10

The following is a listing of the original program to generate the lower side band of the input audio frequency (0-4 kHz). Note the carrier frequency is 16 kHz.

```
1   >
2   ****************************************************************
3   *
4   *       Single sideband generator program to run on TMS_320_10
5   *       Written by Ronald Fox
6   *
7   ****************************************************************
8   *
9   AR0     EQU     0
10  AR1     EQU     1
11  PA0     EQU     0
12  *                               data RAM locations
13          AORG    0
14  P0      BSS     1
15  P1      BSS     1
16  P2      BSS     1
17  P3      BSS     1
18  P4      BSS     1
19  P5      BSS     1
20  P6      BSS     1
21  P7      BSS     1
22  P8      BSS     1
23  P9      BSS     1
24  P10     BSS     1
25  P11     BSS     1
26  P12     BSS     1
27  P13     BSS     1
28  P14     BSS     1
29  P15     BSS     1
30  P16     BSS     1
31  P17     BSS     1
32  P18     BSS     1
33  P19     BSS     1
34  P20     BSS     1
35  P21     BSS     1
36  P22     BSS     1
37  P23     BSS     1
38  P24     BSS     1
39  P25     BSS     1
40  P26     BSS     1
41  P27     BSS     1
42  S0      BSS     1
43  S1      BSS     1
44  S2      BSS     1
45  S3      BSS     1
46  S4      BSS     1
47  S5      BSS     1
48  S6      BSS     1
49  X0      BSS     1
50  X1      BSS     1
51  X2      BSS     1
52  X3      BSS     1
53  X4      BSS     1
54  X5      BSS     1
55  X6      BSS     1
56  U0      BSS     1
57  U1      BSS     1
58  U2      BSS     1
59  W0      BSS     1
60  W1      BSS     1
```



```
61  W2      BSS     1
62  Y1      BSS     1
63  Y2      BSS     1
64  Y3      BSS     1
65  Y4      BSS     1
66  Q02     BSS     1
67  Q12     BSS     1
68  Q22     BSS     1
69  Q03     BSS     1
70  Q13     BSS     1
71  Q23     BSS     1
72  Q04     BSS     1
73  Q14     BSS     1
74  Q24     BSS     1
75  Q05     BSS     1
76  Q15     BSS     1
77  Q25     BSS     1
78  Q06     BSS     1
79  Q16     BSS     1
80  Q26     BSS     1
81  Q07     BSS     1
82  Q17     BSS     1
83  Q27     BSS     1
84  Q08     BSS     1
85  Q18     BSS     1
86  Q28     BSS     1
87          BSS     27              27 unused data locations
88  ONE     BSS     1
89  E2      BSS     1
90  E1      BSS     1
91  E0      BSS     1
92  D2      BSS     1
93  D1      BSS     1
94  D0      BSS     1
95  C6      BSS     1
96  C5      BSS     1
97  C4      BSS     1
98  C3      BSS     1
99  C2      BSS     1
100 C1      BSS     1
101 C0      BSS     1
102         BSS     29
103 A01     BSS     1
104 *
105         AORG    0
106 RESET   B       START           reset vector
107 INT     B       START           interrupt vector
108         AORG    32
109 *                       filter coefficients
110 COEFF   DATA    -15640,15640        A01 = 1 / a1, B11 = - b1
111         DATA    -26714,11328    A12 = a1 / a2, A02 = 1 / a2
112         DATA    26714,-11328        B12 = - b1, B22 = - b2
113         DATA    -24617,10810    A13 = a1 / a2, A03 = 1 / a2
114         DATA    24617,-10810        B13 = - b1, B23 = - b2
115         DATA    -22088,10686    A14 = a1 / a2, A04 = 1 / a2
116         DATA    22088,-10686        B14 = - b1, B24 = - b2
117         DATA    -18916,10720    A15 = a1 / a2, A05 = 1 / a2
118         DATA    18916,-10720        B15 = - b1, B25 = - b2
119         DATA    -15207,10896    A16 = a1 / a2, A06 = 1 / a2
120         DATA    15207,-10896        B16 = - b1, B26 = - b2
```

```
121             DATA    -11204,11324        A17 = a1 / a2, A07 = 1 / a2
122             DATA    11204,-11324             B17 = - b1, B27 = - b2
123             DATA    -7200,12474         A18 = a1 / a2, A08 = 1 / a2
124             DATA    7200,-12474              B18 = - b1, B28 = - b2
125             DATA    369,-1108          C0 = h[0], C1 = h[1] + h[2]
126             DATA    1942,29216       C2 = h[3] + h[4], C3 = 2h[5]
127             DATA    816,-4019    C4 = h[0] + h[1], C5 = h[2] + h[3]
128             DATA    19014,2201         C6 = h[4] + h[5], D0 = h[0]
129             DATA    21144,10716  D1 = h[1] + h[2],D2 = h[0] + h[1]
130             DATA    -2201,-21144       E0 = -h[0], E1 = -h[1] -h[2]
131             DATA    -10716                          E2 = -h[0] -h[1]
132     *
133     ****************************************************************
134     *
135     *       Initialisation section
136     *
137     ****************************************************************
138     *
139     START   LDPK    0
140     *                       Initialise ONE to 1
141             LACK    1
142             SACL    ONE,0
143     *                       Read in 43 filter coefficients
144             LT      ONE
145             MPYK    COEFF
146             PAC
147             LARK    AR0,42
148             LARK    AR1,A01
149     RCONST  LARP    AR1
150             TBLR    *-,AR0
151             ADD     ONE,0
152             BANZ    RCONST
153     *
154     ****************************************************************
155     *
156     *       Main loop using straight line code
157     *       Set clock rate to 4.992 MHz for a 16 kHz sample rate
158     *       Loop is 312 clock cycles long
159     *
160     ****************************************************************
161     *
162     LOOP    LARK    AR0,A01
163     *
164             NOP
165             NOP
166             NOP
167             NOP
168             NOP
169             IN      P0,PA0
170             NOP
171             NOP
172             NOP
173             NOP
174             NOP
175             NOP
176             NOP
177             NOP
