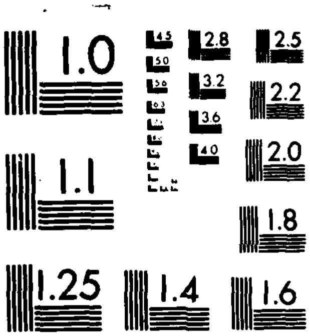
AD-A128 676 INVESTIGATION OF CHARACTERISTICS AND APPLICATIONS OF NEW CCD SIGNAL PROCE..(U) NORTHEASTERN UNIV BOSTON MASS DEPT OF ELECTRICAL ENGINEERING.. B L COCHRUN ET AL. 1/2

UNCLASSIFIED FEB 83 RADC-TR-82-322 F19628-79-C-0165 F/G 9/3 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

AD A128676

RADC-TR-82-322
Final Technical Report
February 1983

INVESTIGATION OF CHARACTERISTICS AND APPLICATIONS OF NEW CCD SIGNAL PROCESSING DEVICES

Northeastern University

Basil L. Cochrun
John G. Proakis

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

DTIC
S ELECTE D
MAY 27 1983
E

83 05 27 003

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-82-322 has been reviewed and is approved for publication.

APPROVED: Jerry Silverman

JERRY SILVERMAN
Project Engineer

APPROVED: Harold Roth

HAROLD ROTH, Director
Solid State Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ESES), Hanscom AFB MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
RADC-TR-82-322

2. GOVT ACCESSION NO.
AD-A128676

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
INVESTIGATION OF CHARACTERISTICS AND APPLICATIONS OF NEW CCD SIGNAL PROCESSING DEVICES

5. TYPE OF REPORT & PERIOD COVERED
Final Technical Report

6. PERFORMING ORG. REPORT NUMBER
N/A

7. AUTHOR(s)
Basil L. Cochrun
John G. Proakis

8. CONTRACT OR GRANT NUMBER(s)
F19628-79-C-0165

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Northeastern University
Department of Electrical Engineering
Boston MA 01731

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
62702F
46001828

11. CONTROLLING OFFICE NAME AND ADDRESS
Rome Air Development Center (ESE)
Hanscom AFB MA 01731

12. REPORT DATE
February 1983

13. NUMBER OF PAGES
142

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Same

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
Same

18. SUPPLEMENTARY NOTES
RADC Project Engineer: Jerry Silverman (ESES)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Signal Processing Modules
Charge-coupled Devices (CCDs)
Sampled Analog Line Data
CCD Building Blocks
Interface Circuitry
Microprocessor Control Software
Recirculating Analog Memory
Analog/Binary Correlator

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This final report on Contract F19628-79-C-0165 describes the work on the development of two high-speed sampled analog signal processing modules based on charge coupled device (CCD) technology. A Z-80 microprocessor serves as the controller for the modules.

The CCDs used in the modules are an analog-binary correlator (ABC), a programmable transversal filter (PTF), a refreshable sampled analog memory (RSAM), a corner-turning memory (CTM) and a Reticon chirp-z transform device. A description of the device is contained in the report and a number of possible inter-connections are considered. Also described are the software and hardware features in the interface between the CCD modules and the microprocessor and the software that was developed for the Z-80 microprocessor.

Preface

This final technical report describes work performed under Contract F19628-79-C-0165 from RADC to Northeastern University, Boston, Massachusetts. The Principal Investigators were Professors Basil L. Cochrun and John G. Proakis. The Technical Contract Monitor for RADC was Dr. Jerry Silverman.

The authors are grateful to S. C. Munroe of the M.I.T. Lincoln Laboratory for discussions regarding the operation of the APC chip and for his suggestions on modification of peripheral circuitry. The authors are also grateful to S. Waldstein of the MITRE Corporation for his assistance in obtaining PTF chips, and for discussions about the characteristics and peripheral circuitry of the PTF.

In addition to the contributions of the Principal Investigators, this report contains contributions by personnel who have worked on this contract during the past three years. Specifically, we wish to acknowledge the work of Mr. George Blustein (software consultant), Ulf Dunberger (a graduate student), and Bill Donaldson, Edward Karaian and Zafer Gulum (Co-op students).

DTIC
COPY
INSPECTED
2

Accession For
NTIS GRA&I ☒
DTIC TAB □
Unannounced □
Justification____________

By____________
Distribution/
Availability Codes

Dist	Avail and/or Special
A	

Table of Contents

Abstract

This final report on Contract F19628-79-C-0165 describes the work on the development of two high-speed sampled analog signal processing modules based on charge coupled device (CCD) technology. A Z-80 microprocessor serves as the controller for the modules.

The CCD's used in the modules are an analog-binary correlator (ABC), a programmable transversal filter (PTF), a refreshable sampled analog memory (RSAM), a corner-turning memory (CTM) and a Reticon chirp-z transform device. A description of the devices is contained in the report and a number of possible interconnections are considered. Also described are the software and hardware features in the interface between the CCD modules and the microprocessor and the software that was developed for the Z-80 microprocessor.

List of Illustrations

1. Introduction

This program involved the development of two high-speed sampled analog signal processing modules based on charge-coupled device (CCD) technology. A microprocessor serves as the controller for the CCD modules. Under program control, each module performs a number of sampled analog signal processing functions, such as correlation, filtering, synchronization, integration and Fourier transformations. The basic building blocks for the modules are the following CCD's.

(a) A 1,024-tap programmable transversal filter (PTF) nine possible configurations

(b) A 512-tap analog-binary correlator (ABC)

(c) A refreshable sampled analog recirculating memory (RSAM)

(d) A two-dimensional memory, called a corner turning memory (CTM)

(e) A Reticon R5601 chirp-z transform device.

One of the two signal processing modules is designed to handle sampled analog line data. That is, it is capable of computing quantities of the form

$$y_k = \sum_{n=1}^{N} a_{k-n} b_n \quad ,$$

where the sequence $\{a_n\}$ represents a set of N samples of the input signal and $\{b_n\}$ represents a reference vector consisting of N points. This type of computation is encountered in performing correlation, filtering and synchronization. In addition, the module has the capability of performing integration. The second signal processing module handles two-dimensional, array-formatted data. That is, it is capable of performing two-dimensional signal processing operations such as filtering, correlation and Fourier transforms.

The general configuration for the sampled analog signal processing system that was designed is illustrated in Figure 1.1. The CCD signal processing module is controlled by the microprocessor through a digital interface. In order to demonstrate that the CCD signal processor is performing its specified functions, it was necessary to also implement an analog signal generator which provides the analog input to the CCD signal processing module. The microprocessor is programmed to control the signal generation in the analog signal generator. Noise may be added to the signal from the signal generator and the combined signal is the input to the CCD signal processing module.

The microprocessor selected to serve as the controller is the Zilog Z-80. For the development of software, the microprocessor was augmented with a dual-floppy disk and a DECWRITER IV terminal.

There are two CCD's that have a digital interface and, thus, lend themselves to microprocessor control. One is the analog/binary correlator (ABC) which is a 512-tap (transversal) delay line with 1-bit (±1) digitally controlled tap weights. The second device is a programmable transversal filter (PTF) which consists of eight 128-tap delay line sections that may be

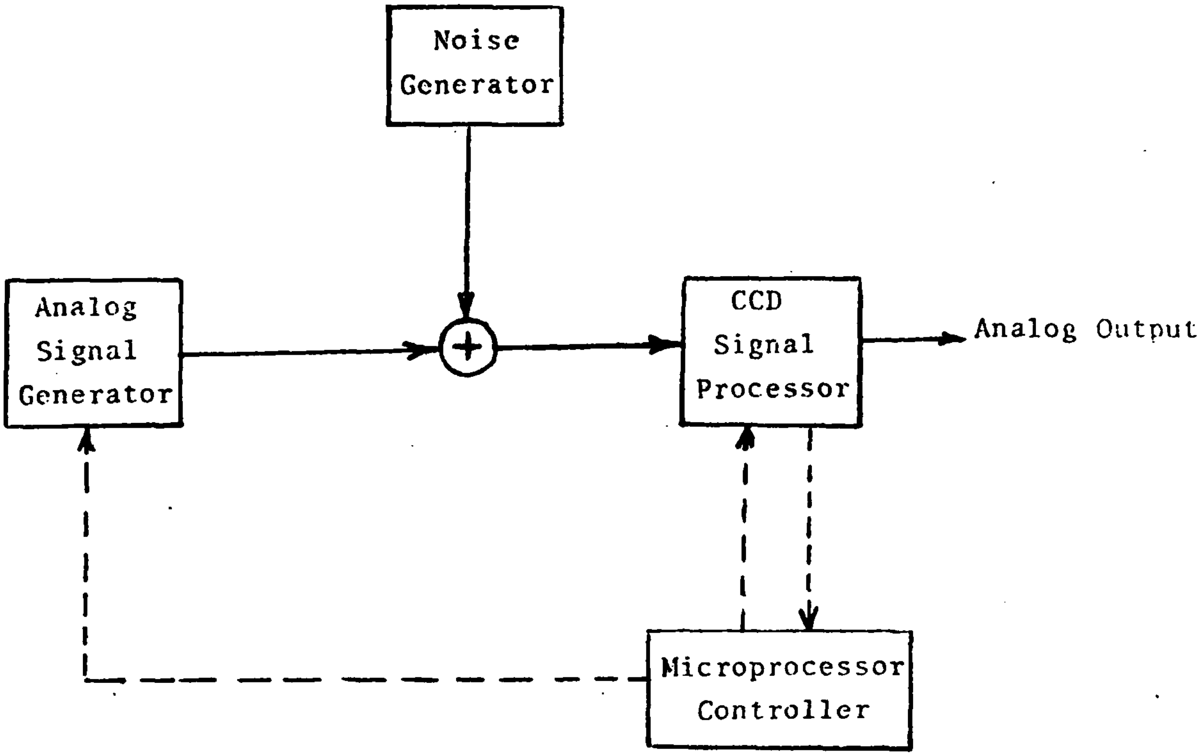


Figure 1.1 General Configuration of Sampled-Analog Signal Processing Module

interconnected in nine different modes. The 1-bit tap coefficients and the mode are dig: ally selected. Thus, the tap coefficients for the ABC and the PTF and the mode for the PTF are controlled by means of the microprocessor.

In order to interface the microprocessor to the ABC and the PTF, it was necessary to design and construct interface circuitry. This circuitry was designed and mounted on a single printed circuit board that is connected to the microprocessor chassis. A direct memory access (DMA) device is used for the purpose of achieving high-speed data transfer from the microprocessor to the PTF and the ABC. The DMA is also used to transfer the digital signal from the microprocessor to the input of the signal generator. A functional block diagram of the system is shown in Figure 1.2.

Aside from the ABC and the PTF, the other sampled analog CCD's have no programmability features. The only potentially interesting connection between these devices and the microprocessor is to feed the output of each device back to the microprocessor via an A/D converter.

The ABC, PTF, RSAM and CTM were provided as GFE. In addition to these devices we purchased two RETICON boards which employ a pair of CCD transversal filters to perform a discrete Fourier transform via the chirp Z-transform algorithm. The line formatted CCD module and the array formatted (two-dimensional) CCD module are to be configured from interconnection of these basic CCD building blocks.

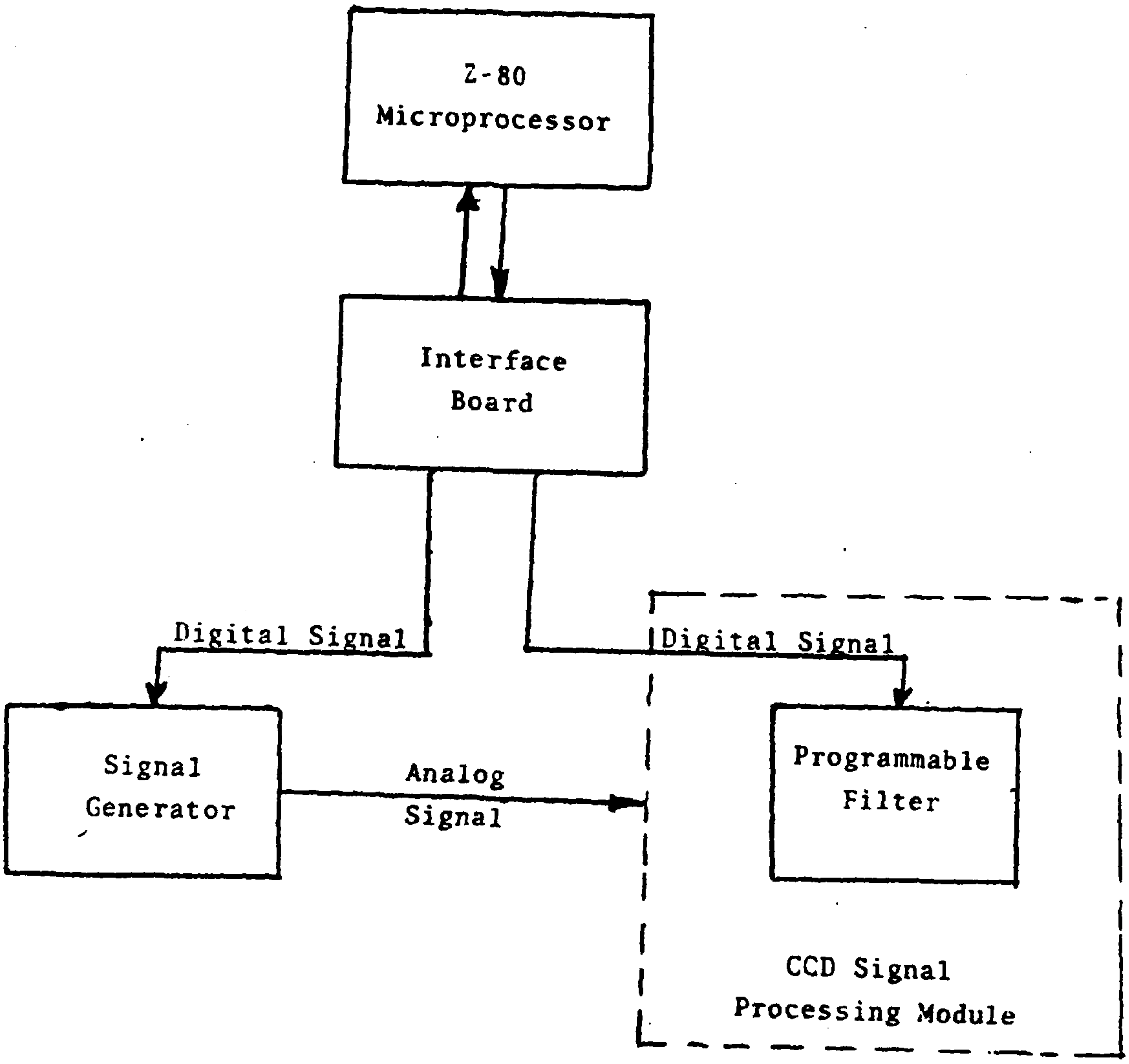


Figure 1.2 Functional Block Diagram of Signal Processing System with the Interface Board

In the following discussion, we briefly describe each of the devices and then consider number of potentially interesting interconnections of the devices. The remainder of the report describes the software and hardware developments performed under this contract.

Refreshable Sampled Analog Memory (RSAM)

The RSAM is a CCD recirculating delay line with the feature that first-order charge transfer losses are recovered and compensated for in the device. The refresh feature allows one to circulate the signal as many as 1,000 times before the signal begins to show some deterioration. The refreshing operation is made possible by following each signal sample by a fixed (trailing) bias signal. The first-order charge transfer losses are collected by the trailing bias charge and the two signals are periodically recombined. Thus, an RSAM with 1,024 sample stages will accommodate 512 signal samples.

Figure 1.3 illustrates a block diagram of the device. In addition to its refresh feature, the device has a nondestructive readout (NDRO) and a pedestal removal feature (dark current subtractor). The latter allows one to remove a fixed amount of charge from each signal sample circulating in the delay line as it passes through this point in the device.

The RSAM is especially suitable for performing coherent integration by simply adding or merging the charges corresponding to the input signal samples with the corresponding charges circulating in the delay line. In order to suppress the buildup of

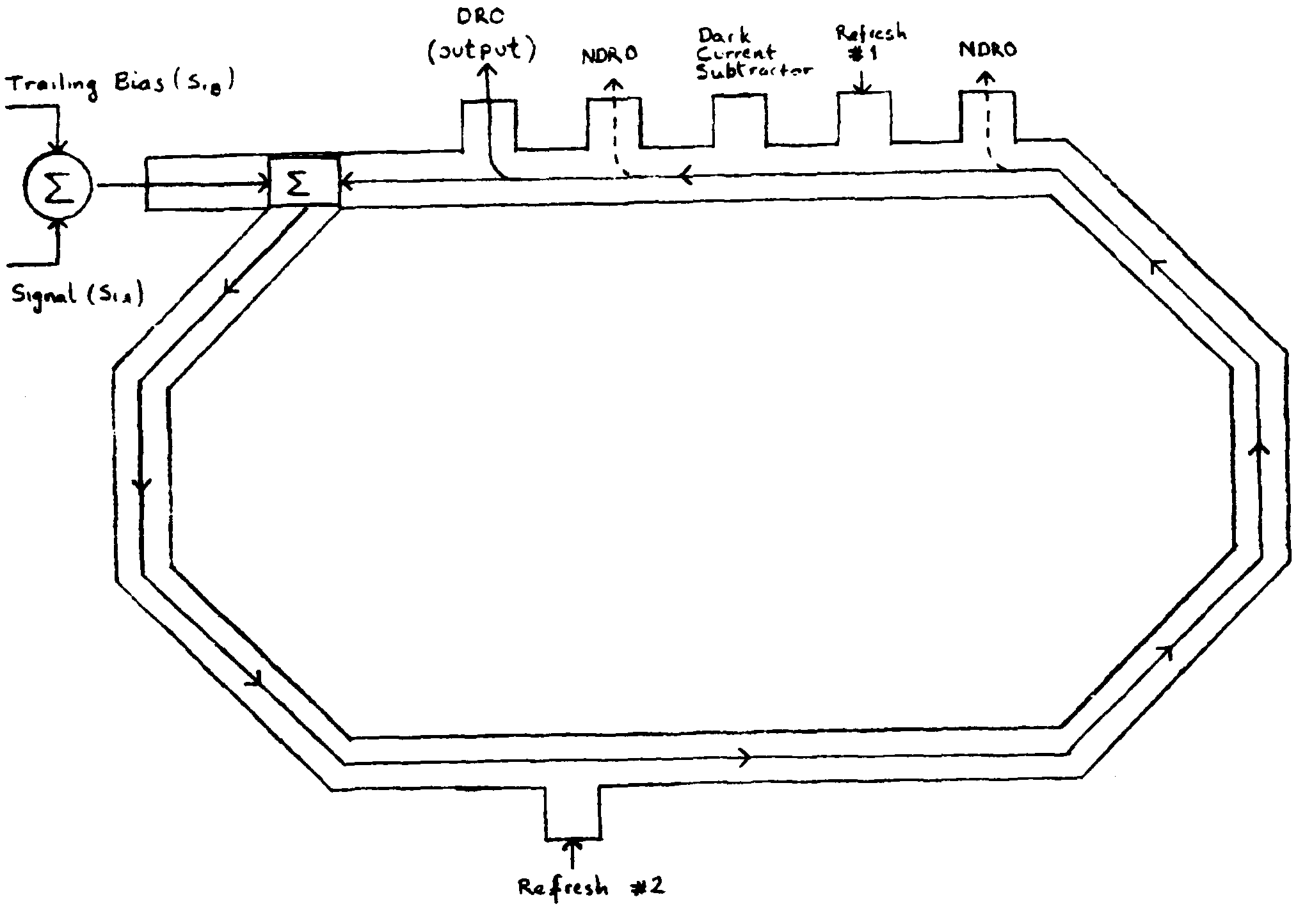


Figure 1.3 Block Diagram of RSAM

DC encountered in coherent integration, the pedestal removal feature can be used. In addition, the device may be used as a simple recirculating delay line or buffer for a set of 512 signal samples.

Support circuitry was constructed for the RSAM and reported in the Interim Report. The circuitry allows us to clock the RSAM at frequencies of 0.625 x 2^n MHz, where n = 0,1,2,3,4. We have tuned and operated the device at clock rates of 1.25 MHz, 2.5 MHz and 5.0 MHz. It operated very well at 1.25 MHz and 2.5 MHz, but its operation at the 5.0 MHz clock rate was marginal. The major problem at 5.0 MHz appears to be the layout of some components in the support circuitry. A modification of the layout of the circuitry will very likely extend the range of operation to 5.0 MHz.

Analog-Binary Correlator (ABC)

The ABC is a 512-stage CCD delay line with a corresponding 512-bit digital shift register reference signal. A block diagram of the ABC is shown in Figure 1.4. The binary reference is loaded into the shift register and latched. When the bit in the latch is a 1, the signal in the corresponding tap is routed to the positive output signal while if the bit is zero, the corresponding signal is routed to the negative output signal as shown in Figure 1.4. Thus, the device can be used as a correlator in a pseudo-noise (PN) spread spectrum communication system or for synchronization purposes. The length of the binary reference signal is variable and selectable over the range from

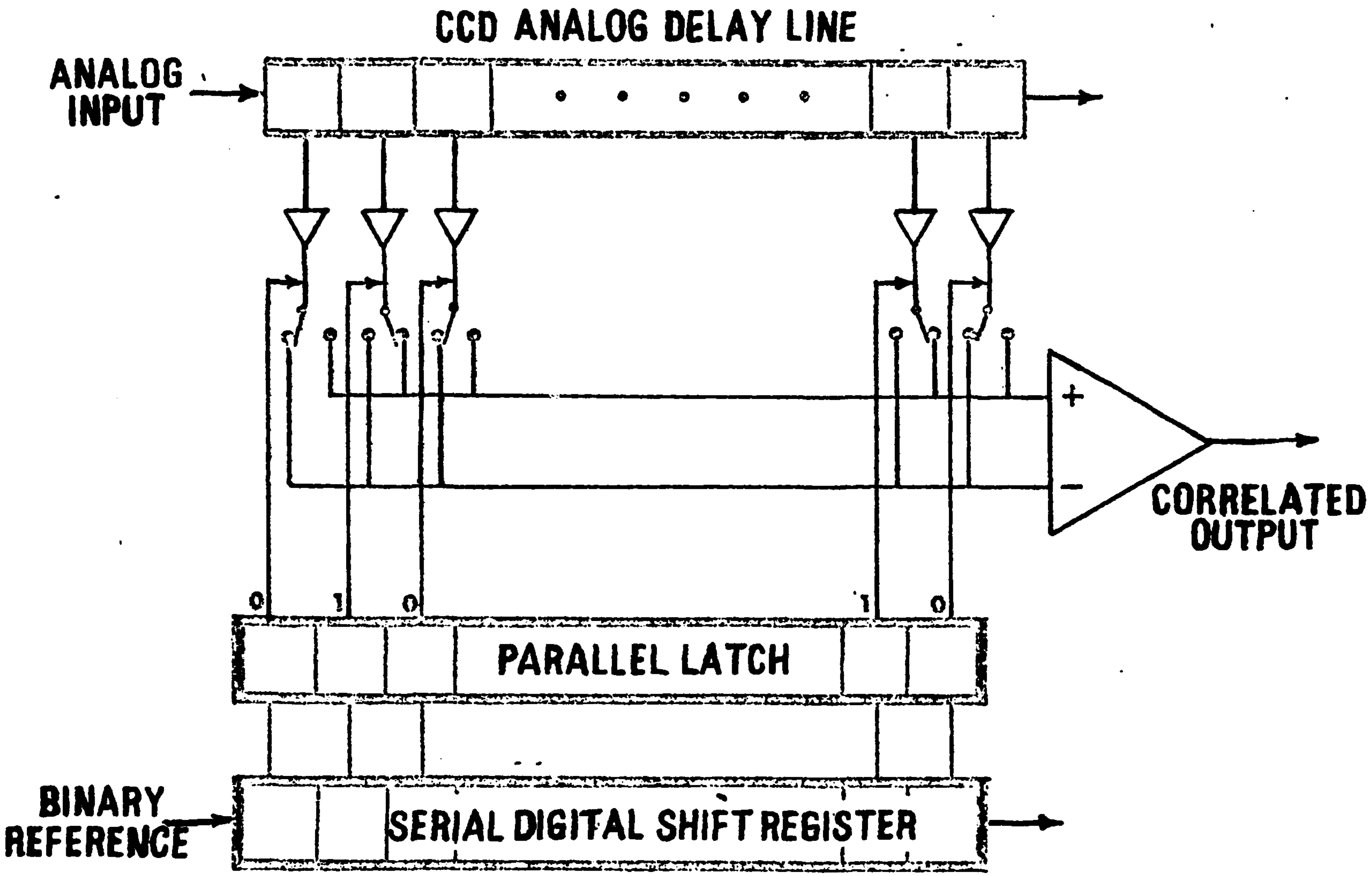


Figure 1.4 Functional Block Diagram of ABC

64 to 512 bits.

The circuitry for the ABC has been constructed and it is described in Section 4 of this report.

Programmable Transversal Filter (PTF)

The PTF consists of eight sections of 128-tap binary-weighted transversal filters. The eight sections can be connected in nine different ways or modes to realize a variety of different types of correlators and filters. Table 1.1 illustrates the nine different modes and the corresponding four-bit binary representation which allows the mode to be selected automatically.

For example, Figure 1.5 illustrates the device as a 128-tap filter with 8-bit coefficient accuracy. In this configuration, the device is suitable as an FIR (finite-duration impulse response) filter with characteristics of the following types: lowpass, bandpass, band-elimination, Hilbert transformer and

Table 1.1

Filter Configuration Versus Input Select Word

	Filter Configuration	Mode	Input Word			
			A	B	C	D
(1)	1024-stage by 1-bit	Single	1	0	0	0
(2)	512-stage by 2 bits	Single	0	1	0	0
(3)	256-stage by 4 bits	Single	1	1	0	0
(4)	128-stage by 8 bits	Single	0	0	1	0
(5)	512-stage by 1 bit	Dual	1	0	1	0
(6)	256-stage by 2 bits	Dual	0	1	1	0
(7)	128-stage by 4 bits	Dual	1	1	1	0
(8)	256-stage by 1 bit	Quad	0	0	0	1
(9)	128-stage by 2 bits	Quad	1	0	0	1

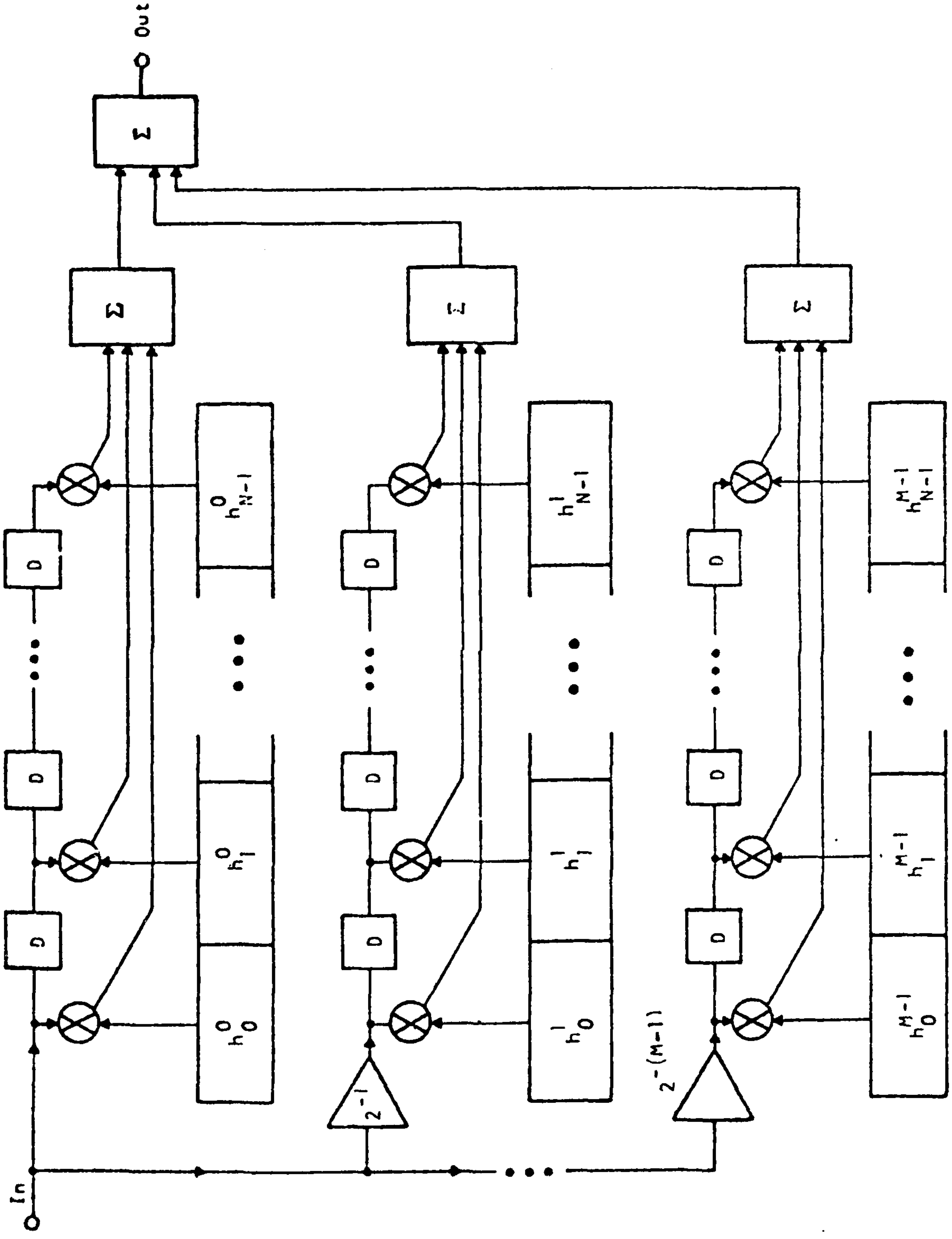


Figure 1.5 PTF Configured as a 128-Tap Filter with 8-Bit Coefficients

differentiation. It may also be used in this mode as an adaptive filter for equalization or notch filtering. In the other modes the device is suitable for performing correlation and synchronization in PN spread spectrum communication systems, radar systems and navigation or ranging systems.

The support circuitry for the PTF has been constructed. It is described in Section 5 of this report.

Corner Turning Memory (CTM)

The CTM is a two-dimensional CCD memory which is appropriate for performing row-column transformations on two-dimensional array data. Figure 1.6 illustrates a block diagram of the CTM. The device has been designed and fabricated as a 32 x 32 memory and a 64 x 64 memory. Its range of frequencies for good operation is 1 MHz to 7 MHz.

The CTM is especially suitable for array-type signal processing operations, such as in the computation of the two-dimensional discrete Fourier transform (DFT) and for Doppler sorting in radar applications. Section 7 describes the use of the CTM in the computation of the two-dimensional DFT.

Reticon Chirp-z Transform Device

The Reticon chirp-z transform consists of two 512-tap CCD transversal filters with preset tap weights whose values are of the form cos π n^2/N and sin π n^2/N where N = 512 and n = 0,1,...,511.

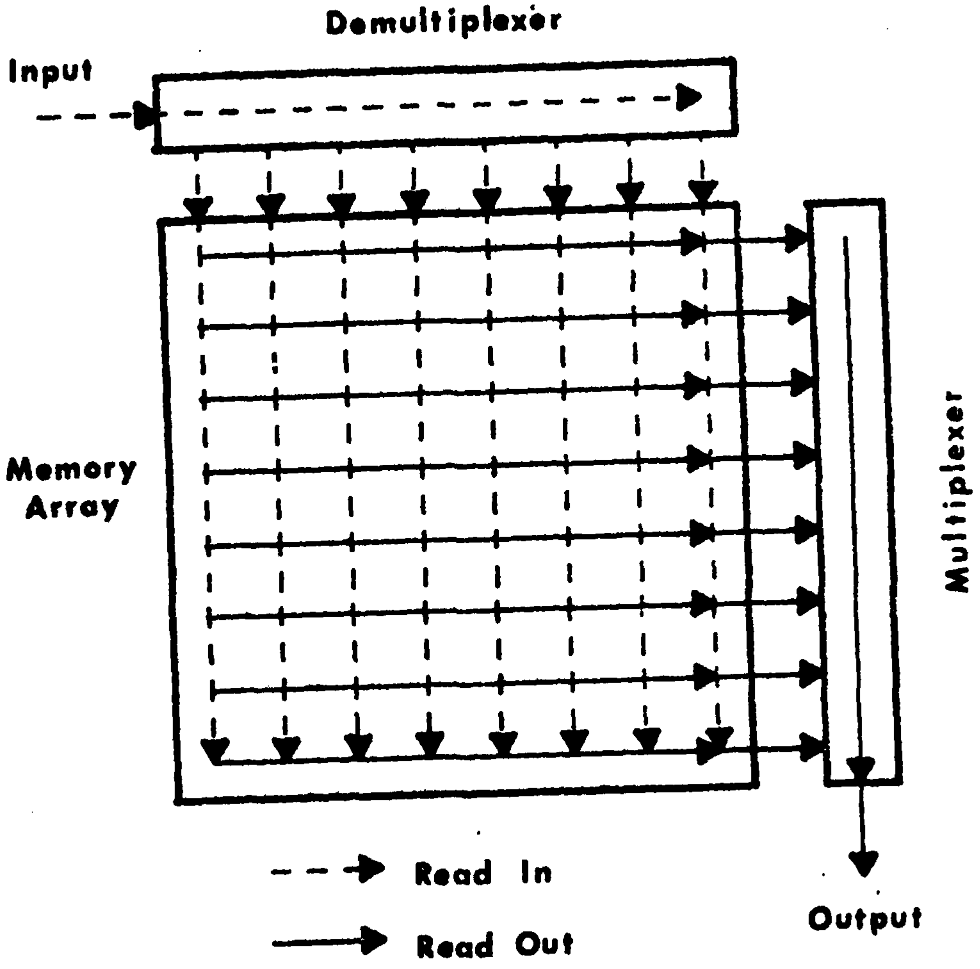


Figure 1.6 Functional Block Diagram of CTM

The CCD's are used to implement the computation of 512-point DFT using the chirp-z transform algorithm. A block diagram illustrating the computations performed in the algorithm is shown in Figure 1.7.

The Reticon devices have a dynamic range of 60 dB and a frequency resolution in the DFT of $F_S/512$, where F_S is the sampling frequency. The latter is limited to below 200 kHz by the support circuitry used to implement the DFT computation.

Section 7 of this report describes the use of the Reticon device in combination with the CTM to implement a two-dimensional DFT.

Some Potential Interconnections

The signal processing modules for handling line formatted and array formatted data were designed to have the capability of performing correlation, synchronization, moving target indication (MTI) and general programmable filtering such as notch filtering and adaptive equalization. Toward this end we designed a line data module which consists of two RSAM's, one of which can be used as an integrator and the other as a long delay line, a PTF, an ABC and a Reticon R5601. The various configurations of these devices are controlled by a number of programs stored in the microprocessor.

Figure 1.8 illustrates a configuration in which two RSAM's and an ABC are connected together to yield a module that can perform correlation or synchronization of the same input signal

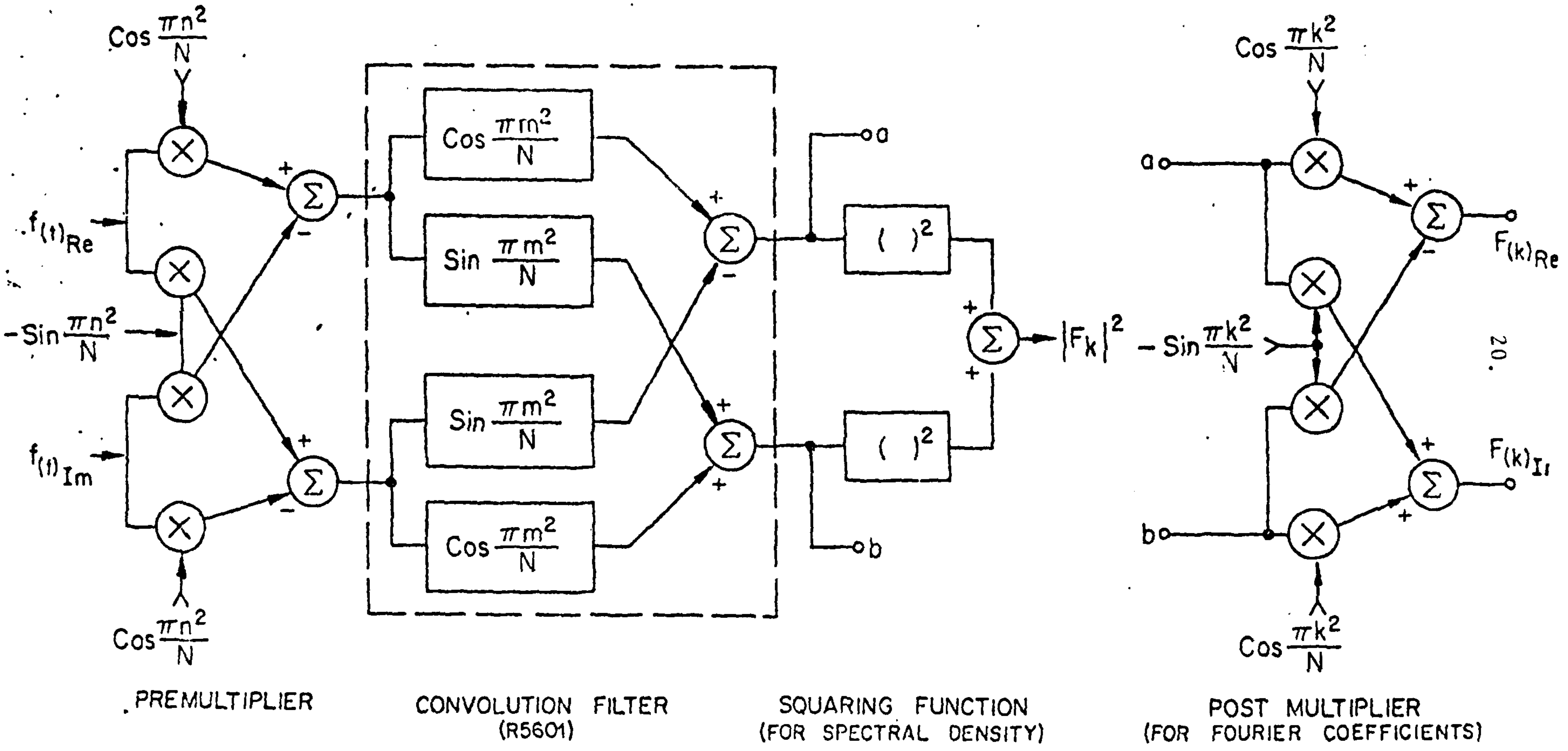

$\text{Cos}\,\frac{\pi n^2}{N}$
$f_{(t)Re}$
$-\text{Sin}\,\frac{\pi n^2}{N}$
$f_{(t)Im}$
$\text{Cos}\,\frac{\pi n^2}{N}$
$\text{Cos}\,\frac{\pi m^2}{N}$
$\text{Sin}\,\frac{\pi m^2}{N}$
$\text{Sin}\,\frac{\pi m^2}{N}$
$\text{Cos}\,\frac{\pi m^2}{N}$
a
b
$(\ )^2$
$(\ )^2$
$|F_k|^2$
$\text{Cos}\,\frac{\pi k^2}{N}$
a
$-\text{Sin}\,\frac{\pi k^2}{N}$
b
$\text{Cos}\,\frac{\pi k^2}{N}$
$F_{(k)Re}$
$F_{(k)I_I}$
PREMULTIPLIER
CONVOLUTION FILTER
(R5601)
SQUARING FUNCTION
(FOR SPECTRAL DENSITY)
POST MULTIPLIER
(FOR FOURIER COEFFICIENTS)

Figure 1.7 Computation of DFT Using the Chirp-z Transform Algorithm

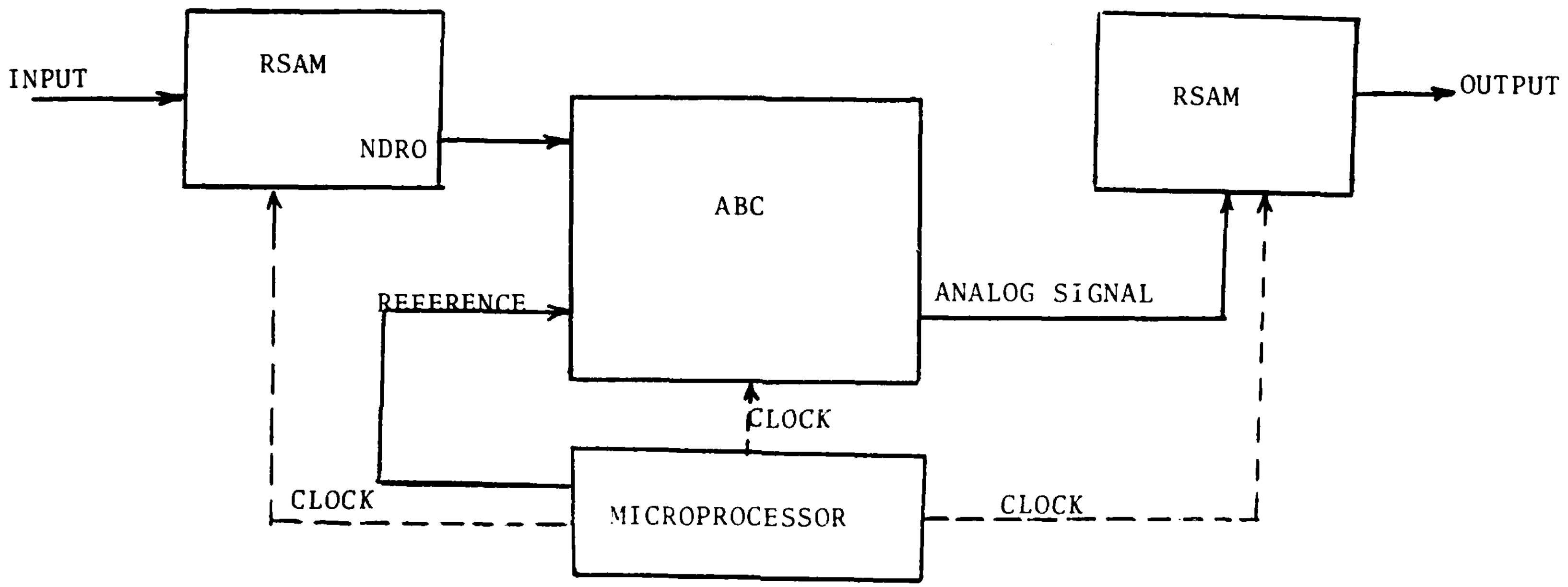


Figure 1.8 A Configuration of CCD's for Processing Line-Formatted Data

with a number of different pseudo-noise (PN) reference signals, and followed by coherent integration (reference 1).

Another possible configuration is illustrated in Figure 1.9. Here, the PTF is used as an adaptive notch filter and the RSAM serves as a delay line. The Reticon device performs a spectral analysis of the input signal and passes the spectral coefficients to the microprocessor through an A/D. On the basis of the spectral estimate, a notch filter impulse response is designed in the microprocessor and the coefficient data is fed to the PTF for realizing the notch filter.

A configuration appropriate for performing two-dimensional, array-type signal processing is shown in Figure 1.10. In this case two CTM's and two Reticon 5601 devices are interconnected so as to perform a two-dimensional discrete Fourier transform (DFT). The operation of this configuration is described in some detail in Section 7.

Figure 1.11 illustrates yet another configuration, using a PTF and two CTM's. With the various modes available in the PTF, this signal processing configuration can be used for multichannel digital communications, multiple correlation, synchronization, and image processing (Reference 1).

Several other configurations were considered in the course of this contract but for the sake of brevity, these will not be described here. Lack of time and manpower prevented us from implementing the various configurations described above.