178             NOP
179             NOP
180             NOP
```



```
181             NOP
182             NOP
183             NOP
184             NOP
185             NOP
186             NOP
187             NOP
188             NOP
189     *                       Stage 2 of lower interpolator
190     *                       to supply fourth samples
191             ZAC
192             MPY     D0
193             LTD     W1
194             MPY     D1
195             LTD     W0
196             MPY     D2
197             LTA     P0
198             ADD     Y4,15           change ADD to SUB for other sideband
199             ADD     ONE,14          for round-off
200             SACH    Y4,1
201     *                       Stage 1 of Hilbert transformer
202             LAC     P1,14
203             MPY     *-              A01
204             LTA     Q12
205             MPY     *-              B11
206             LTA     Q12
207             ADD     ONE,14          for round-off
208             SACH    Q02,1
209             ADD     Q02,15
210             OUT     Y3,PA0
211             SACH    Q02,1
212     *                       Stage 2 of Hilbert transformer
213             LAC     Q22,14
214             DMOV    Q12
215             MPY     *-              A12
216             LTD     Q02
217             MPY     *-              A02
218             LTA     Q13
219             MPY     *-              B12
220             LTA     Q23
221             MPY     *-              B22
222             LTA     Q13
223             ADD     ONE,14          for round-off
224             SACH    Q03,1
225             ADD     Q03,15
226             SACH    Q03,1
227     *                       Stage 3 of Hilbert transformer
228             LAC     Q23,14
229             DMOV    Q13
230             MPY     *-              A13
231             LTD     Q03
232             MPY     *-              A03
233             LTA     Q14
234             MPY     *-              B13
235             LTA     Q24
236             MPY     *-              B23
237             LTA     Q14
238             ADD     ONE,14          for round-off
239             SACH    Q04,1
240             ADD     Q04,15
```

```
241             SACH    Q04,1
242     *                               Stage 4 of Hilbert transformer
243             LAC     Q24,14
244             DMOV    Q14
245             MPY     *-              A14
246             LTD     Q04
247             MPY     *-              A04
248             LTA     Q15
249             MPY     *-              B14
250             LTA     Q25
251             MPY     *-              B24
252             LTA     Q15
253             ADD     ONE,14          for round-off
254             SACH    Q05,1
255             ADD     Q05,15
256             SACH    Q05,1
257     *                               Stage 5 of Hilbert transformer
258             LAC     Q25,14
259             DMOV    Q15
260             MPY     *-              A15
261             LTD     Q05
262             MPY     *-              A05
263             LTA     Q16
264             MPY     *-              B15
265             LTA     Q26
266             MPY     *-              B25
267             LTA     Q16
268             ADD     ONE,14          for round-off
269             SACH    Q06,1
270             ADD     Q06,15
271             SACH    Q06,1
272     *                               Stage 6 of Hilbert transformer
273             LAC     Q26,14
274             DMOV    Q16
275             MPY     *-              A16
276             LTD     Q06
277             MPY     *-              A06
278             LTA     Q17
279             MPY     *-              B16
280             LTA     Q27
281             MPY     *-              B26
282             LTA     Q17
283             ADD     ONE,14          for round-off
284             SACH    Q07,1
285             ADD     Q07,15
286             SACH    Q07,1
287     *                               Stage 7 of Hilbert transformer
288             LAC     Q27,14
289             DMOV    Q17
290             MPY     *-              A17
291             LTD     Q07
292             MPY     *-              A07
293             OUT     Y4,PA0
294             LTA     Q18
295             MPY     *-              B17
296             LTA     Q28
297             MPY     *-              B27
298             LTA     Q18
299             ADD     ONE,14          for round-off
300             SACH    Q08,1
```



```
301             ADD     Q08,15
302             SACH    Q08,1
303     *                               Stage 8 of Hilbert transformer
304             LAC     Q28,14
305             DMOV    Q18
306             MPY     *-              A18
307             LTD     Q08
308             MPY     *-              A08
309             LTA     X1
310             MPY     *-              B18
311             LTA     X2
312             MPY     *-              B28
313             LTA     X6
314             ADD     ONE,14          for round-off
315             SACH    X0,1
316             ADD     X0,15
317             SACH    X0,1
318     *                               Stage 1 of upper interpolator
319     *                               to supply even samples
320             ZAC
321             MPY     C0
322             LTA     X5
323             MPY     C1
324             LTA     X4
325             MPY     C2
326             LTA     X3
327             MPY     C3
328             LTA     X2
329             MPY     C2
330             LTA     X1
331             MPY     C1
332             LTA     X0
333             MPY     C0
334             LTA     U2
335             ADD     ONE,14          for round-off