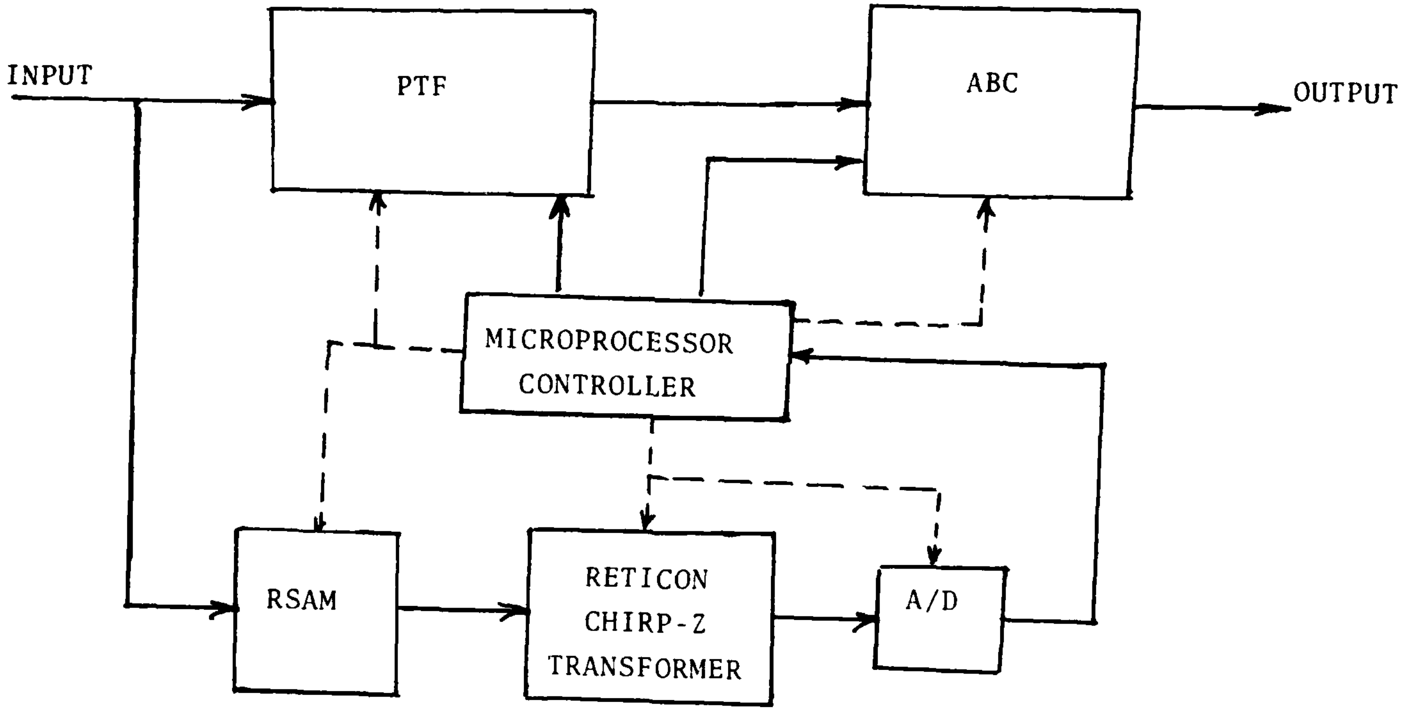


Figure 1.9 A Configuration of CCD's for Performing Adaptive Notch Filtering Followed by PN Correlation

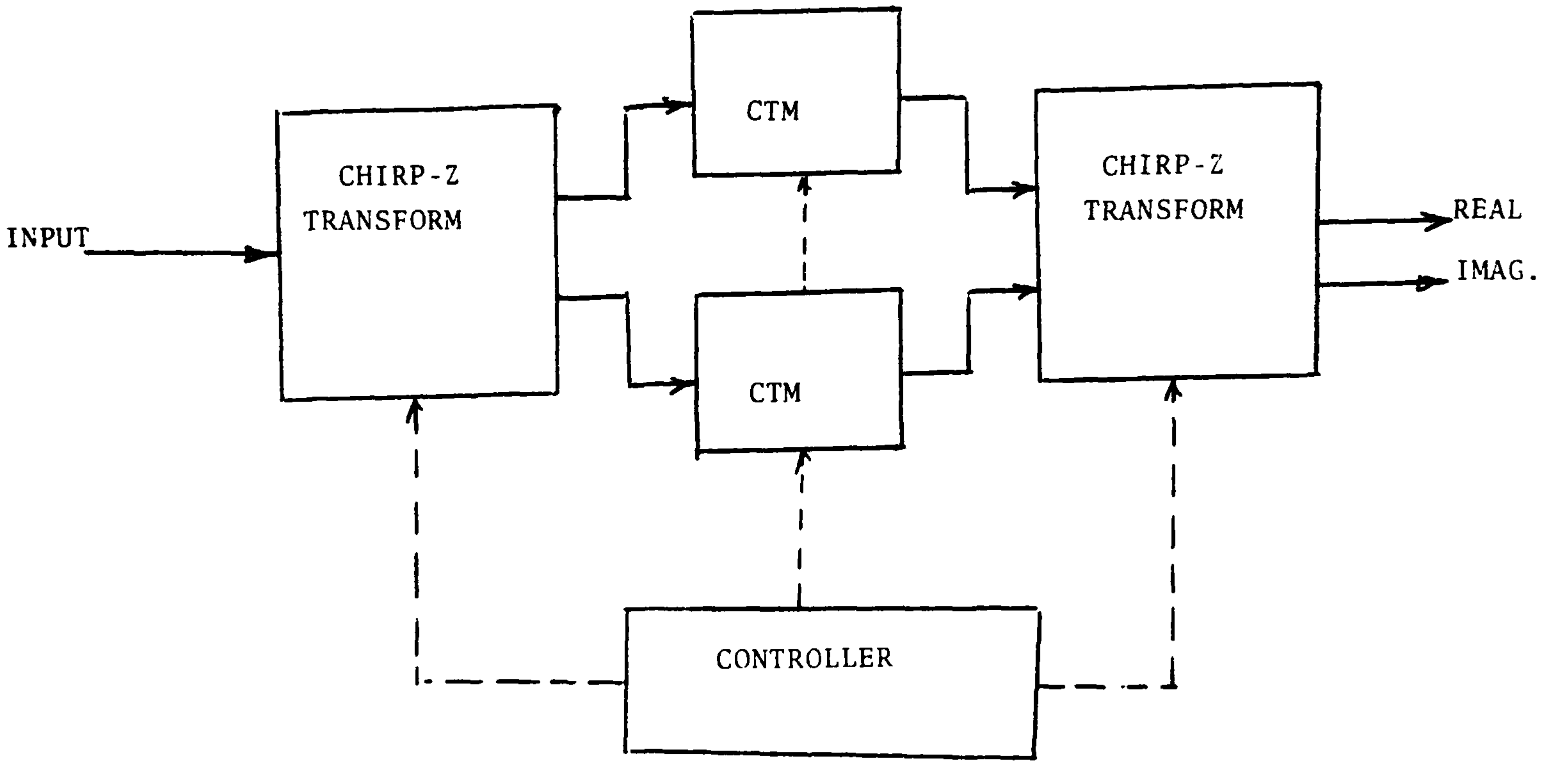


Figure 1.10 A Configuration of CCD's for Computing a Two-Dimensional DFT

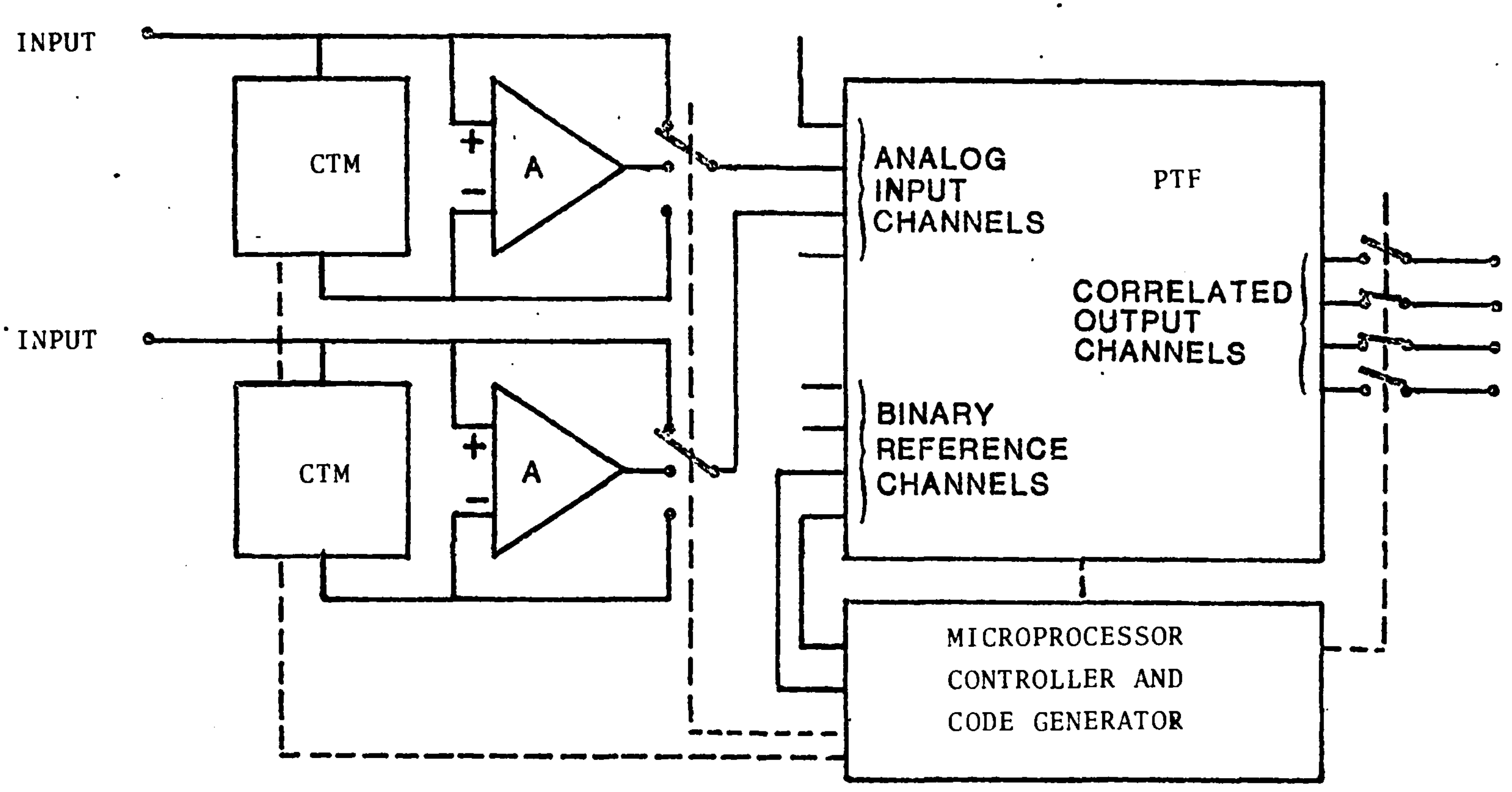


Figure 1.11 A Configuration of CCD's for General Matrix-Type Signal Processing

2. Software Development for the Z-80 Microprocessor

The Z-80 microprocessor configures and controls the signal processing modules. It is also a source of selectable test data to exercise the CCD's described in Section 1.

Most of the software written for the Z-80 microprocessor has been done in FORTRAN. This was the case for programs involving microprocessor functions that do not demand great speed and tight coding. Such programs are used primarily for set-up, i.e., interrogation, parameter selection, and data transformation appropriate for loading the PTF. For microprocessor activities that do demand high speed and tight coding, such as signal generation and DMA control functions and transfers, we programmed in assembly language. The benefits of this approach are primarily:

(a) easier interactive operator control

(b) avoidance of some assembly language housekeeping functions, such as, stack register set-up and load point selection

(c) availability in FORTRAN of almost all logical operations present in assembly language

(d) ability to manipulate 8-bit byte variables

(e) easier linking of main program with subroutines.

Some minor benefits are surrendered when adopting the FORTRAN framework. Whereas, working under control of the object time debugger affords the programmer access to each memory location and the ability to set breakpoints at any instruction, this FORTRAN has no such non-invasive debugging accessories. To overcome this we wrote some binary and hexadecimal print-out routines. Since these are subroutines, they are easily invoked and their removal from a running program is no trouble.

Description and List of Programs

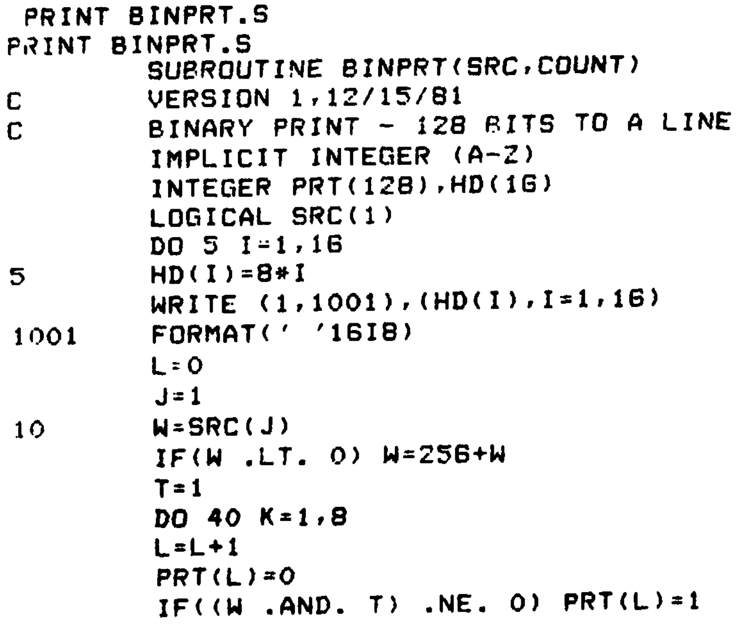
The following is a list of the programs and a brief description of each.

Routine: BINPRT

Language: FORTRAN

Purpose: Utility

This subroutine accepts as input a one-dimensional stream, and prints the result on the terminal. The order of the printed data is from low order bit to high order bit for each byte. Bytes are outputed in succession, up to 128 bits (16 bytes) per line.



```
 PRINT BINPRT.S
PRINT BINPRT.S
        SUBROUTINE BINPRT(SRC,COUNT)
C       VERSION 1,12/15/81
C       BINARY PRINT - 128 RITS TO A LINE
        IMPLICIT INTEGER (A-Z)
        INTEGER PRT(128),HD(16)
        LOGICAL SRC(1)
        DO 5 I=1,16
5       HD(I)=8*I
        WRITE (1,1001),(HD(I),I=1,16)
1001    FORMAT(' '16I8)
        L=0
        J=1
10      W=SRC(J)
        IF(W .LT. 0) W=256+W
        T=1
        DO 40 K=1,8
        L=L+1
        PRT(L)=0
        IF((W .AND. T) .NE. 0) PRT(L)=1
```

```
C       WRITE (1,2000),K,L,W,T,PRT(L)
C2000   FORMAT(' ',5(2X,I8))
40      T=T+T
        IF(L .LT. 128) GO TO 60
        WRITE(1,1000) (PRT(M),M=1,128)
1000    FORMAT(1H ,128I1)
        L=0
60      IF(J .EQ. COUNT) GO TO 90
        J=J+1
        GO TO 10
90      IF(MOD(J,16) .EQ. 0) GO TO 100
        WRITE(1,1000) (PRT(M),M=1,L)
100     RETURN
        END
%
```

Routine: FILCOF

Language: FORTRAN

Purpose: Generate filter coefficients

This subroutine generates binary coefficients to be loaded on to the programmable filter. A total of 1,024 bits are always produced.

The routine is interactive. The operator is requested to input binary strings of up to 24 bits in length and the number of repetitions of the submitted string. Requests are repeated until a 1,024-bit string is produced or until a null string is requested. The latter terminates the process and '0' filler bits are appended until a count of 1,024 is reached.

The program takes care of all counting.

```
PRINT FILCOF.S
PRINT FILCOF.S
        SUBROUTINE FILCOF(FILT)
C       6/2/82. VERSION 1
C       THIS SUBROUTINE LETS THE OPERATOR SPECIFY THE FILTER COEFFICIENTS
C       AS BINARY STRINGS. THE STRING AND ITS REPETITIONS ARE INPUTS.
C       1024 BITS WILL ALWAYS BE PRODUCED. ZERO FILLER BITS WILL BE
```

```
C        PLACED AT THE END.
         IMPLICIT INTEGER (A-Z)
         LOGICAL FILT(1),STRING(24),INS(8),BLANK,ONE,ZERO
         DATA INS/Z'80',64,32,16,8,4,2,1/
         DATA C128/128/,C8/8/,BLANK/' '/,ZERO/'0'/,ONE/'1'/
         DATA C24/24/
C        INITIALIZE
         DO 30 K=1,C128
30       FILT(K)=0
         CONTINUE
         J=1
         K=1
         WRITE(1,1000)
1000     FORMAT(' THIS PROCESS GENERATES THE COEFFICIENTS TO BE SENT TO'/
     1   ' THE FILTER. THERE WILL ALWAYS BE 1024 BITS. THOSE UNSPECI-'/
     2   ' FIED AT THE END WILL BE ZEROS. EACH SUB-STRING MAY BE UP TO'/
     3   ' 24 BITS LONG. THE REPETITIONS OF THIS STRING MUST BE A POS-'/
     4   ' ITIVE NUMBER. A NULL SUB-STRING OR A REPETITION OF 0 ENDS '/
     5   ' PROCESSING.'/)
40       WRITE (1,1001)
1001     FORMAT(' PLEASE ENTER A BINARY STRING OF UP TO 24 BITS ')
         READ(1,1002),(STRING(I),I=1,24)
1002     FORMAT(24A1)
C        DETERMINE LENGTH OF STRING
         DO 70 IX=1,C24
         BIT =STRING(IX)
         IF(BIT .EQ. BLANK) GO TO 100
         STRLEN=IX
         IF((BIT.NE. ONE) .AND. (BIT.NE. ZERO)) GO TO 90
70       CONTINUE
         GO TO 120
90       WRITE(1.1003)
1003     FORMAT(' INVALID CHARACTER.')
         GO TO 40
C        STRLEN NOW HAS STRING LENGTH
100      IF(IX .EQ. 1) GO TO 500
C        NOW OBTAIN REPETITIONS
120      WRITE(1,1004)
1004     FORMAT(' HOW MANY TIMES SHALL THIS STRING BE REPEATED?  ')
         READ (1,1005) REP
1005     FORMAT(I4)
         IF(REP .EQ.0) GO TO 500
C        NOW PLACE BITS INTO DESTINATION STRINGS
         DO 180 IRX=1,REP
C        SINGLE STRING CONTROL
         DO 160 JX=1,STRLEN
         IF(STRING(JX) .EQ. ZERO) GO TO 130
         FILT(J)=FILT(J) .OR. INS(K)
130      IF(K .EQ. C8) GO TO 150
         K=K+1
         GO TO 160
```

```
150     IF(J .EQ. C128) GO TO 500
        J=J+1
        K=1
160     CONTINUE
C       CHECK REPETITIONS
180     CONTINUE
        GO TO 40
500     RETURN
        END
%
```

Routine: RNGEN

Language: Z-80 Assembly

Purpose: Generate random bits

This subroutine generates maximal length shift register sequences for polynomials up to degree 31. It is in a form to be used as a FORTRAN subroutine.

A set-up entry uses the degree of the polynomial, initial state of the shift register and the tap positions to initialize the program.

A second entry is called to generate the pseudo-random bit stream. This entry requires the number of 8-bit bytes to be generated, up to 128, and the destination location. Successive entries to this second entry proceed from the conditions in effect when this part of the routine was last used.

If it is desired to institute a new set of initial conditions the first entry must be used again.

```
PRINT RNGEN.S
PRINT RNGEN.S
;       TITLE   "MAXIMAL LENGTH SRS,TO DEGREE 31"
;THE INITIAL STATE OF THE SHIFT REGISTER WILL BE IN 'DATA'
```

```
;THE POLYNOMIAL WILL BE IN 'POLY',BOTH RIGHT JUSTIFIED.
;VERSION 1,6/16/81, DO NOT INCREMENT HL
        GLOBAL  SRSET,RNGEN
SRSET   NOP                     ;USE ONE BYTE ONLY
        LD      A,(HL)
        LD      (DEG),A         ;STORE POLY DEGREE
        LD      (LODAT),BC      ;SAVE PTR TO INITIAL STATE OF S.R.
        EX      DE,HL
        LD      DE,POLY
        LD      BC,4
        LDIR
        LD      HL,(LODAT)
        LD      DE,DATA
        LD      BC,4
        LDIR                    ;POLY AND INITIAL STATE NOW SET
        XOR     A               ;ZERO THE 5TH BYTE OF THE POLY
        LD      (POLY+4),A
        LD      C,0             ;C WILL HAVE MMOD(DEG,8)
        LD      A,(DEG)         ;POLYNOMIAL DEGREE
RPT1    CP      8
        JP      M,INJP
        SUB     8
        INC     C
        JR      RPT1            ;REG A HAS COUNT OF BITS LEFT
INJP    LD      B,0
        INC     C
        INC     C               ;LOOP COUNT IS ACTUAL COUNT
        LD      HL,INLOOP
        LD      (HL),C
        DEC     C               ;UNDO 3RD INSTRUCTION UP
        LD      HL,DATA+4
        CP      0               ;clear carry flag
        SBC     HL,BC
        LD      (STRT),HL
        LD      E,1             ;SET UP EXTRACTOR
INLP    CP      0
        JR      Z,SETXT
        SLA     E
        DEC     A
        JR      INLP
SETXT   LD      HL,EXT
        LD      (HL),E
        RET                     ;END OF INITIALIZATION
;MAIN ENTRY, CALLED BY CALL(BYTES,DESTINATION )
RNGEN   NOP
        LD      A,(HL)
        LD      (LEN),A         ;BYTES TO BE GENERATED
        PUSH    DE
        POP     IY              ;POINTS TO DELIVERY VECTOR
        LD      A,80H           ;FIFTH BYTE WILL GET NEW BITS
        LD      (DATA+4),A
        LD      A,(LEN)         ;OUTPUT DATA COUNTER IN REG. C
        LD      C,A
```

```
NEWBIT  LD      E,0                 ;START 4 CYCLESTO GET NEXT OUT BIT
        LD      H,0                 ;WILL HAVE LAST CARRY BIT
        LD      HL,INLOOP
        LD      B,(HL)
        LD      IX,(STRT)
WSHFT   LD      D,(IX+0)            ;PROCESS NEXT BYTE
        LD      A,(IX+5)
        AND     D                   ;DATA ANDED WITH POLY TO REG A
        XOR     E                   ;'1' BITS IN SAME POSITION CANCEL
        LD      E,A                 ;INTERIM RESULTS TO REG E
        XOR     A                   ;SET TO SAVE NEXT CARRY BIT
        RR      H                   ;OLD CARRY TO CY
        RR      D                   ;SHIFT DATA BIT 0 TO CY REG
        ADC     A,A                 ;SAVE CARRY IN
        LD      H,A                 ;REG H
        LD      (IX+0),D            ;STORE SHIFTED BYTE
        INC     IX                  ;CARRY BITS NOT AFFECTED
        DJNZ    WSHFT
        RR      H                   ;WHEN 8 BITS DONE CARRY OCCURS
        JR      NC,SLP              ;IF 8 BITS NOT DONE REPEAT
        LD      (IY+0),D            ;REGISTER D HAS NEW BYTE
        INC     IY
        LD      A,80H               ;INITIALIZE
        LD      (DATA+4),A
        DEC     C
SLP     XOR     A
        XOR     E                   ;DETERMINE PARITY  IN REG F
        JP      PE,NUTH             ;AND SET HIGH ORDER DATA BIT
        LD      IX,(STRT)           ;IN HIGH ORDER BYTE
        LD      A,(EXT)
        OR      (IX+0)
        LD      (IX+0),A
NUTH    XOR     A                   ;SEE IF OUTPUT IS DONE
        ADD     A,C
        JP      NZ,NEWBIT
        RET
DATA    DEFS    5                   ;DATA SHIFT REGISTER PLUS ONE BYTE
POLY    DEFS    5                   ;POLYNOMIAL COEFFICIENTS
DEG     DEFB    -1                  ;DEGREE OF POLYNOMIAL
LEN     DEFB    4                   ;NUMBER OF OUTPUT BYTES
EXT     DEFB    1                   ;LEADING BIT THAT IS SET
STRT    DEFW    2                   ;STARTING ADDRESS FOR LOOP
INLOOP  DEFB    -5                  ;INNER LOOP(BYTES IN POLY)
LODAT   DEFS    2                   ;PTR TO INITIAL STATE OF S.R.
        END
%
```

Routine: HEXPRT

Language: FORTRAN

Purpose: Utility

This subroutine converts a one-dimensional array of bytes to a string of ASCII hexadecimal bytes for printing. The left-most hexadecimal character is followed by the right-most hexadecimal character in the printing format.

Sixteen bytes, 32 hexadecimal characters, are printed on each line.

```
PRINT HEXPRT.S
PRINT HEXPRT.S
           SUBROUTINE HEXPRT(HEXB,COUNT)
C CONVERT A VECTOR OF HEX BYTES AND PRINT 16 TO A LINE
C VERSION 1,7/9/81
           IMPLICIT INTEGER (A-Z)
           LOGICAL HEXB(1),PRT(32),CHAR(16)
           DATA CHAR/Z'30',Z'31',Z'32',Z'33',Z'34',Z'35',Z'36',Z'37',
        1  Z'38',Z'39',Z'41',Z'42',Z'43',Z'44',Z'45',Z'46'/
           I=1
           J=1
100        T=HEXB(I)
           IF(T .LT. 0) T=256+T
           IX=MOD(T,16)+1
           PRT(J+1)=CHAR(IX)
           IX=T/16 +1
           PRT(J)=CHAR(IX)
           J=J+2
           IF(J .LT. 32) GO TO 140
110        WRITE (1,1000) (PRT(K),K=1,32)
1000       FORMAT(1H ,16(2X,2A1))
           J=1
140        IF(I .EQ. COUNT) GO TO 150
           I=I+1
           GO TO 100
150        LST=MOD(COUNT,16)*2
           IF(LST .EQ. 0) GO TO 160
           WRITE (1,1000) (PRT(K),K=1,LST)
160        RETURN
           END
%
```

Program: HAFFIL

Language: FORTRAN

Purpose: Test correct loading of output ports and D/A operation through the DMA

This program loads the status words, modes and data to the output ports and devices in order to test the hardware configuration in an orderly manner. After each transfer that changes the system state, the program pauses and permits the engineer to verify that the action occurred. Each action is identified.

The last step causes the DMA to continually feed the D/A through a FIFO. The display can be monitored on a scope.

```
PRINT HAFFIL.S
PRINT HAFFIL.S
C       HAFFIL.S 5/26/82 SPECIAL TESTER FOR FTP
C       STATUS WORDS ARE SENT TO THE OUTPUT PORTS BEFORE AN DATA IS SENT
C       AFTER STATUS IS OUT, SEND MODE THEN FILTER COEFFICIENTS, THEN
C       SEND NEW STATUS WORD, AND FOLLOW WITH DMA TRANSFER TO D/A.
        IMPLICIT INTEGER (A-Z)
        LOGICAL RECT(128),RECTRN(130),S,ANSWER,MODEBT,MODADD
        LOGICAL STATDA,STATFI,STAADD,SIGMAD,BZER
        LOGICAL A,B,DUM,CON
        DATA DMAADD/Z'E3'/,DISDMA/Z'83'/,FI1024/Z'EF'/,MODADD/Z'F3'/
        DATA STATDA/Z'3C'/,STATFI/Z'7C'/,STAADD/Z'E7'/
        DATA SIGMAD/Z'F2'/,BZER/Z'00'/
        DATA RECT/128*Z'11'/,RECTRN/16*Z'01',112*Z'00'/
        DATA A,B/Z'AA',Z'55'/,CON/'C'/
        DATA COUNT/128/,UPDOWN/+1/,C127/127/,S/'S'/,MODE/1/
        DATA MODEBT/Z'01'/
        DATA SOFTRS/Z'FE'/
C       SOFTWARE RESET
        CALL OUT(SOFTRS,BZER)
C       DISABLE THE DMA
        CALL OUT(DMAADD,DISDMA)
C       SET UP STATUS WORD FOR FILTER TRANSFER
10      CALL OUT(STAADD,STATFI)
        WRITE (1,1004)
1004    FORMAT(' -7C-, THE STATUS WORD,HAS BEEN SENT TO FILTER AT -E7-'/
     1  ' PRESS C.R. TO CONTINUE.')
        READ (1,1005) DUM
```

```
1005        FORMAT(1A1)
C           SEND MODE TO FILTER
C           MODEBT = MODE
            CALL OUT(MODADD,MODEBT)
            WRITE (1,1006)
1006        FORMAT(' -01-,THE MODE,HAS BEEN SENT TO FILTER AT -F3- '/,
         1  ' PRESS C.R. TO CONTINUE')
            READ (1,1005) DUM
            WRITE (1,1000)
1000        FORMAT(' FILTER COEFFICIENTS  ')
            CALL HEXPRT(RECT,COUNT)
C           NOW LOAD THE FILTER
30          DO 50 I=1,1
            CALL LOADFL(RECT,COUNT,UPDOWN,FI1024)
50          CONTINUE
            WRITE (1,1007)
1007        FORMAT(' THE FILTER HAS BEEN LOADED AT -EF-'/
         1  ' TO CONTINUE PRESS -C- ,ELSE WE RELOAD FILTER    )
            READ (1,1005) DUM
            IF(DUM .NE. CON) GO TO 30
C           NOW GET THE FIFO TO THE SIGNAL GENERATOR GOING
C           SET UP STATUS FOR D TO A
            CALL OUT(STAADD,STATDA)
            WRITE (1,1008)
1008        FORMAT(' -3C-,THE STATUS WORD, HAS BEEN SENT TO THE SIGNAL '/
         1  ' GENERATOR AT -E7-'/
         2  ' PRESS C.R. TO CONTINUE')
            READ (1,1005) DUM
C           SET MODE IN SIGNAL GENERATOR
            CALL OUT(SIGMAD,BZER)
            WRITE (1,1009)
1009        FORMAT(' ,-00-,THE MODE,HAS BEEN SENT TO THE SIGNAL GENERATOR'/
         1  ' AT -F2-.'/
         2  ' PRESS C.R. TO CONTINUE')
            READ (1,1005) DUM
            CALL DMASET(RECTRN,C127)
            WRITE (1,1010)
1010        FORMAT(' DMA GOING, 16 BYTES OF -01- 112 OF -00-')
100         WRITE(1,1002)
1002        FORMAT(' TO RETURN TO THE SYSTEM TYPE AN -S- '/,
         1  ' ELSE WE START OVER.  ')
            READ (1,1003)ANSWER
1003        FORMAT(1A1)
            IF(ANSWER .NE. S) GO TO 10
            CALL OUT(DMAADD,DISDMA)
            STOP
            END
%
```

Program: REFORM

Language: FORTRAN

Purpose: To rearrange the coefficients of the programmable transversal filter for proper loading

This subroutine transforms the filter coefficients in the Z-80 memory so that the filter hardware ordering requirements are met. The calling routine furnishes the filter mode and the addresses of the source bytes and destination bytes. 128 bytes, (1,024 bits), are always processed.

```
PRINT REFORM.S
PRINT REFORM.S
C       GENERAL REFORMATTING PROGRAM TO PREPARE COEFFICIENTS FOR THE
C       PROGRAMMABLE FILTER.  REQUIRES -SRC- THE SOURCE BYTES, -DST-
C       THE DESTINATION BYTES, AND THE MODE.
C       VERSION 1, 1/28/82
        SUBROUTINE REFORM(SRC,DST,MODE)
        IMPLICIT INTEGER (A-Z)
        DIMENSION INPOS(8,3),BITNUM(8),DISP(8)
        LOGICAL SRC(1),DST(1),XTR(8)
        LOGICAL BYTOUT
        DATA XTR/1,2,4,8,16,32,64,Z'80'/
        DATA INPOS/1,129,257,385,513,641,769,897,
     1            1023,1024,767,768,511,512,255,256,
     2            1021,1022,509,510,1023,1024,511,512/
        DATA DISP/1,-2,-4/
        DATA C8/8/,C128/128/
C**********************************************************************
C       SET UP INITIAL SOURCE BIT NUMBERS
        DO 30 L=1,C8
30      BITNUM(L)=INPOS(L,MODE)
C       K CONTROLS BYTE OUTPUT, ALWAYS 128 BYTES
        DO 300 K=1,C128
C       J CONTROLS BITS WITHIN A BYTE,(DESTINATION BYTES)
        BYTOUT=0
        DO 100 J=1,C8
        BYTNUM=BITNUM(J)/C8
        BITPOS=MOD(BITNUM(J),C8)
        IF(BITPOS .EQ. 0) GO TO 60
        BYTNUM=BYTNUM+1
        GO TO 80
60      BITPOS=C8
```

```
80        IF((SRC(BYTNUM) .AND. XTR(BITPOS)) .NE. 0)
     X    BYTOUT =BYTOUT .OR. XTR(J)
100       CONTINUE
C         OUTPUT BYTE COMPLETE
          DST(K) = BYTOUT
C         SET UP BIT POSITIONS FOR NEXT BYTE
          DO 200 L=1,C8
200       BITNUM(L) =BITNUM(L) +DISP(MODE)
C         END OF LARGE LOOP
300       CONTINUE
          RETURN
          END
%
```

Program: PRTEXC

Language: FORTRAN

Purpose: To exercise the output ports

This routine transmits a user supplied data byte to a user selected output port a specified number of times. It can be repeated as often as desired or restarted with a new selection of parameters. It was written to validate correct hardware operation.

```
PRINT PRTEXC.S
PRINT PRTEXC.S
C         PRTEXC.S
C         OUTPUT PORT EXERCIZER
C         VERSION 1,3/17/82
          IMPLICIT LOGICAL (A-Z)
          DIMENSION INCHAR(2)
          INTEGER COUNT,MAX,J
          DATA YES /'Y'/,EXIT/'E'/
          DATA MAX/30000/
C*****************************************************************
          WRITE (1,1000) MAX
1000      FORMAT(' OUTPUT PORT EXERCIZER'/,
     1    ' THIS PROGRAM WILL OUTPUT A DATA BYTE TO A SELECTED OUTPUT '/,
     2    ' PORT. THE USER CHOOSES THE NUMBER OF TIMES THE DATA IS '/,
     3    ' WRITTEN.  THE OUTPUT PORT ADDRESS AND THE OUTPUT DATA ARE '/,
     4    ' SPECIFIED BY TWO HEXADECIMAL CHARACTERS.  THE NUMBER OF  '/,
     5    ' TIMES THE DATA IS WRITTEN IS SPECIFIED BY A DECIMAL'/,
     6    ' INTEGER LESS THAN OR EQUAL TO'I5//)
```

```
30      WRITE (1,1001)
1001    FORMAT(' WHAT IS THE OUTPUT PORT ADDRESS?   ')
        READ (1,1002) (INCHAR(J),J=1,2)
1002    FORMAT(2A1)
        CALL BYTFRM(INCHAR,PORTAD,ERRIND)
C       WRITE (1,2000)
C2000    FORMAT(' THE PORT ADDRESS FROM HEXPRT'/)
C       CALL HEXPRT(PORTAD,1)
        IF(ERRIND .NE. 1) GO TO 50
        WRITE(1,1003)
1003    FORMAT(' THE OUTPUT PORT ADDRESS IS INVALID'/)
        GO TO 30
50      WRITE (1,1004)
1004    FORMAT(' WHAT IS THE OUTPUT DATA?  ')
        READ (1,1002) (INCHAR(J),J=1,2)
        CALL BYTFRM(INCHAR,DATOUT,ERRIND)
C       WRITE (1, 2001)
C2001    FORMAT(' THE OUTPUT DATA FROM HEXPRT'/)
C       CALL HEXPRT(DATOUT,1)
        IF(ERRIND .NE. 1) GO TO 70
        WRITE (1,1005)
1005    FORMAT(' THE OUTPUT DATA IS INVALID'/)
        GO TO 50
70      WRITE(1,1006)
1006    FORMAT(' TIMES DATA SHOULD BE WRITTEN, (DECIMAL)    ')
        READ (1,1007) COUNT
1007    FORMAT(I5)
        IF(COUNT .LE. MAX) GO TO 90
        WRITE (1,1008) MAX
1008    FORMAT(' THE OUTPUT COUNT IS TOO LARGE, IT MUST BE < OR = 'I5/)
        GO TO 70
90      DO 110 J=1,COUNT
        CALL OUT(PORTAD,DATOUT)
110     CONTINUE
        WRITE (1,1009)
1009    FORMAT(' AGAIN?    ')
        READ (1,1010) REPLY
1010    FORMAT(1A1)
        IF(REPLY .EQ. YES) GO TO 90
                WRITE (1,1011)
1011    FORMAT(' TO EXIT TYPE AN ''E'' ,ELSE WE START ANEW.    ')
        READ (1,1010) REPLY
        IF(REPLY .NE. EXIT) GO TO 30
        STOP
        END
%
```

Program: BINLTR

Language: FORTRAN

Purpose: Utility

This subroutine accepts as input a one-dimensional array of bytes, converts the data to a binary stream and prints the results. The order of the printed data is from high-order to low-order for each byte. Bytes are output in succession, up to 128 bits (16 bytes) per line.

```
PRINT BINLTR.S
PRINT BINLTR.S
        SUBROUTINE BINLTR(SRC,COUNT)
C       6/2/82. BINARY PRINT ROUTINE,LEFT TO RIGHT
        IMPLICIT INTEGER (A-Z)
        LOGICAL SRC(1),INS(8),PRT(128),W,ZERO,ONE
        INTEGER HD(16)
        DATA INS/Z'80',64,32,16,8,4,2,1/
        DATA ZERO/'0'/,ONE/'1'/,C128/128/
C       HEADINGS
        DO 10 I=1,16
        HD(I)=8*I
10      CONTINUE
        WRITE(1,1000),(HD(I),I=1,16)
1000    FORMAT(' '16I8)
        L=0
        J=1
20      W=SRC(J)
        DO 40 K=1,8
        L=L+1
        PRT(L)=ZERO
        IF((W .AND. INS(K)) .NE. 0) PRT(L)=ONE
40      CONTINUE
        IF(L .LT. C128) GO TO 80
        LINLEN =C128
60      WRITE(1,1001),(PRT(L),L=1,LINLEN)
1001    FORMAT(' '128A1)
        L=0
80      IF(J .EQ. COUNT) GO TO 90
        J=J+1
        GO TO 20
90      IF(L .EQ. 0) GO TO 200
        LINLEN=L
        GO TO 60
200     RETURN
        END
%
```

Program: DMASET

Language: Z-80 Assembly

Purpose: Set-up DMA and FIFO stack to load the signal generator

This is a FORTRAN callable subroutine specialized to work with the DMA and FIFO at fixed output port addresses. The hardware is set up to load the signal generator through the FIFO. When the FIFO stack has been emptied to a triggering level, the DMA is set in action again to load the FIFO.

The subroutine requires the memory address of the data and the number of bytes to be transmitted at each call.

```
PRINT DMASET.S
PRINT DMASET.S
;       SUBROUTINE DMASET
;       VERSION 2,10/22/81. CORRECT BLOCK LENGTH SETUP
; THIS MODULE SETS UP THE DMA TO TRANSFER DATA FROM THE Z80 MEMORY TO THE
; PROGRAMMABLE FILTER. THE DATA FIRST GOES TO A FIFO STACK WHICH IN TURN
; FEEDS THE FILTER. THE ADDRESS OF THE SOURCE AND THE LENGTH OF THIS BYTE
; DATA STRING ARE IN THE CALLING SEQUENCE OF THE CALLING PROGRAM.
;
        GLOBAL  DMASET
DMAADR  EQU     0E3H            ;DMA BUS ADDRESS
FIFOAD  EQU     0EBH            ;FIFO BUS ADDRESS
DMASET  NOP
        LD      (DATSTR),HL     ;SET UP STARTING DATA ADDRESS
        LD      A,(DE)          ;SET UP BLOCK LENGTH
        LD      (BLKLEN),A
        INC     DE
        LD      A,(DE)
        LD      (BLKLEN+1),A
        LD      C,DMAADR        ;DMA ADDRESS TO REG. C FOR I/O
        LD      HL,DMABEG
        LD      B,DMAEND-DMABEG ;NUMBER OF BYTES TO BE SENT TO THE DMA
        OTIR                    ;SET UP DMA
        LD      A,087H          ;DMA ENABLED,WRG COMMAND
        OUT     (C),A
        NOP
        RET
;       DMA COMMANDS AND PARAMETERS
```

```
DMABEG  DEFB    083H            ;WR6,DISABLE DMA
        DEFB    0C3H            ;WRG,RESET-DO SIX TIMES
        DEFB    0C3H
        DEFB    0C3H
        DEFB    0C3H
        DEFB    0C3H
        DEFB    0C3H            ;ALL SHOULD BE RESET
WR0     DEFB    079H            ;REGISTER GROUP 0
DATSTR  DEFW    -1              ;STARTING ADDRESS OF DATA SOURCE
BLKLEN  DEFW    -2              ;LENGTH OF DATA VECTOR,BYTES
WR1     DEFB    054H            ;REGISTER GROUP 1
        DEFB    06AH            ;SET UP IORQ TIMING,2 CYCLE TIMING
WR2     DEFB    078H            ;REG. GROUP 2
        DEFB    04AH            ;2 CYCLE,WRITE EARLY END
WR4     DEFB    0C5H            ;REG. GRP. 4
        DEFB    FIFOAD          ;FIFO BUS ADDRESS
WR6     DEFB    0CFH            ;LOAD B ADDRESS,RESET BLOCK COUNT
WR5     DEFB    0A2H            ;AUTO RESTART
WR0A    DEFB    05H             ;MAKE'A' THE SOURCE,'B' THE DESTINATION
WR6A    DEFB    0CFH            ;LOAD A ADDRESS,RESET BLOCK COUNT
DMAEND  EQU     $               ;END OF DMA COMMANDS
        END
%
```

Program: LOADFL

Language: Z-80 Assembly

Purpose: Load the programmable transversal filter when addressed as an output port

This is a FORTRAN callable subroutine. The caller supplies the data address, the number of bytes to be sent to the filter and the output port address. Up to 128 bytes of data will be output at each call.

```
PRINT LOADFL.S
;       SUBROUTINE LOADFL
;       SRC     16 BIT ADDRESS OF THE DATA IN RAM
;       VERSION 1,2/4/82, MODIFICATION OF LOADOD OF 12/15/81.
;       V1.,SPECIALIZE TO LOAD FILTER. FILTER ADDRESS WILL COME FROM
;       V1., CALLER. ENABLING OF THE DEVICE IS NOT NECESSARY.
;       COUNT   THE NUMBER OF BYTES TO BE TRANSMITTED
;       UPDOWN  IF +1 THE DATA IS STORED IN ASCENDING ORDER
;               ELSE IF -1 IN DESCENDING ORDER.
;       DEVADD  FILTER ADDRESS
;*********************************************************************
        GLOBAL  LOADFL
LOADFL  NOP
        LD      (SRC),HL        ;STORE SOURCE ADDRESS
        LD      A,(DE)          ;BYTE COUNT,LIMIT TO LOW ORDERBYTE,255
        LD      (COUNT),A
        PUSH    BC
        POP     HL              ;POINTS TO LIST OF 2 ADDRESS POINTERS
        LD      IX,UPDOWN
        LD      B,2
LOOP    LD      E,(HL)          ;LOW ORDER OF PARAMETER POINTER
        INC     HL
        LD      D,(HL)          ;HIGH ORDER OF PARAMETER POINTER
        LD      A,(DE)          ;LOW ORDER OF PARAMETER
        LD      (IX),A
        INC     IX
        INC     HL
        DJNZ    LOOP
; PARAMETERS ARE NOW STORED
        LD      HL,(SRC)        ;SET UP SOURCE ADDRESS
        LD      A,(COUNT)       ;COUNT TO B
        LD      B,A
        LD      A,(DEVADD)      ;OUTPUT DATA PORT ADDRESS TO C
        LD      C,A
        LD      A,(UPDOWN)
        ADD     A,0
        JP      M,DECR
;IF IN INCREASING ORDER
        OTIR
        JP      LAST
;IF IN DECREASING ORDER
DECR    OTDR
LAST    RET
;PARAMETER ADDRESSES
SRC     DEFW    -2              ;SOURCE ADDRESS
COUNT   DEFB    -3              ;BYTE COUNT
UPDOWN  DEFB    -4              ;STORAGE ORDER
DEVADD  DEFB    -8              ;OUTPUT PORT BUS ADDRESS FOR DATA
        END
%
```

Program: BYTFRM

Language: FORTRAN

Purpose: Utility

This subroutine converts two ASCII characters into one 8-bit byte.

The calling routine provides the input characters, a location for the output byte and a location for an error indicator. An error is noted if any of the input ASCII characters are not in the set of hexadecimal characters.

```
PRINT BYTFRM.S
PRINT BYTFRM.S
         SUBROUTINE BYTFRM(INCHAR,OUTBYT,ERRIND)
C        VERSION 1,3/17/82. CONVERT TWO INPUT HEXADECIMAL CHARACTERS
C        TO ONE BYTE, OUTBYT. ERRIND = 0, NO INPUT ERRORS; ERRIND = 1 ,
C        INPUT CHARACTER IN ERROR.
         IMPLICIT LOGICAL (A-Z)
         INTEGER K,J
         DIMENSION INCHAR(2),HEX(2,16),ALPHAB(16)
         DATA ALPHAB/'0','1','2','3','4','5','6','7','8','9','A','B',
      1  'C','D','E','F'/
         DATA HEX/0,0,Z'10',Z'01',Z'20',Z'02',Z'30',Z'03',Z'40',Z'04',
      2  Z'50',Z'05',Z'60',Z'06',Z'70',Z'07',Z'80',Z'08',Z'90',Z'09',
      3  Z'A0',Z'0A',Z'B0',Z'0B',Z'C0',Z'0C',Z'D0',Z'0D',Z'E0',Z'0E',
      4  Z'F0',Z'0F'/
C*********************************************************************
         OUTBYT = 0
         DO 100 J=1,2
         W=INCHAR(J)
         DO 50 K= 1,16
         IF(W .NE. ALPHAB(K)) GO TO 50
         OUTBYT = HEX(J,K) .OR. OUTBYT
         GO TO 100
50       CONTINUE
         ERRIND = 1
         RETURN
100      CONTINUE
         ERRIND = 0
         RETURN
         END
%
```

Program: SETREG

Language: FORTRAN

Purpose: To initialize the shift register and tap position register for the RNGEN program

This subroutine is given the tap positions, for a 32-bit long register, that are to be set to '1'. It is used for setting the initial state of the shift register and the tap vector for the maximal-length shift register program.

The master routine which calls this subroutine informs the user as to which register is being set up. Up to 10 taps, from positions 1 to 32, are permitted. Position 1 is the low order.

```
 PRINT SETREG.S
PRINT SETREG.S
      SUBROUTINE SETREG(TAPS,POSIT,DEST,IND)
C     2/17/82 VERSION 1
C     GIVEN TAP POSITIONS OR BIT POSITIONS,LOW ORDER AT THE RIGHT,
C     THIS SUBROUTINE WILL INITIALIZE THE SHIFT REGISTER, DEST, OF
C     DIMENSION 4, WHERE DEST(1) HAS THE HIGH ORDER BITS. THERE MAY
C     BE UP TO 10 TAPS.
C********************************************************************
      IMPLICIT INTEGER (A-Z)
      DIMENSION POSIT(10)
      LOGICAL DEST(4),EXT(8),IND
      DATA EXT/1,2,4,8,16,32,64,Z'80'/
      DATA JUPL/4/,C32/32/,C8/8/,C5/5/
C     ZERO THE DEST VECTOR BEFORE INSERTING BITS
      DO 50 J=1,JUPL
50    DEST(J)=0
C     CHECK THAT TAP POSITIONS ARE IN BOUNDS
      DO 70 J=1,TAPS
      IF((POSIT(J).LE.0) .OR. (POSIT(J).GT. C32)) GO TO 200
70    CONTINUE
C     TAP POSITIONS ARE O.K.
      IND=.TRUE.
C     NOW INSERT THE BITS
      DO 100 J=1,TAPS
      BITPOS=MOD(POSIT(J),C8)
      BYTNUM=POSIT(J)/C8
      IF(BITPOS .NE. 0) GO TO 80
      BITPOS=C8
      GO TO 90
```

```
80        BYTNUM=BYTNUM+1
90        BYTNUM=C5-BYTNUM
          DEST(BYTNUM)=DEST(BYTNUM) .OR. EXT(BITPOS)
100       CONTINUE
C         NORMAL RETURN
          GO TO 210
C         ERROR IN INPUT
200       IND=.FALSE.
210       RETURN
          END
%
```

3. Microprocessor Interface to CCD Modules

This section describes the microprocessor-CCD interface board. It is an expanded version of the description given previously in the interim report.