336             SACH    U0,1
337     *                               Stage 2 of upper interpolator
338     *                               to supply first samples
339             ZAC
340             MPY     D2
341             LTA     U1
342             MPY     D1
343             LTA     U0
344             MPY     D0
345             LTA     S6
346             ADD     ONE,14          for round-off
347             SACH    Y1,1
348     *                               Stage 1 of lower interpolator
349     *                               to supply even samples
350             ZAC
351             MPY     C0
352             LTA     S5
353             MPY     C1
354             LTA     S4
355             MPY     C2
356             LTA     S3
357             MPY     C3
358             LTA     S2
359             MPY     C2
360             LTA     S1
```

```
                MPY     C1
                LTA     S0
                MPY     C0
                LTA     W2
                ADD     ONE,14          for round-off
                SACH    W0,1
*                               Stage 2 of lower interpolator
*                               to supply first samples
                ZAC
                MPY     D2
                LTA     W1
                MPY     D1
                LTA     W0
                MPY     D0
                LTA     U2
                ADD     Y1,15           change ADD to SUB for other sideband
                ADD     ONE,14          for round-off
                SACH    Y1,1
                OUT     Y1,PA0
*                               Stage 2 of upper interpolator
*                               to supply second samples
                ZAC
                MPY     D0
                LTD     U1
                MPY     D1
                LTD     U0
                MPY     D2
                LTD     X5
                ADD     ONE,14          for round-off
                SACH    Y2,1
*                               Stage 1 of upper interpolator
*                               to supply odd samples
                ZAC
                MPY     C4
                LTD     X4
                MPY     C5
                LTD     X3
                MPY     C6
                LTD     X2
                MPY     C6
                LTD     X1
                MPY     C5
                LTD     X0
                MPY     C4
                LTA     U2
                ADD     ONE,14          for round-off
                SACH    U0,1
*                               Stage 2 of upper interpolator
*                               to supply third samples
                ZAC
                MPY     E2
                LTA     U1
                MPY     E1
                LTA     U0
                MPY     E0
                LTA     U2
                ADD     ONE,14          for round-off
                SACH    Y3,1
*                               Stage 2 of upper interpolator
*                               to supply fourth samples
```



```
                ZAC
                MPY     E0
                LTD     U1
                MPY     E1
                LTD     U0
                MPY     E2
                LTA     W2
                ADD     ONE,14          for round-off
                SACH    Y4,1
*                               Stage 2 of lower interpolator
*                               to supply second samples
                ZAC
                MPY     E0
                LTD     W1
                MPY     E1
                LTD     W0
                MPY     E2
                LTD     S5
                ADD     Y2,15           change ADD to SUB for other sideband
                ADD     ONE,14          for round-off
                SACH    Y2,1
*                               Stage 1 of lower interpolator
*                               to supply odd samples
                ZAC
                MPY     C4
                LTD     S4
                MPY     C5
                LTD     S3
                MPY     C6
                LTD     S2
                MPY     C6
                LTD     S1
                MPY     C5
                LTD     S0
                MPY     C4
                LTA     W2
                ADD     ONE,14          for round-off
                SACH    W0,1
*                               Stage 2 of lower interpolator
*                               to supply third samples
                ZAC
                MPY     E2
                LTA     W1
                MPY     E1
                LTA     W0
                MPY     E0
                LTA     W2
                ADD     Y3,15           change ADD to SUB for other sideband
                ADD     ONE,14          for round-off
                OUT     Y2,PA0
                SACH    Y3,1
*                               Delay for lower path
                DMOV    P27             Be sure to put P27 below S0
                DMOV    P26
                DMOV    P25
                DMOV    P24
                DMOV    P23
                DMOV    P22
                DMOV    P21
                DMOV    P20
```



```
                DMOV    P19
                DMOV    P18
                DMOV    P17
                DMOV    P16
                DMOV    P15
                DMOV    P14
                DMOV    P13
                DMOV    P12
                DMOV    P11
                DMOV    P10
                DMOV    P9
                DMOV    P8
                DMOV    P7
                DMOV    P6
                DMOV    P5
                DMOV    P4
                DMOV    P3
                DMOV    P2
                DMOV    P1
                DMOV    P0
                B       LOOP
                END
<
```