The purpose of the microprocessor-CCD interface board is to provide a flexible and easily reconfigurable high-speed interface between the microprocessor and the CCD modules. The interface board was designed to provide for transfer of data tables in memory directly to the CCD modules. The data tables are generated by the microprocessor during the initialization phase. A direct memory access controller, DMAC, has been incorporated to support this type of transfer. The reason for generating the data in the form of tables is that the CPU will not be able to do it in real time, but instead will serve the role of a system controller once an experiment is started.

The general structure of the board is divided into four functional blocks. These are: the DMA controller (and the Board Control Word), the bus interface logic which includes the simultaneous transfer logic, the two I/O ports and the high-speed FIFO buffer with its associated logic. Figure 3.1 illustrates the general configuration of the interface board.

The DMAC is programmed by writing data into its control registers. One of several functional modes can be selected. These are (1) search only, (2) sequential transfer, and (3) simultaneous transfer (some external logic is needed for this mode). Variables such as block length, transfer address, match

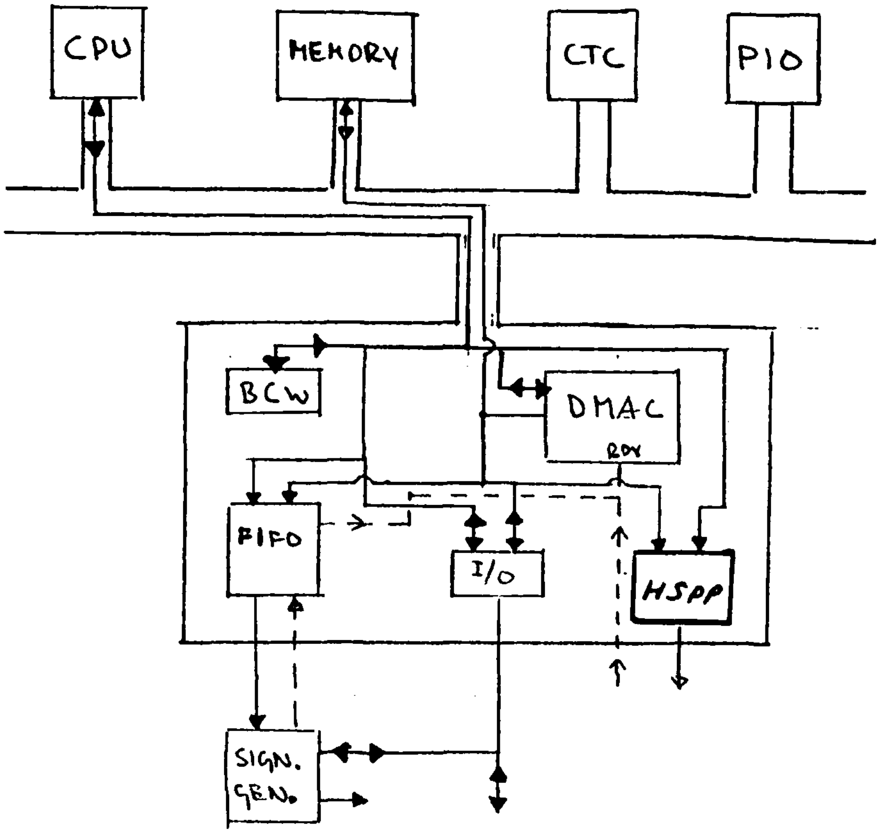


Figure 3.1 General Configuration of Interface Board

pattern, interrupt generation, block/byte transfer, and cycle length are also under program control. The manner in which this is done and the functional description of the DMAC hardware are contained in the technical literature provided by Zilog on the DMA [reference 2].

The high-speed transfer logic is located on the board and extensive signal buffering is needed towards the system bus. Some signals are bidirectional and, hence, we have used bidirectional buffers. The fact that both the CPU and the DMAC can be bus master complicates the logic controlling the buffer direction.

In the case of simultaneous DMA transfer, which is programmed as search only mode, a write signal is created at the same time as memory read is generated, so that the actual transfer is taking place in the same cycle. Thus, simultaneous transfer increases the transfer rate by a factor of two. At this time, the DMAC is the bus master and, hence, the bus interface logic will turn the direction of the data buffer accordingly.

To make the interface circuitry as flexible as possible, necessary changes in the hardware when changing transfer mode and direction is also under software control. For this purpose, a control word, called the Board Control Word (BCW), has been established, which the CPU can write to as well as read from.

There are two I/O ports on the board. One of them is an ordinary bidirectional I/O port covering four I/O addresses. This port is to be used for control signalling. The other is a special purpose write only port, that we call the high-speed parallel port (HSPP). It has the following features. It can be either transparent or latched, the selection being made via the BCW. It can also be addressed in an ordinary fashion or selected as a simultaneous port by means of the BCW.

The FIFO buffer is intended to provide the signal generator with data and handshaking signals to the DMAC. Its presence provides for a minimal software overhead during operation. The surrounding logic is needed for synchronization and for warning the DMAC that the FIFO is nearly empty.

Besides the abovementioned blocks, address-decoding logic is also present on the board to decode the 32 I/O addresses dedicated to the interface board. Below is one example of how the function of the interface board can be set up.

- The CPU loads the DMAC with commands and addresses, activates the required hardware function by writing to the BCW and then enables the DMAC.
- The DMAC requests the system bus for starting a transfer to the FIFO buffer, for instance. When the FIFO is filled the DMAC will sense this and return the bus to the CPU, only to request it again when the FIFO needs more data.
- The CPU can now do some calculations, transfers or general control of the experiment. Alternatively, the CPU can reprogram the DMAC to perform a different transfer while it is waiting for the FIFO flag signal for reloading. The timing requirements during this flip-flop type of

action are very sensitive, and errors can easily occur if caution is not taken.

Bus Interface Logic

The purpose of the bus interface logic is to control the direction of the bidirectional data buffers connecting the interface board's bus to the system data bus. The direction depends on whether the CPU or the DMAC is bus master and whether a read or write operation is taking place.

In general, when the CPU is controlling the bus, the buffer data flows toward the interface board except during a read operation or an interrupt acknowledge cycle. The interrupt acknowledge is issued from the CPU in order to let the requesting peripheral place an interrupt vector on the data bus. When the DMAC is controlling the bus the data flow is from the interface board to the system bus in all cases except during a read from a device not located on the board.

The Boolean expressions for controlling the direction of the data buffer is:

$$F = R \cdot C \cdot \overline{Y}_4 + \overline{R} \cdot B + B \cdot C + \bar{R} \cdot \bar{C} \cdot I$$

where

$I = \overline{MI} + \overline{IORQ} + \overline{IEI} + IEO$

R = read

C = chip select

B = bus acknowledge

$\overline{Y}_4$ = port I/O group address

$\overline{MI}$ = machine cycle one

IORQ = input/output request

IEI = interupt enable in

IEO = interupt enable out

Besides the data buffers, there are also buffers for control and address signals. The address lines connecting the DMAC with the system bus are used only when the DMAC is bus master. Hence, a set of unidirectional three-state buffers are needed. They are simply turned on by the bus acknowledge signal.

Some of the control signals are unidirectional and are continuously buffered. Others are bidirectional and have a buffer having its direction controlled by bus acknowledge. Thus, they lead out toward the system bus when the DMAC is bus master and otherwise they lead in the reverse direction.

Special consideration must be given to buffer direction when a simultaneous transfer is taking place. This topic is treated below.

Simultaneous Transfer Logic

Due to the fact that the DMAC is not supporting this mode, additional hardware is needed to generate the missing signals. Specifically, when transfering from memory to a port (fixed address) the DMAC is programmed to perform a search (read operations) and in order to write into the port a write signal has to be generated from the read signal. The ordinary bidirectional control signal buffer is disabled during simultaneous transfer and write, read and memory request are either bypassing the buffer or being especially generated as described above. When performing a simultaneous transfer from a port to memory, the logic level of the read signal leading into the bus interface logic must be inverted in order to enable the data buffers to be directed out to the system bus.

The simultaneous transfer logic is only active when both the DMAC is bus master, i.e., bus acknowledged is active, and the bus control word (BCW) bit S_7 is set at the same time.

The Board Control Word (BCW)

The BCW provides a method for changing the hardware function under software control. Besides being writable it can also be read by the CPU which is useful for debugging and makes it unnecessary to keep a copy in a memory location. Some bits of the BCW determine the data flow while some others are controlling the output ports to enable several of them to be tied together in order to minimize the use of wires on the system bus. The BCW format and bit definition are as follows:

S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

S_7: A "1" will enable the simultaneous logic. NOTE: The DMAC has to be disabled before changing this bit; also, before re-enabling, the DMAC has to be programmed for simultaneous transfer.

S_6: A "0" will connect the DMAC ready pin to the FIFO buffer logic. A "1" will connect the DMAC ready pin to the parallel port. In both cases will ready active, i.e., low, enable the DMAC to resume transfer in the burst mode.

S_5: Not yet used.

S_4: A "1" will be one of the conditions to enable the output of the parallel port.

S_3: A "1" means 8-bit correlation, and is also another condition to enable the parallel port. A "0" means 1-bit correlation. This bit and the following was originally defined to be used in conjunction with the ABC CCD device. Some kind of redefinition will take place in the future.

S_2: A "1" will connect (enable) the high speed parallel port (HSPP) to the correlator. A "0" will connect a skewed version of the test signal to the correlator (ABC).

S_1: Not yet used.

S_0: A "1" will allow the parallel port to be transparent during simultaneous transfer, while a "0" will cause it to be latched.

High Speed FIFO Port

The high speed FIFO port was especially designed to provide the signal generator with high speed data. Extreme care was taken to avoid loss of data and other errors. It consists of a 128-byte deep FIFO preceded by a latch. The latch was introduced to hold the last written byte because, during simultaneous transfer, the timing of the handshake could not give the DMAC a stop transfer, i.e., not ready, early enough to prevent it from doing another transfer. Consequently, without the hold register, the last transferred byte would be lost if the FIFO did not get a read between the last and the next to the last byte transferred.

The timing which accomplishes that is of a pipeline model. This means that the register is loaded during the later part of a transfer cycle and loaded into the FIFO during the earlier part of the next transfer cycle. To make the timing consistent, the different modes have the same general timing. There is a difference, however, between simultaneous and sequential transfer (including CPU transfer). The difference is in the timing signal that triggers some of the logic functions (timing is considered below).

There are a few D-type flip-flops connected to form a finite state machine. One is used only for reset purposes, while another is used to represent the zero state to which one can only return from a reset. The purpose of the reset latch is to prevent the signal generator from reading from the FIFO before it is completely filled initially. This is done in order to create a known starting state of the system. One flip-flop is simply detecting whether the FIFO is full or not and when it is, it will generate the load signal, clearing the ready signal, the reset latch and load the counter, which is explained below. Two D-type flip-flops are connected in a "bite your own tail" fashion to function as a storage element (state) indicating that a byte is ready for transfer from the latch into the FIFO. Their output signal is called data available, DA. Finally, another D-latch monitors both DA and the input ready, IR, of the FIFO and generates a write pulse to it on the appropriate clock pulse.

The counter is interconnected with the FIFO in such a way that for each byte shifted out it will be incremented until carry out is generated. This causes a latch-up of the same counter. Each time the FIFO gets into the full state this counter will be loaded, by the load signal from the D-latch mentioned above, with a preset value selectable via a DIP-switch, representing the number of bytes that will be shifted out before more data is requested from the DMAC.

Other Onboard Ports

At the present time there are two more distinct ports on the interface board, the High Speed Parallel Port (HSPP) and the general I/O port. The latter is actually used for two different tasks, each having its own address (two more addresses can be assigned to this port). The two tasks consist of giving a mode command to the signal generator and to the PTF. This and the high-speed parallel port are both of a three-state type so they can be connected in parallel either to each other or with other ports.

The HSPP was designed especially to support the simultaneous transfer mode of the DMA. It is, however, only a writeable port but its output stage can be controlled from the BCW in two ways; the output enable is governed by bits S_2, S_3 and S_4 and during simultaneous transfer the latch can be selected to be transparent by bit S_0.

The corresponding Boolean expression controlling the gating function of the latch is:

$$F = \overline{\bar{B} \cdot \bar{A} + \bar{B} \cdot \bar{W}_1 + \bar{W}_1 \cdot \bar{T}}$$

where $\bar{W}_1 = \bar{W} + \bar{A} \cdot \overline{B \cdot S_6 \cdot S_7}$

A = port address

B = bus acknowledge

W = write

T = transparent

S_6 and S_7 = status bits

This option was incorporated to give maximum flexibility and minimum delay for high-speed and critical timing situations. In the non-transparent mode it functions as a latch. Hence, it can, for example, be read by the I/O port if the two are connected in parallel.

Address Decoding

The Z-80 microprocessor has an I/O addressing range of OO_H - FF_H* (256 distinct addresses) of which some already are reserved at system level. The MCB board (where the CPU is located) has decoded the address space CO_H - CF_H (a total of 32 as follows:

MDC: CF_H - $D3_H$

CTC: $D4_H$ - $D7_H$

*H stands for hex, i.e., the base of the numbers is 16.

PIO: $D8_H$ - DB_H

USART: DC_H - DF_H

Not used: CO_H - CE_H

This means that the available address space is OO_H - BF_H and EO_H - FF_H where we choose to reserve the latter 32 addresses for the interface board in the following manner.

Eight I/O groups were defined to correspond to specific types of devices and ports. Then four subgroups enable each I/O group to respond to and serve four local addresses.

I/O Group	Address Group	Function/device supported
$\overline{Y}_0$:	EO_H - $E3_H$	DMAC
$\overline{Y}_1$:	$E4_H$ - $E7_H$	Control, status
$\overline{Y}_2$:	$E8_H$ - EB_H	FIFO buffer
$\overline{Y}_3$:	EC_H - EF_H	Parallel port
$\overline{Y}_4$:	FO_H - $F3_H$	I/O port
$\overline{Y}_5$:	$F4_H$ - $F7_H$	Clock rate
$\overline{Y}_6$:	$F8_H$ - FB_H	Correlation skew
$\overline{Y}_7$:	FC_H - FF_H	Software controlled resets

Subgroup

$\overline{X}_0$: -

$\overline{X}_1$: -

$\overline{X}_2$: -

$\overline{X}_3$: -

$\overline{X}_4$: 0 4 8 C

$\overline{X}_5$: 1 5 9 D

$\overline{X}_6$: 2 6 A E

$\overline{X}_7$: 3 7 B F

As an example of how to use this decoding scheme, the following is the way in which it is used at the present.

	Address
DMAC 1	$\overline{Y}_0 \cdot \overline{X}_7 = E3_H$
BCW	$\overline{Y}_1 \cdot \overline{X}_7 = E7_H$
FIFO 1	$\overline{Y}_2 \cdot \overline{X}_7 = EB_H$
High Speed Parallel port 1 (HSPP)	$\overline{Y}_3 \cdot \overline{X}_7 = EF_H$
I/O port - signal generator mode	$\overline{Y}_4 \cdot \overline{X}_6 = F2_H$
I/O port - PTF mode	$\overline{Y}_4 \cdot \overline{X}_7 = F3_H$
Software Reset	$\overline{Y}_7 \cdot \overline{X}_6 = FE_H$
Software Master Reset	$\overline{Y}_7 \cdot \overline{X}_7 = FF_H$

Timing Description

Most of the timing specifications are identical to those that are common for most microcomputer systems. Due to the fact that we are using a Z-80 DMAC and development system, and are adapting our hardware to this architecture, all the timing specifications for the system bus and the DMAC are valid and can be found in the Zilog literature.

There are a few timing considerations that need some attention. The interface board is designed to handle two-cycle simultaneous transfer using a system clock of 4 MHz (Z-80A). Figure 3.2 illustrates the timing for the two-cycle simultaneous transfer. It might even be possible to connect the board to a 6 MHz version of Z-80, but an investigation of its external timing is advisable before trying.

The timing diagram for two-cycle sequential transfer is shown in Figure 3.3a. During sequential DMA transfer it is important that the DMAC is programmed in the following way: The I/O port has to be programmed with $\overline{\text{IORQ}}$ and $\overline{\text{WR}}$ ending early (half a cycle) and the memory port has to be programmed with $\overline{\text{IORQ}}$ ending early and normal end for $\overline{\text{MEMRQ}}$ and $\overline{\text{RD}}$.

During simultaneous transfer the DMAC memory port has to be programmed in the same way as above.

There is a difference in the timing for the FIFO buffer depending on whether one is performing a CPU output cycle or a sequential DMA transfer on one hand and a simultaneous DMA transfer on the other. The difference is on which edge or clock phase the SI and IR signals are generated, as shown in Figure 3.3b In the case of simultaneous transfer, the sequence of controlling edges is negative, positive and negative going, while in the other two cases it is the opposite (compare Figures 3.2, 3.3a and 3.3b). BCW bit S_7, which activates the simultaneous

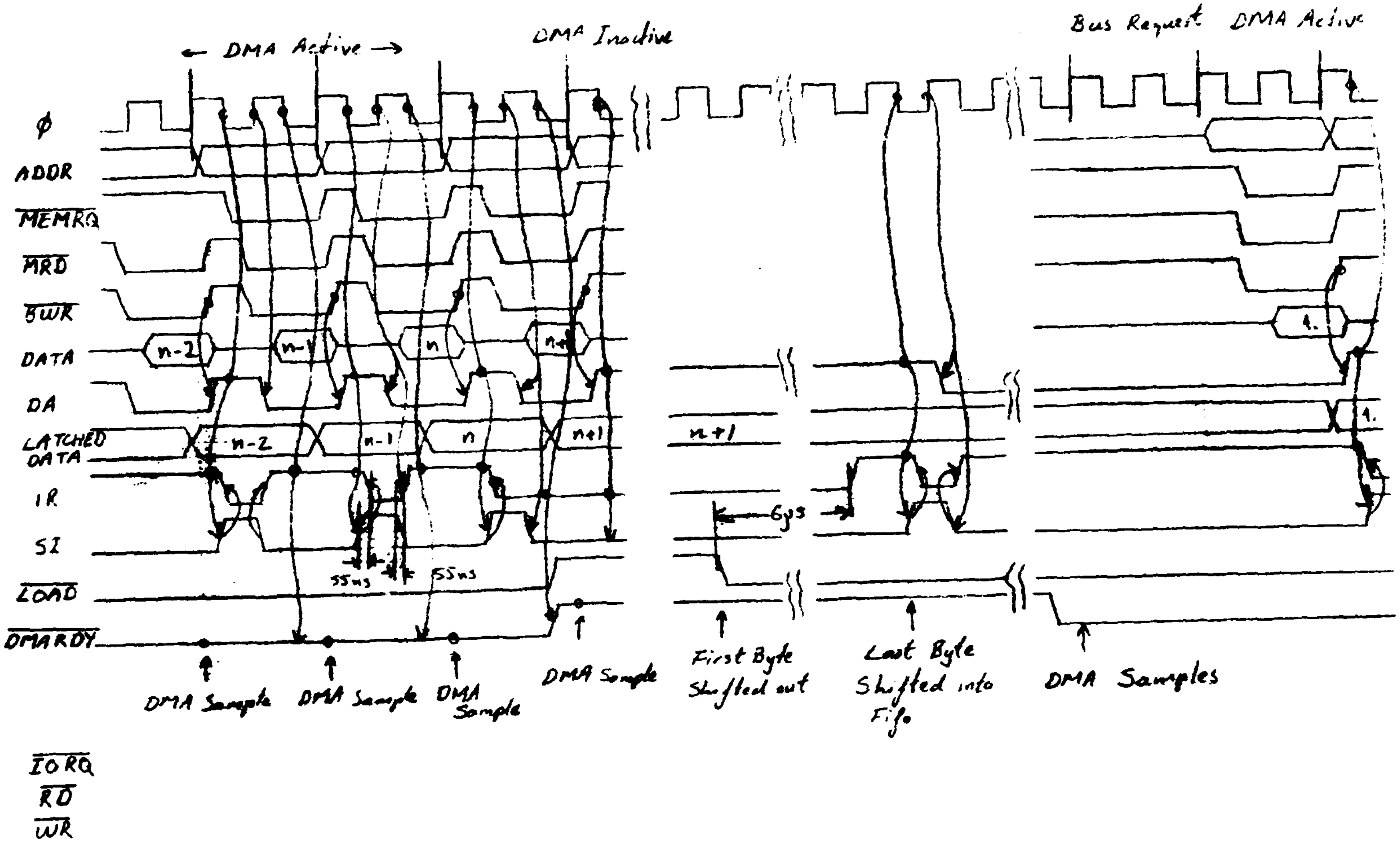


Figure 3.2 Timing Diagram for 2-cycle Simultaneous DMA Transfer to FIFO

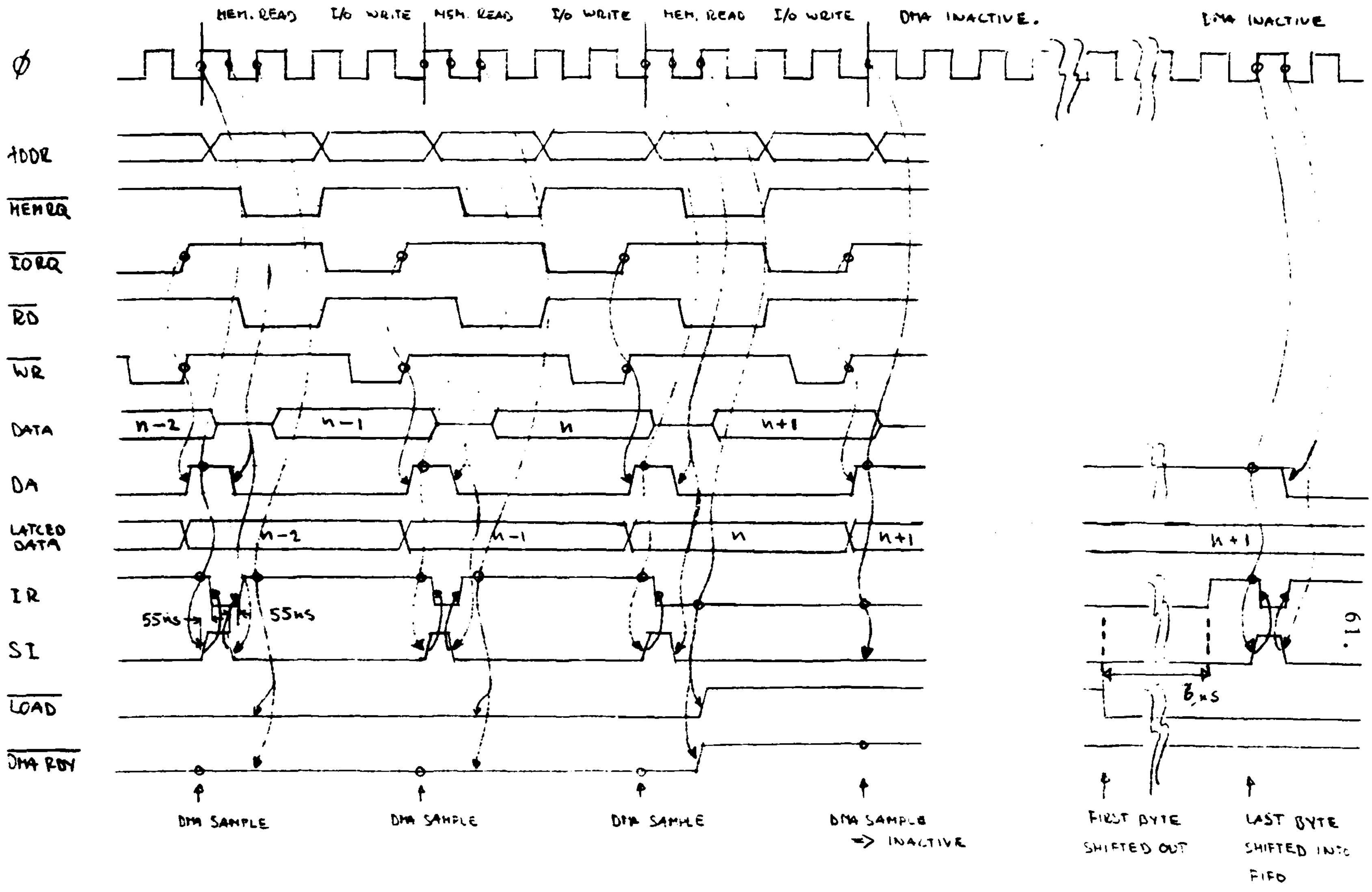


Figure 3.3a Timing Diagram for 2-cycle Sequential DMA Transfer to FIFO

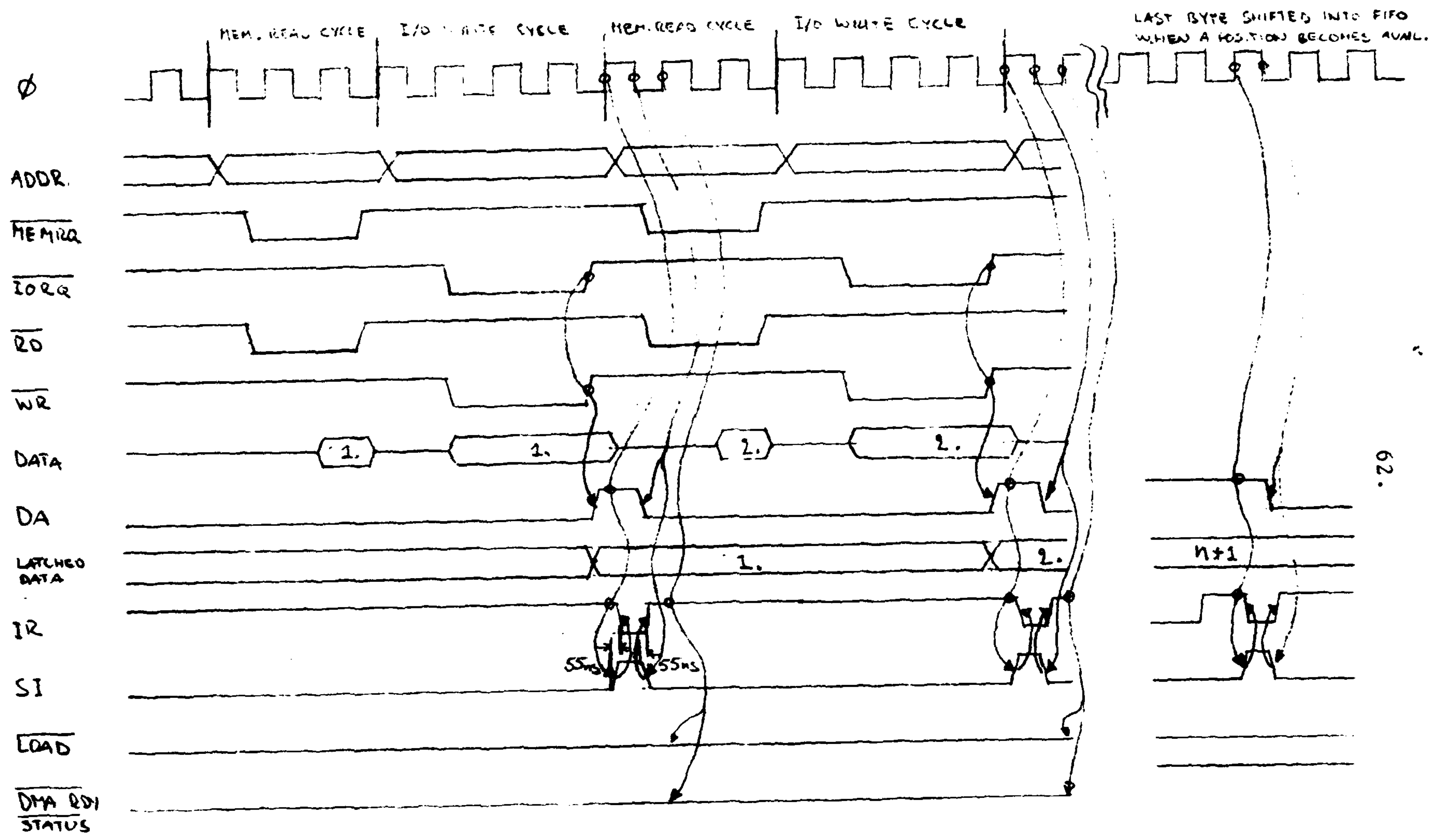


Figure 3.3b Timing Diagram for Part of I/O Write Instruction, "OTIR", to the FIFO

transfer logic, also changes the polarity of the clock input to the appropriate flip-flops.

The reason for the difference in timing indicated above is the following. In order to make the timing of the sequential DMA transfer appear as a CPU output cycle to the FIFO handshake logic, the variable timing option of the DMAC had to be used. This option was used to specify "early end" to the $\overline{\text{IORQ}}$ and $\overline{\text{WR}}$ signals. It was not possible, however, to use the same timing during the two-cycle simultaneous DMA transfer from memory operation, because the data from the memory would not be valid when $\overline{\text{BWR}}$ (board write) would go high again. Instead, the signals $\overline{\text{MEMRQ}}$ and $\overline{\text{RD}}$, from which $\overline{\text{BWR}}$ and $\overline{\text{MRD}}$ (memory read) are generated by the simultaneous transfer logic, have to be programmed as a "normal end".

Two more signals can be generated during the same transfer mode, $\overline{\text{BRD}}$ and $\overline{\text{MWR}}$ (board read and memory write). $\overline{\text{MWR}}$ is led through a D-type flip-flop in such a way that the leading edge of the artificially generated write signal is delayed one-half clock cycle in order to make it look like an ordinary CPU write signal.

An interesting aspect of the handshaking between the DMA process and the signal generator is that they are two asynchronous processes that are synchronized every time the FIFO reaches the "full state." At that particular time all parts of the logic are synchronized to the exact number of bytes in the FIFO because the logic is reset first at the very first byte shifted out after the FIFO becomes full.

Programming the Interface Board

To make the interface board operate in the intended way one has to transmit, via software, certain commands to the DMAC in the form of tables or single commands as well as giving the BCW the appropriate bit pattern. The control sequences for each transfer mode and operation can easily be stored in memory in the form of separate tables. These tables have fields to contain addresses, number of bytes to be transferred, etc., which can easily be changed by manipulating the corresponding memory location before writing the chosen table via an OTIR instruction to the DMAC.

An important point is that the DMAC has to be disabled before change of the BCW bits S_7 and S_6 can be done. If a change of transfer mode is made then the appropriate bits of the BCW have to be changed before the DMAC can be enabled again.

DMAC Program Examples

This section contains programming tables which are chosen to be a minimum set of command sequences for utilizing most of the features of the interface board. It is desirable and in some cases necessary, to create other command sequences. Information on how this is done can be found in Zilog's literature on the DMAC, specifically in the Z-80 DMA Technical Manual.

Below is a summary listing of the routines provided followed by detailed listings. Note that some of the tables can be strung together where appropriate.

1. Initialize DMAC for sequential transfer. Memory to FIFO. Not enabled, auto restart, no interrupt, burst mode, two-cycle, early end on signals where necessary.
2. Initialize DMAC for simultaneous transfer. Memory to FIFO. Not enabled, auto restart, no interrupt, burst mode, two-cycle, early end on signals where necessary.
3. Enable DMAC.
4. Disable DMAC.
5. Load DMAC and Enable.
6. Change from sequential to simultaneous transfer.
7. Change from simultaneous to sequential transfer.
8. Change port address.
9. Change of table address for sequential transfer.
10. Change of table address for simultaneous transfer.
11. Change of table length for sequential transfer.
12. Change of table length for simultaneous transfer.

13. Change table address and length and port for sequential transfer.

14. Change table address and length and port for simultaneous transfer.

15. Change of cycle length, two-cycle to three-cycle.

16. Change of cycle length, three-cycle to two-cycle.

1. INITIALIZE DMA - SEQ. MEM → FIFO

79	(B. temp. source) needed because fixed address.	0 1 1 1 1 0 0 1	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
XX		Block length low byte	
XX		Block length high byte	
54	(A is MEM & increments)	0 1 0 1 0 1 0 0	WR1
CA	(2 cycle, $\overline{\text{IORQ}}$ early end)	1 1 0 0 1 0 1 0	
78	(B is I/O, fixed)	0 1 1 1 1 0 0 0	WR2
4A	(2 cycle, $\overline{\text{IORQ}}$ & $\overline{\text{WR}}$ early end)	0 1 0 0 1 0 1 0	
C5	(Burst, no Interrupt)	1 1 0 0 0 1 0 1	WR4
EB		FIFO I/O Addr. (low byte only)	
CF	(Load B, reset block count)	1 1 0 0 1 1 1 1	WR6
A2	(Auto Restart, Rdy. active low)	1 0 1 0 0 0 1 0	WR5
05	(A source)	0 0 0 0 0 1 0 1	WRØ
CF	(Load A, reset block count)	1 1 0 0 1 1 1 1	WR6

NOTE: control word needs to be programmed (I/O addr. E7).

$$(0\ 0)_H \qquad 0\ 0\ X\ \underset{\underbrace{0\ 0\ 0}_{\text{enable 11-port}}}{X\ X\ X}\ X\ 0$$

Above table needs to be read out to DMA (I/O addr. E3).

2. INITIALIZE DMA - SIM. MEM → FIFO

7E	(A source)	0 1 1 1 1 1 1 0	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
XX		Block length low byte	
XX		Block length high byte	
54	(A is MEM & increments)	0 1 0 1 0 1 0 0	WR1
CA	(2 cycle, $\overline{\text{IORQ}}$ early end)	1 1 0 0 1 0 1 0	
C1		1 1 0 0 0 0 0 1	WR4
A2	(Auto Restart, Rdy. active low)	1 0 1 0 0 0 1 0	WR5
CF	(Load A, reset block count)	1 1 0 0 1 1 1 1	WR6

NOTE: control word needs to be programmed (I/O addr. E7).

$(80)_H$ 1 0 X X X X X 0

3. ENABLE DMA

87	1 0 0 0 0 1 1 1	WR6

4. DISABLE DMA

83	1 0 0 0 0 0 1 1	WR6

NOTE: The control word must be appropriately set before enabling the DMA. Also, disable DMA before changing the two first bits of the control word.

5. LOAD DMA AND ENABLE

CF		1 1 0 0 1 1 1 1	WR6
87		1 0 0 0 0 1 1 1	WR6

6. CHANGE SEQ. → SIM. WITHOUT LOAD

06	(A source)	0 0 0 0 0 1 1 0	WRØ
54	(A is MEM & increments)	0 1 0 1 0 1 0 0	WR1
CA		1 1 0 0 1 0 1 0	

7. CHANGE SIM. → SEQ. WITHOUT LOAD

01	(B temp. source)	0 0 0 0 0 0 0 1	WRØ
54		0 1 0 1 0 1 0 0	WR1
CA	(2 cycle, $\overline{\text{IORQ}}$ early end)	1 1 0 0 1 0 1 0	
78		0 1 1 1 1 0 0 0	WR2
4A		0 1 0 0 1 0 1 0	

NOTE: 5 Load DMA and Enable will start the tables from the beginning. 3 Enable DMA can be used if one wants to continue from where we currently are in the table.

NOTE: Before Enable DMA the correct control word must be given.

8. CHANGE OF PORT-ADDR. (SEQ)

01	(B temp. source)	0 0 0 0 0 0 0 1	WRØ
C5	(Burst, no interrupt)	1 1 0 0 0 1 0 1	WR4
XX		I/O Addr. low byte only	
CF	(Load B, reset block count)	1 1 0 0 1 1 1 1	WR6
05	(A source)	0 0 0 0 0 1 0 1	WRØ

NOTE: No change in memory table above. If restart required then utilize 5 LOAD DMA & ENABLE. If not, 3 ENABLE DMA.

9. CHANGE OF TABLE ADDR. (SEQ)

1D	(A source)	0 0 0 1 1 1 0 1	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
CF	(Load A, reset block count)	1 1 0 0 1 1 1 1	WR6

10. CHANGE OF TABLE ADDR. (SIM)

1E	(A source)	0 0 0 1 1 1 1 0	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
CF	(Load A, reset block count)	1 1 0 0 1 1 1 1	WR6

11. CHANGE OF TABLE LENGTH (SEQ)

65	0 1 1 0 0 1 0 1	WRØ
XX	Block length low byte	
XX	Block length high byte	

12. CHANGE OF TABLE LENGTH (SIM)

66	0 1 1 0 0 1 1 0	WRØ
XX	Block length low byte	
XX	Block length high byte	

NOTE: No load is included - the new block length will be loaded when "Auto restart." Include in such case ENABLE DMA 3. If new block length is wanted immediately then we LOAD DMA & ENABLE DMA 5.

13. CHANGE TABLE ADDR. & LENGTH AND PORT. (SEQ)

79	(B temp. source)	0 1 1 1 1 0 0 1	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
XX		Block length low byte	
XX		Block length high byte	
C5	(Burst, no interrupt)	1 1 0 0 0 1 0 1	WR4
XX		I/O Addr (low byte only)	
CF	(Load B, reset block count)	1 1 0 0 1 1 1 1	WR6
05	(A source)	0 0 0 0 1 0 1 0	WRØ

14. CHANGE TABLE ADDR. & LENGTH (SIM)

7E		0 1 1 1 1 1 1 0	WRØ
XX		MEM Addr. low byte	
XX		MEM Addr. high byte	
XX		Block length low byte	
XX		Block length high byte	

NOTE: In 13 and 14 the new block addr. and length will be loaded at Auto restart. Use ENABLE DMA 3.

If immediately change is needed use 5 LOAD AND ENABLE DMA.

15. CHANGE CYCLE LENGTH 2 → 3 (SEQ)

54	(A is MEM, increments)	0 1 0 1 0 1 0 0	WR1
C9	(3 cycle, $\overline{\text{IORQ}}$ end early)	1 1 0 0 1 0 0 1	
78	(B is I/O, fixed)	0 1 1 1 1 0 0 0	WR2
49	(3 cycle, $\overline{\text{IORQ}}$ & $\overline{\text{WR}}$ end early)	0 1 0 0 1 0 0 1	

16. CHANGE CYCLE LENGTH 2 → 3 (SIM)

54	(A is MEM, increments)	0 1 0 1 0 1 0 0	WR1
C9		1 1 0 0 1 0 0 1	

Pin Definitions

The following is a list of the different signals used to feed and interface the signal generator and the CCD signal processing modules.

J1 - J11			
16	1	CLR	
15	2	D_0	
17	4	D_1	
78	6	D_2	SIGN.
19	8	D_3	GEN.
80	10	D_4	
21	12	D_5	
83	14	D_6	
87	16	D_7	
20	18	$\overline{\text{OR}}$	
18	20	$\overline{\text{SO}}$	
120	9	GND	
108	13	C_7	
47	25	C_6	
106	11	C_5	
45	23	C_4	
104	21	C_3	
43	19	C_2	
41	17	C_1	
39	15	C_0	
44	7	$\overline{\text{SIGN. GEN. MODE}}$	

J2 - J12			
46	1	$\overline{\text{PTF MODE}}$	
48	2	D_0	
49	3	I/O CONTROL	
50	4	D_1	
53	5	$\overline{\text{DMA RDY}}$	Parallel Port
52	6	D_3	
55	7	STATUS	
54	8	D_4	
56	10	D_6	
58	12	D_7	
112	14	D_2	
57	24	$\overline{\text{WR STROBE}}$	
117	22	D_5	
121	9	GND	
108	13	C_7	
47	25	C_6	
106	11	C_5	
45	23	C_4	
104	21	C_3	
43	19	C_2	
41	17	C_1	
39	15	C_0	

D = data bits

C = control bits

The backplane bus has been modified by interconnecting the pins indicated below.

J1 - J2		
10	10	MASTER RESET
81	81	DAISY CHAIN
88	88	$\overline{\text{BUSAK}}$
109	109	$\overline{\text{BUSRQ}}$

Circuit Diagram for the Interface Board

The circuit diagram for the interface board is contained on the following pages. For convenience, it is subdivided into six separate drawings. The division has been made in a way which clusters related functions on the same sheet. Figure 3.4 illustrates the block diagram of the interface board.

Figure 3.5 illustrates the DMA controller and the address buffers necessary to enable the DMAC to drive the system bus. A few control signals are also in this drawing.

Figure 3.6 shows the buffering of the control signals, the bus interface logic and the simultaneous transfer logic. Note the D-type flip-flop which ensures that the artificially generated write signal going out to the system bus will look like an ordinary CPU generated one.

Figure 3.7 contains the board control word, BCW, and its buffers to enable both write and read. The CPU is, however, the only device allowed to perform either of these operations.

Figure 3.8 shows the high-speed FIFO port together with all its related logic for synchronization, timing and handshaking. Note the two XOR gates that control which clock edge will cause an action, via bit S_7 of the BCW.

Figure 3.9 illustrates the circuit for the high-speed parallel port, HSPP, its control logic, handshaking signals and the decoding logic for all on-board addressable devices and functions.

Figure 3.10 shows the general purpose I/O port that is directly connected to the system bus. It is in its inactive state

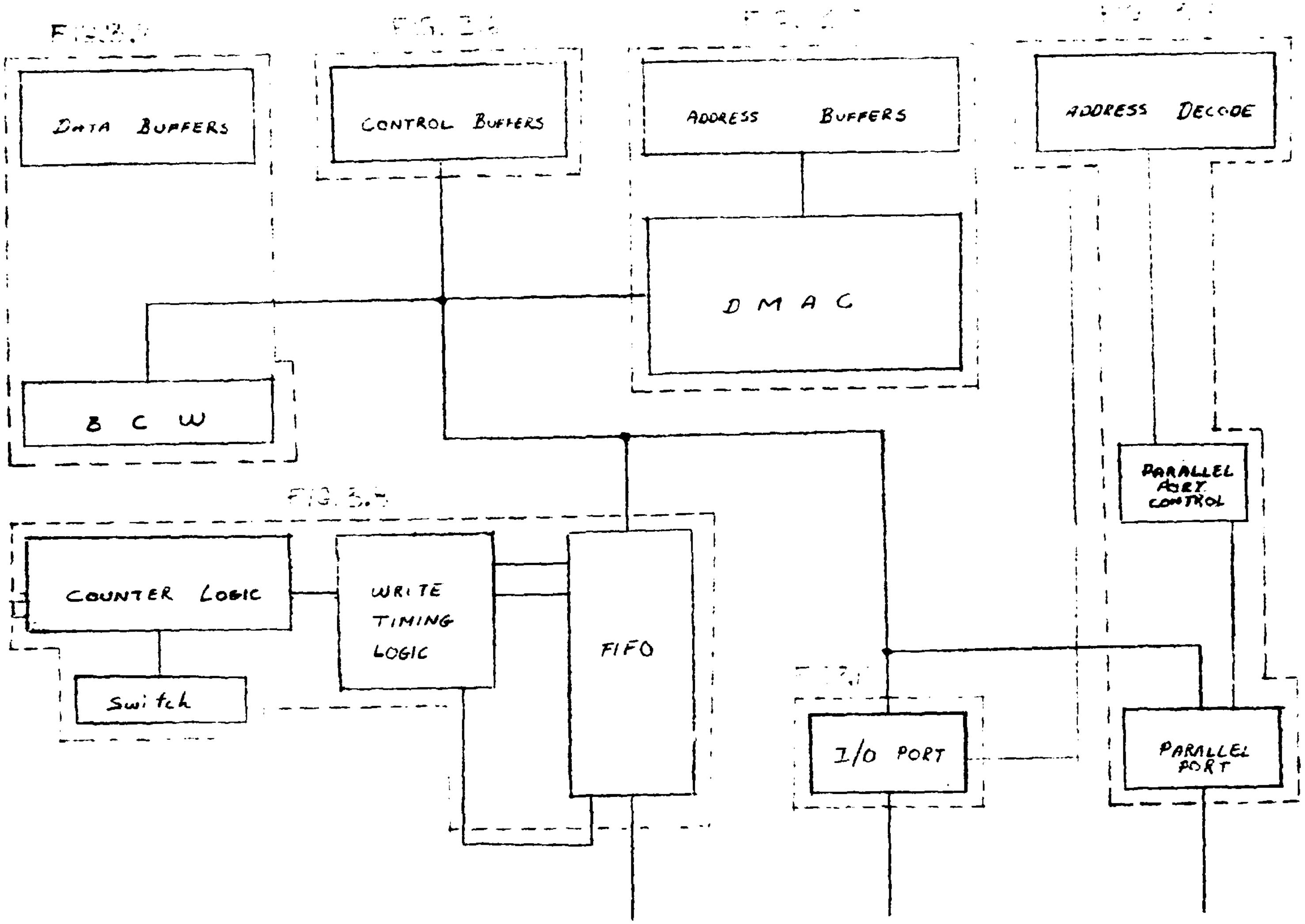
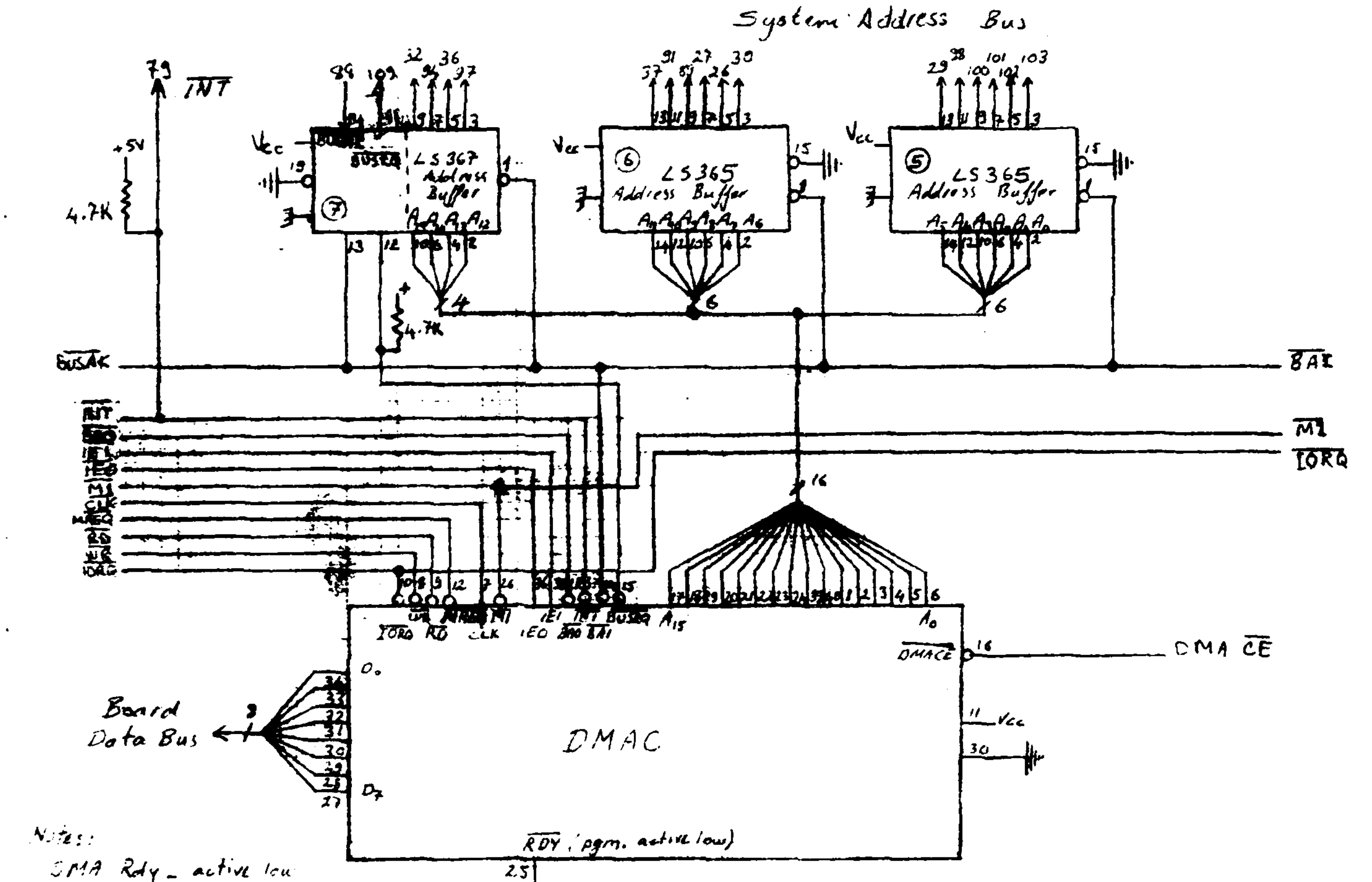


Figure 3.4 Block Diagram of Interface Board



Notes:

DMA Rdy – active low

Board is designed to handle 2-cycle DMA transfers

Sim. Transfer: Pgm. – MEM port $\overline{IORQ}$ and early. $\overline{MREQ}$ & $\overline{RD}$ [illegible]

Seq. Transfer: Pgm. I/O port. $\overline{IORQ}$ & $\overline{WR}$ and early. MEM. port $\overline{IORQ}$ and early. $\overline{MEMRQ}$ & $\overline{RE}$ [illegible] end.

Figure 3.5 DMA Controller and Address Buffers

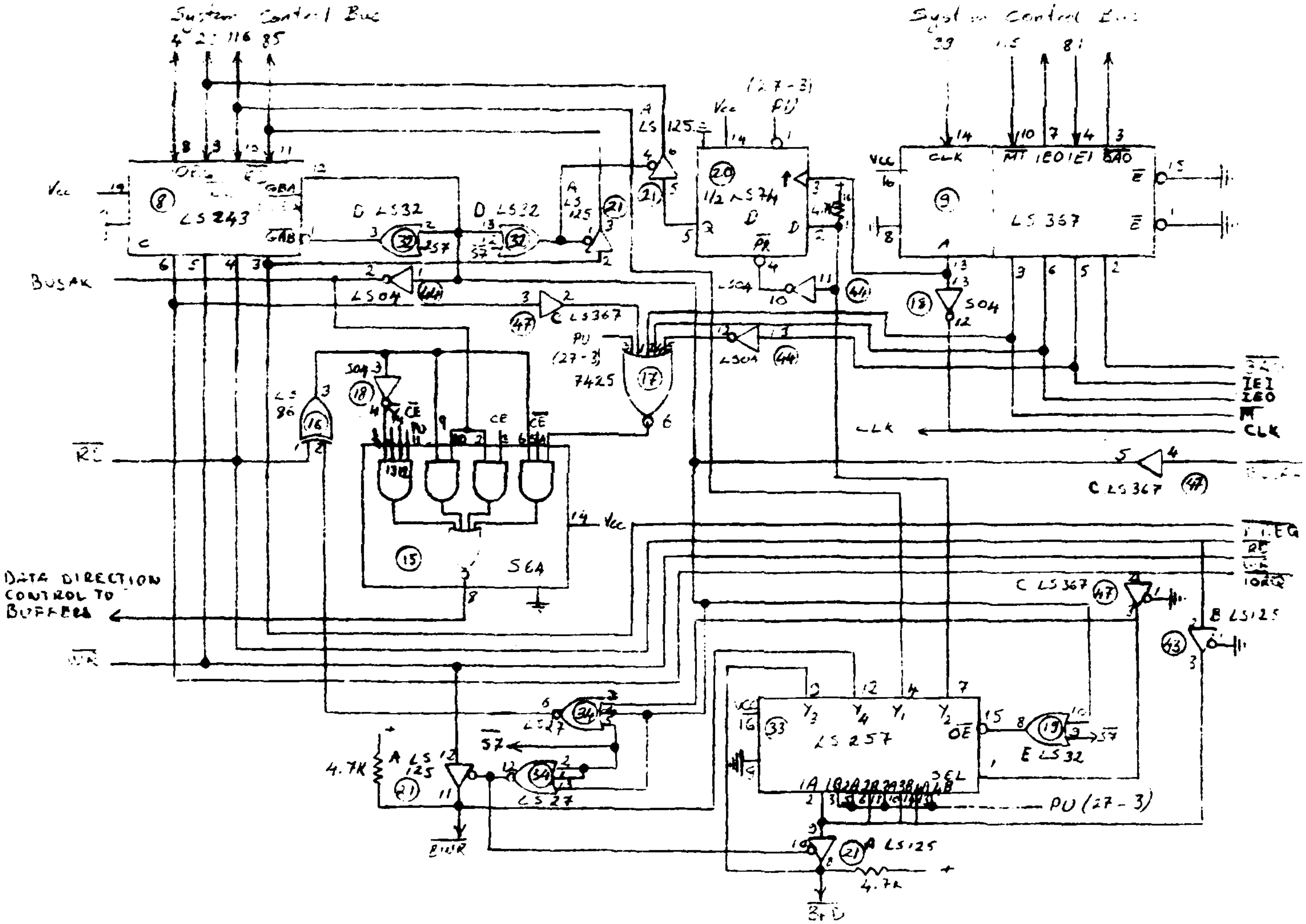
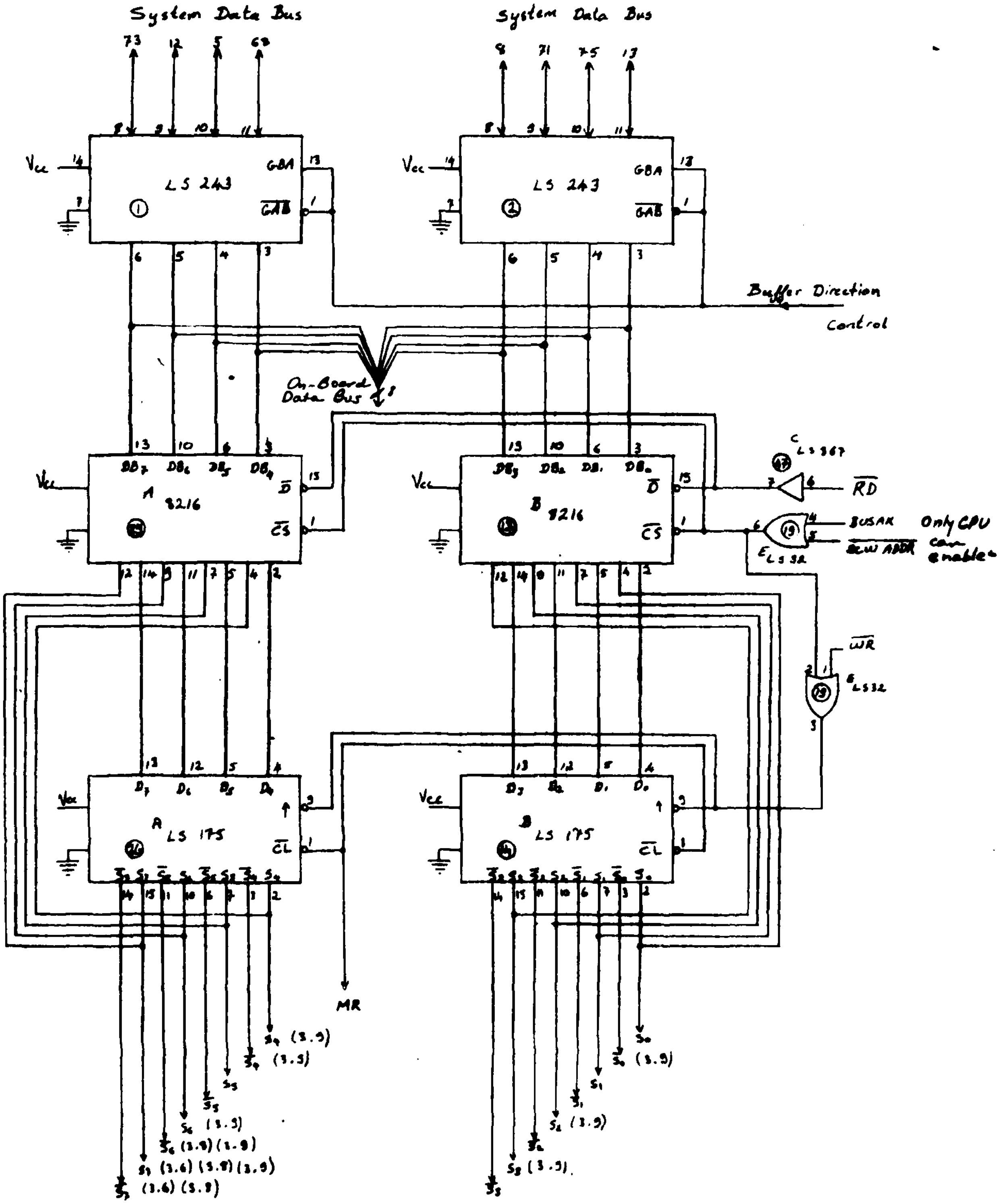


Figure 3.6 Buffers for Control Signals, Bus Interface Logic and Simultaneous Transfer Logic



( ) = figure number

Figure 3.7 Board Control Word and Buffers

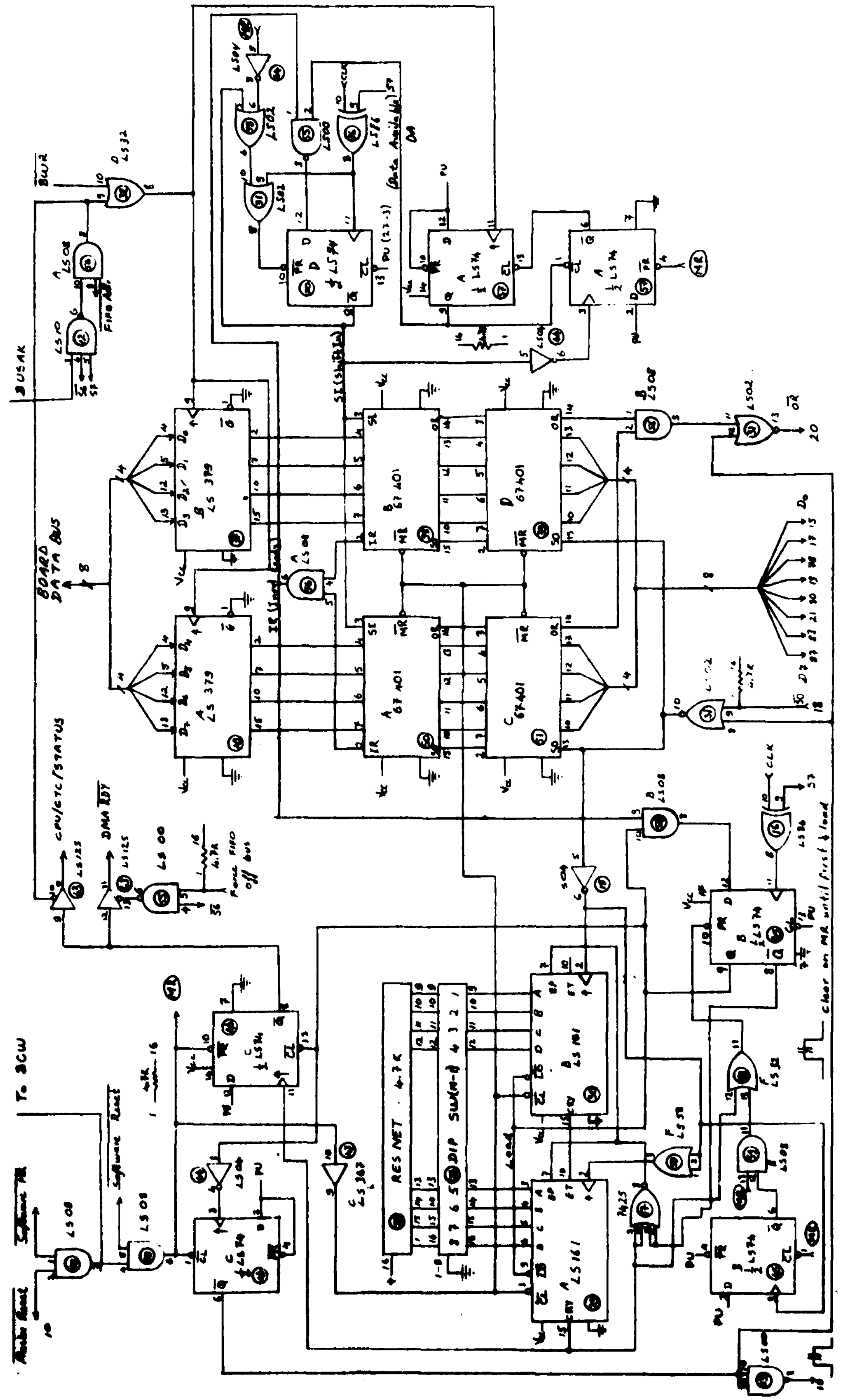


Figure 3.8 High-Speed FIFO Port and Associated Logic

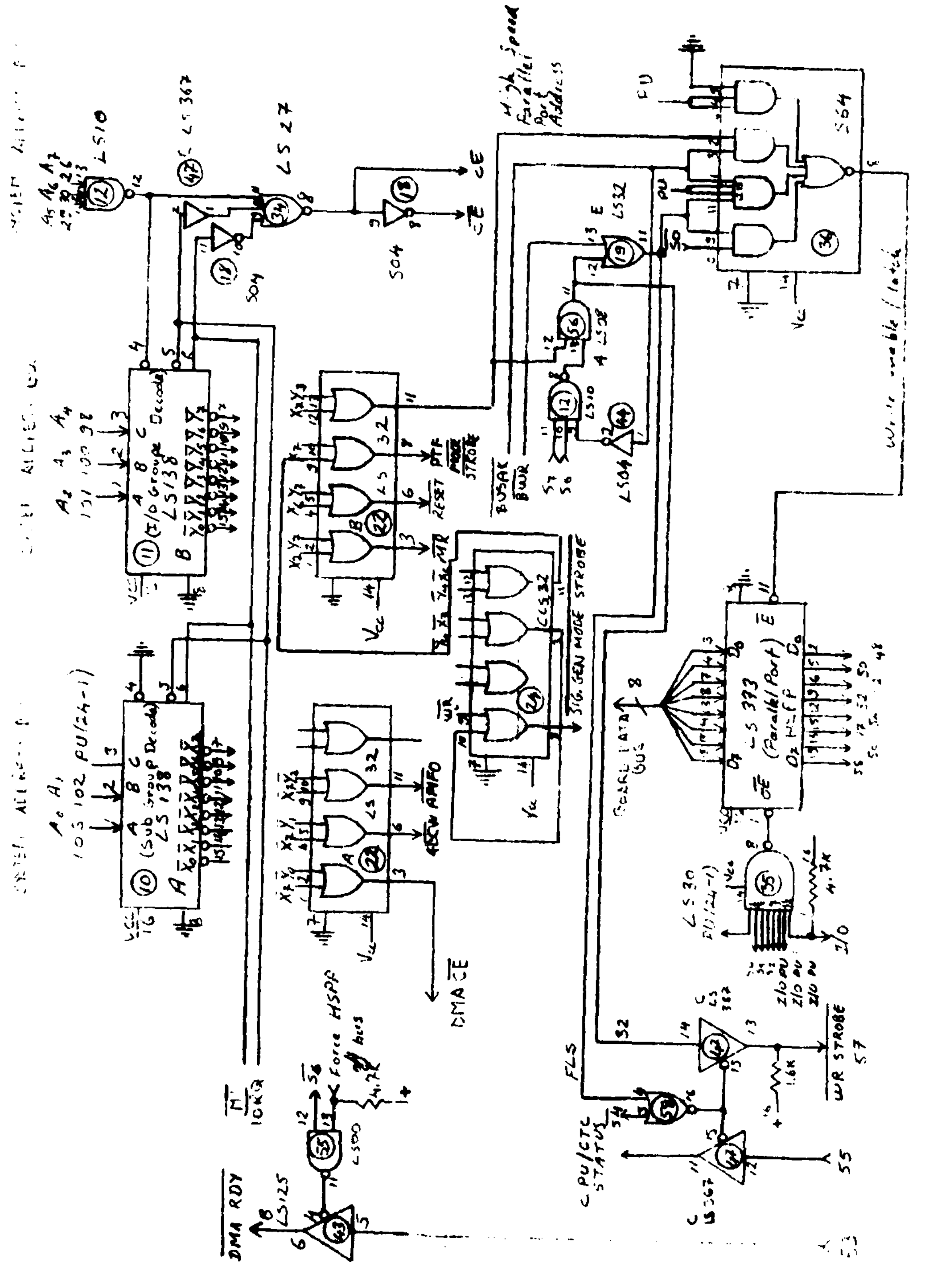


Figure 3.9 High-Speed Parallel Port and Associated Logic

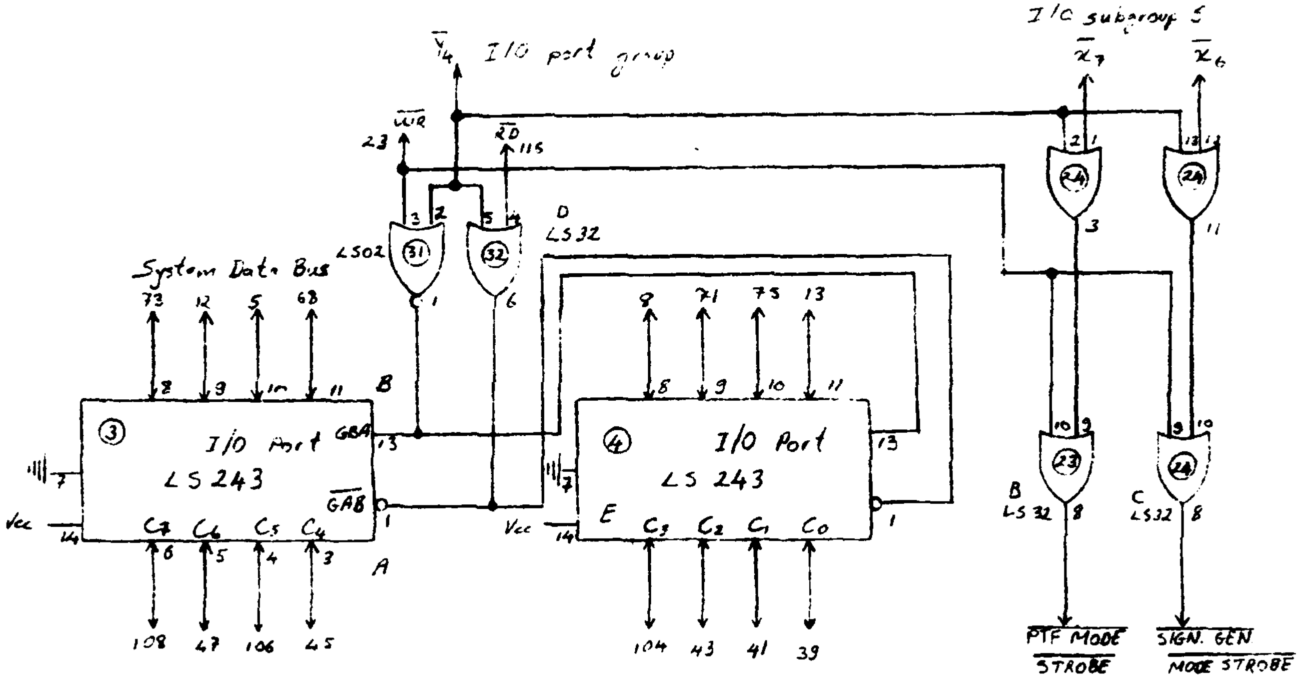


Figure 3.10 General Purpose I/O Port

4. Analog Binary Correlator (ABC)

Introduction

The Analog Binary Correlator (ABC) is a programmable 512-stage CCD tapped delay line capable of correlating a variety of binary-coded waveforms. The basic architecture of the ABC is shown in Figure 4.1 (see References 3 and 4).

The program register is a binary serial-in/parallel-out shift register comprised of two 256-bit sections through which the binary reference code is loaded. The binary latches constitute a static memory for storing the reference code. The isolation switches control the parallel shift of data from the program register to the binary latches. They allow the reference code to be updated while the correlator is still acting on the reference code previously stored. The routing switches steer the output of a given tap to the appropriate summing bus and are controlled by the reference code stored in the binary latches. The Tap FET's in conjunction with the routing switches convert the charge fluctuations of the CCD Floating gate to tap currents which are then summed on the summing buses. The CCD tapped delay line consists of two 256-stage CCD registers coupled by a corner turning circuit. Figure 4.2 is a one-stage slice showing all of the devices from the program register to the CCD register.

The clock voltages shown in Figure 4.2 that are generated off chip are ϕ_A, ϕ_B, ϕ_{ISO}, ϕ_{RESET} and ϕ_T. ϕ_A and ϕ_B are two phase clock signals for shifting the reference code into the program register. For complete loading, 512 cycles of ϕ_A and ϕ_B

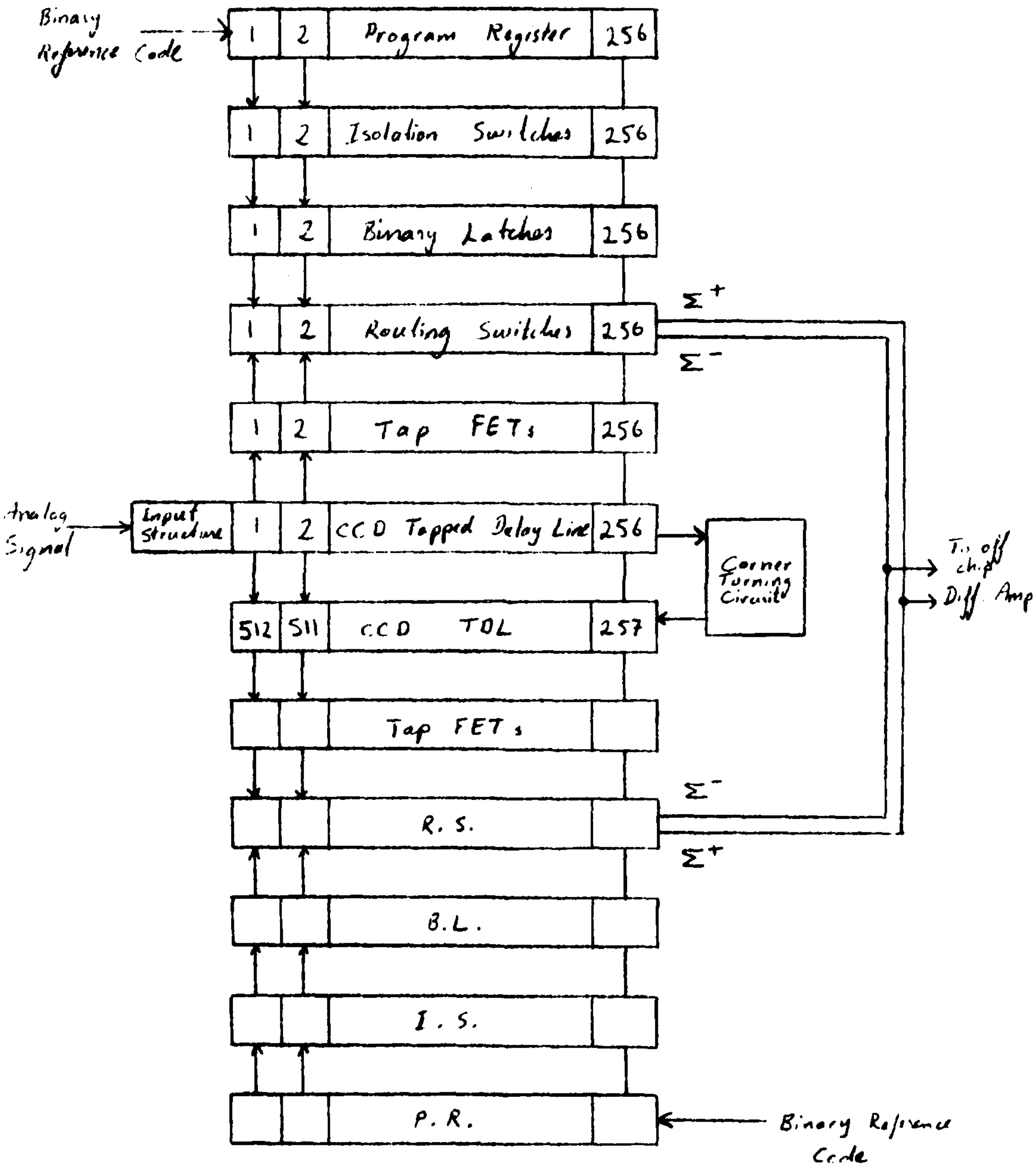


Figure 4.1 Basic Architecture of the ABC

are required. ϕ_{ISO} effects parallel transfer of the reference code from the program register to the binary latches. ϕ_{RESET} enables the binary latch to receive the data from the program register. ϕ_{FG} is generated on chip and resets the floating gate to a DC reference for each sample interval.

Not shown in Figure 4.2 is the master clock, MCK and ϕ_{SI}. ϕ_{SI}, synchronized with ϕ_T and MCK, is a clock signal used to sample the analog input signal and convert it to charge packets at the input to the CCD register.

Figure 4.3 is a simplified block diagram and timing for the ABC when operating under control of the DMA. The Master Clock, MCK, is crystal controlled and buffered and is used to synchronize the DMA control functions. ϕ_{SI} and ϕ_T, at TTL levels, runs at the MCK rate but a delay circuit controls the relative timing between the two clocks. A dual clock driver, 2N74265, generates $\bar{\phi}_A$ and $\bar{\phi}_B$ when the output from the control latch to the 2N74265 is high. This input is timed for 512 cycles of both $\bar{\phi}_A$ and $\bar{\phi}_B$. A dual MOS clock driver converts the $\bar{\phi}_A$ and $\bar{\phi}_B$ signals to MOS levels. When the program register is completely loaded the disable signal turns off ϕ_A and ϕ_B leaving ϕ_A high and ϕ_B low. The disable pulse is used as the input to the shift register, which is clocked by MCK, to obtain first the $\phi_{\overline{RESET}}$ and then the ϕ_{ISO} pulses. Valid correlation occurs following the ϕ_{ISO} pulse.

Circuit Description

The ABC is partitioned using three single Tektronix bins in a TM 504 power module. The power supply module makes use of the

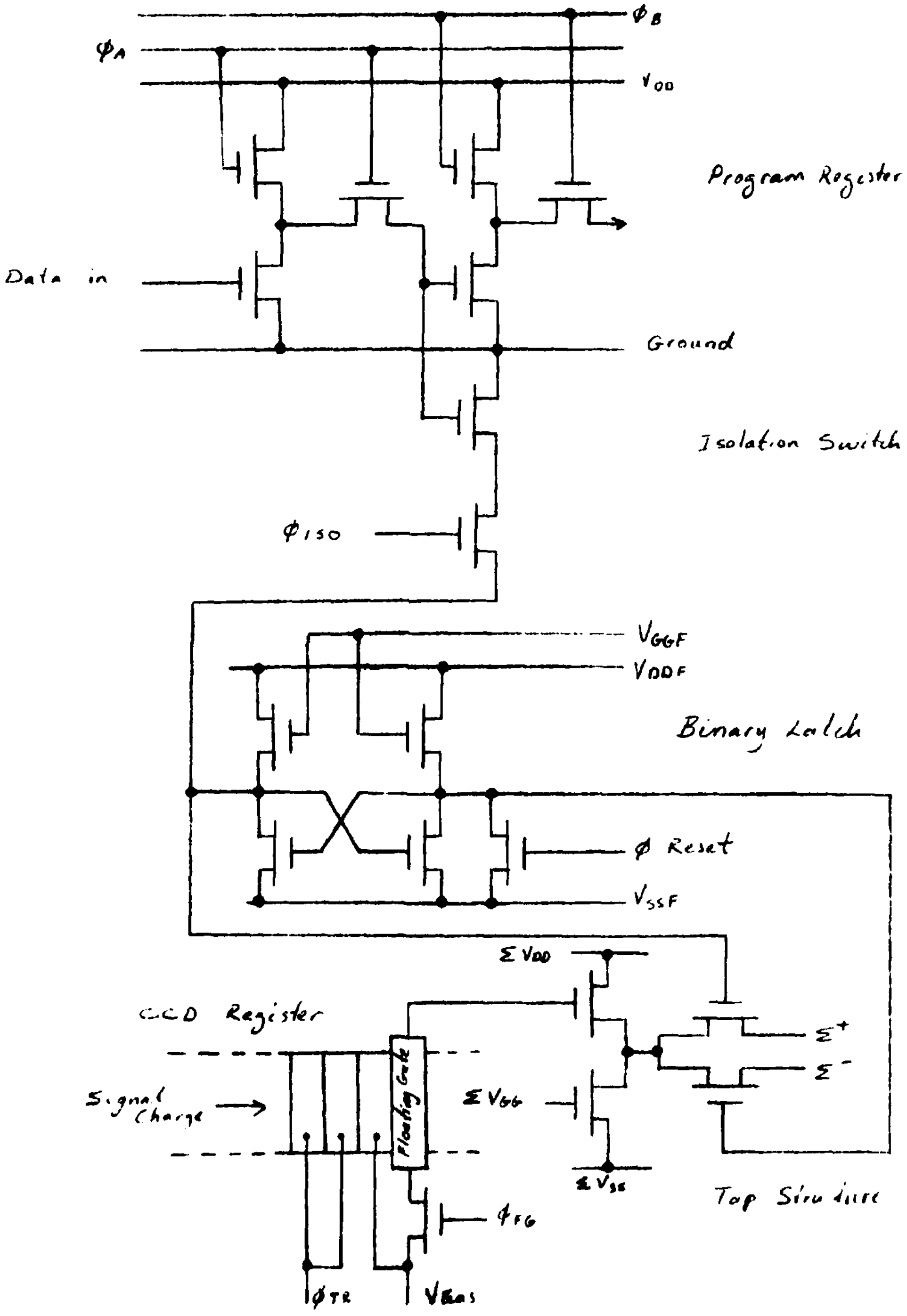


Figure 4.2 One-Stage Slice Showing all Devices from Program Register to CCD Register

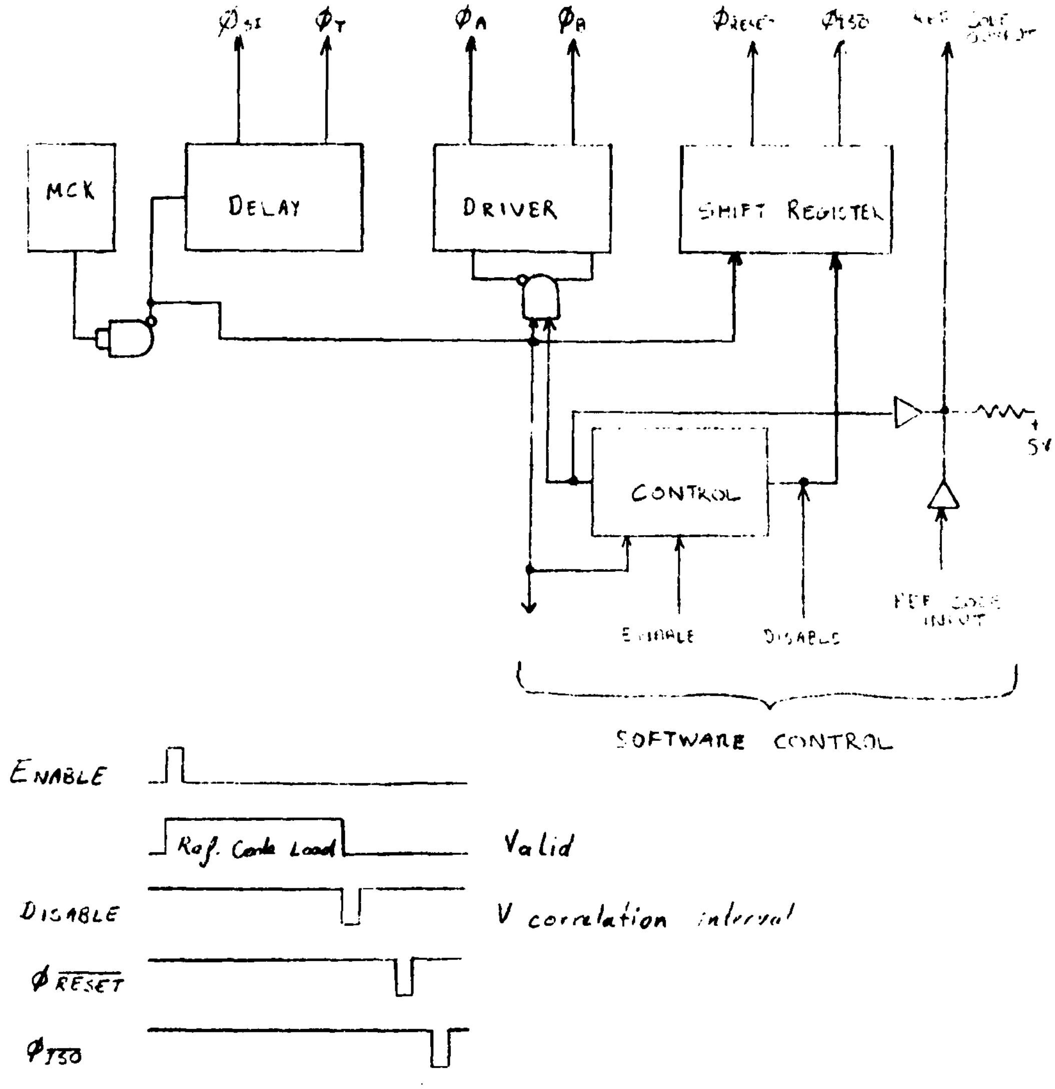


Figure 4.3 Simplified Block Diagram and Timing for the ABC

internal power available from the TM 504 to obtain ±15 volts, 5 volts and +30 volts. The circuit diagram is shown in Figure 4.4. Figure 4.5 contains the circuit details of the digital board, i.e., MCK, ϕ_T, ϕ_{SI}, ϕ_A, ϕ_B, ϕ_{RESET} and ϕ_{ISO}.

Figures 4.6(a), (b) and (c) show the circuit details of the analog board and bias voltages required by the chip. The driver circuit for ϕ_T and ϕ_{SI} is shown in Figure 4.6(a). The input PN code driver circuitry is indicated in Figure 4.6(b). This circuit also indicates the automatic fat zero compensating circuitry, using LM 108-1 in a closed-loop configuration which negates any necessity for readjustment when chips are interchanged.

Figure 4.6(c) contains additional bias supplies required by the chip. In addition there is another closed-loop compensating circuit using LM 108-2 for controlling V_{DD} and V_{GG} of the tap FET's.

Figure 4.7 is a simplified representation of the ABC chip showing the input-output connections. The correlated output is shown as transformer coupled. However, at the frequencies we will be operating the ABC we anticipate using a differential video amplifier in an attempt to eliminate the shortcomings of the transformer.

ABC Interface with the Microprocessor

The coefficients (reference signal) used in the ABC are generated by the microprocessor. Figure 4.8 shows a block diagram of the microprocessor-ABC interface. In typical

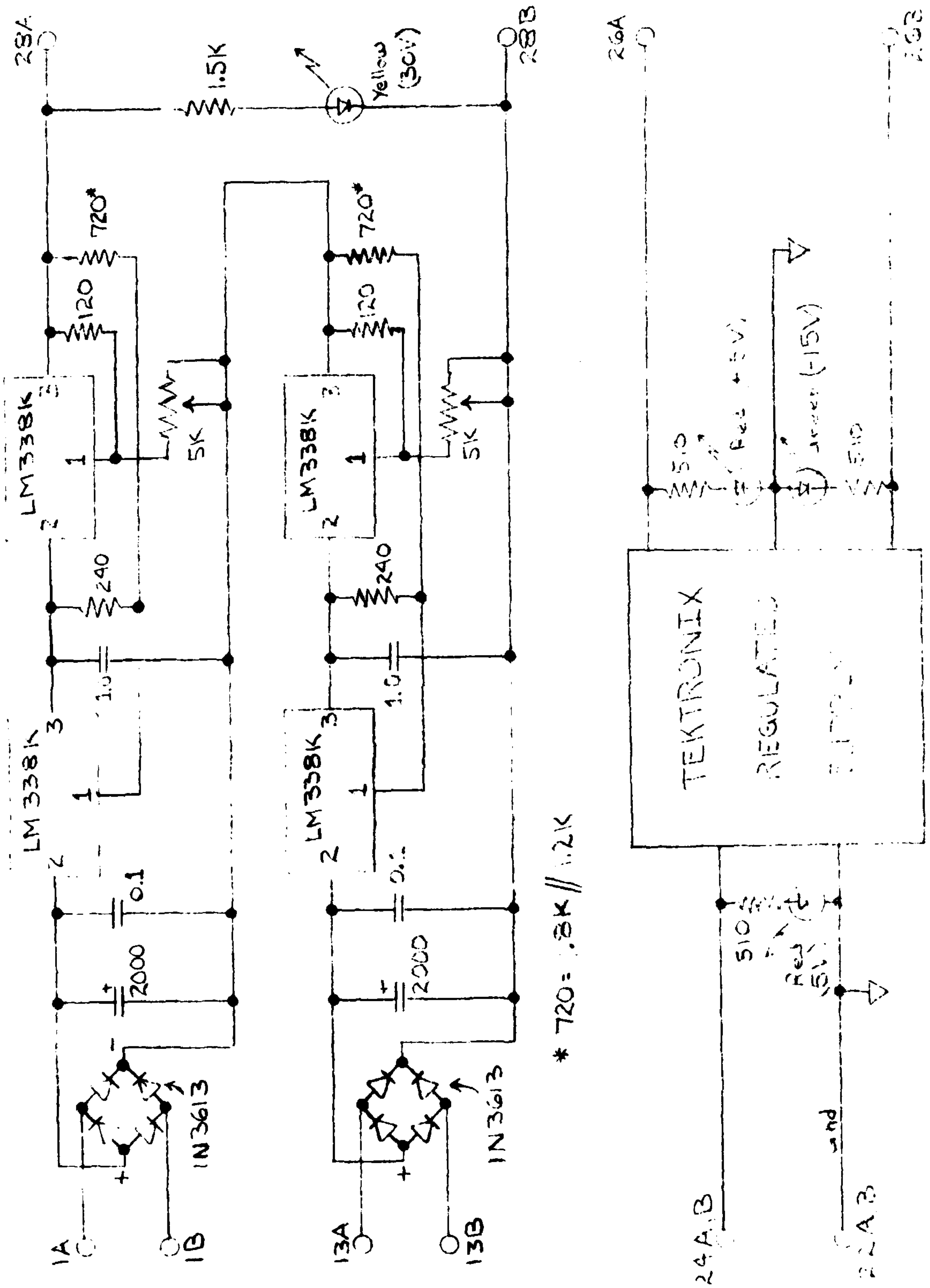


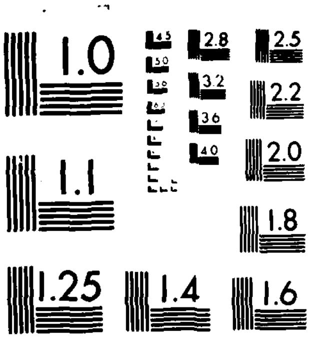
Figure 4.4 ABC Power Supply Module

AD-A128 676 INVESTIGATION OF CHARACTERISTICS AND APPLICATIONS OF NEW CCD SIGNAL PROCE..(U) NORTHEASTERN UNIV BOSTON MASS DEPT OF ELECTRICAL ENGINEERING.. B L COCHRUN ET AL. FEB 83 RADC-TR-82-322 F19628-79-C-0165 F/G 9/3

UNCLASSIFIED NL

2/2

END
DATE
FILMED
7-83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

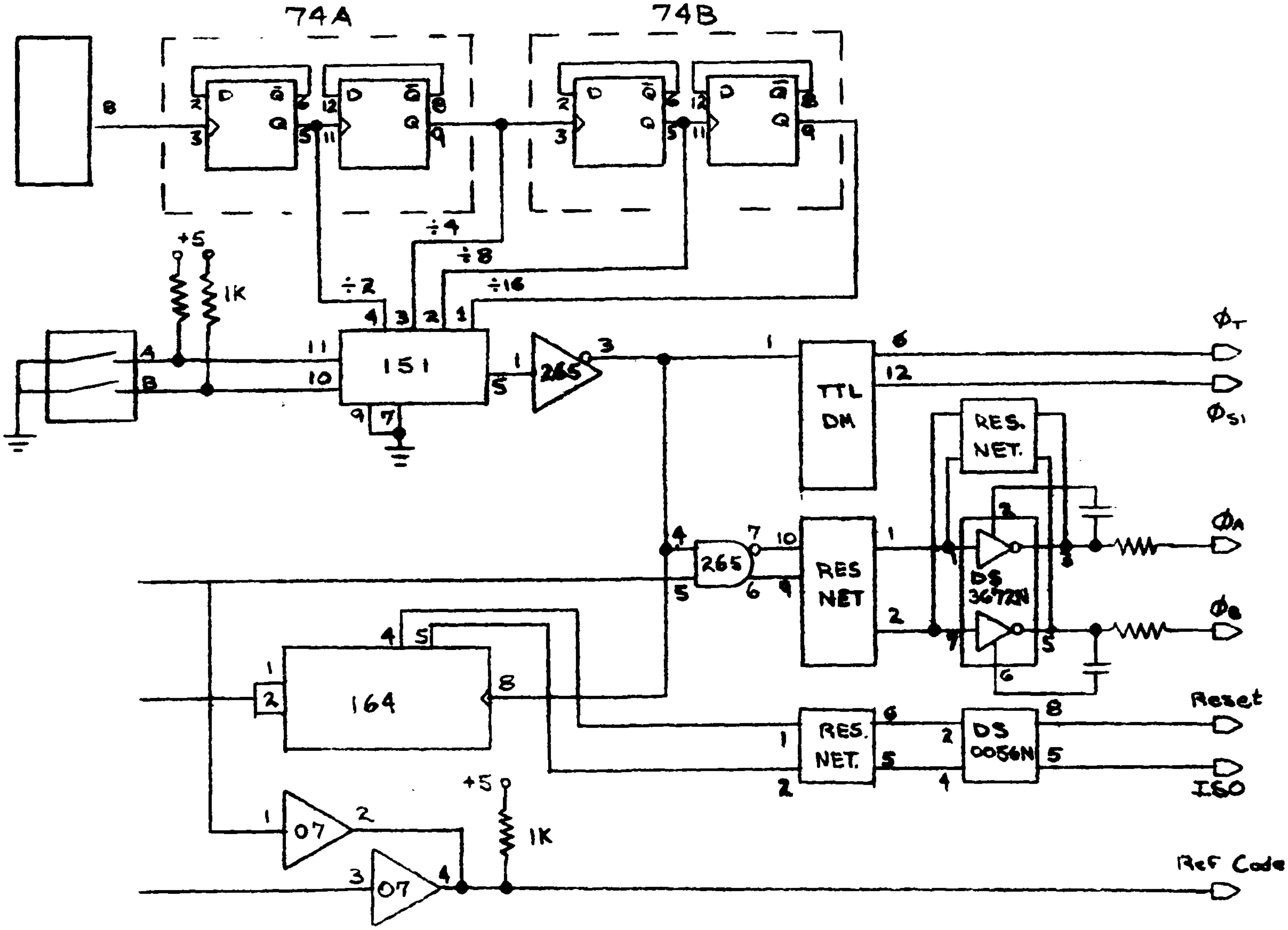


Figure 4.5 Circuit Details of the Digital Board for MCK, ϕ_T, ϕ_{SI}, ϕ_A, ϕ_B, ϕ_{RESET} and ϕ_{ISO}

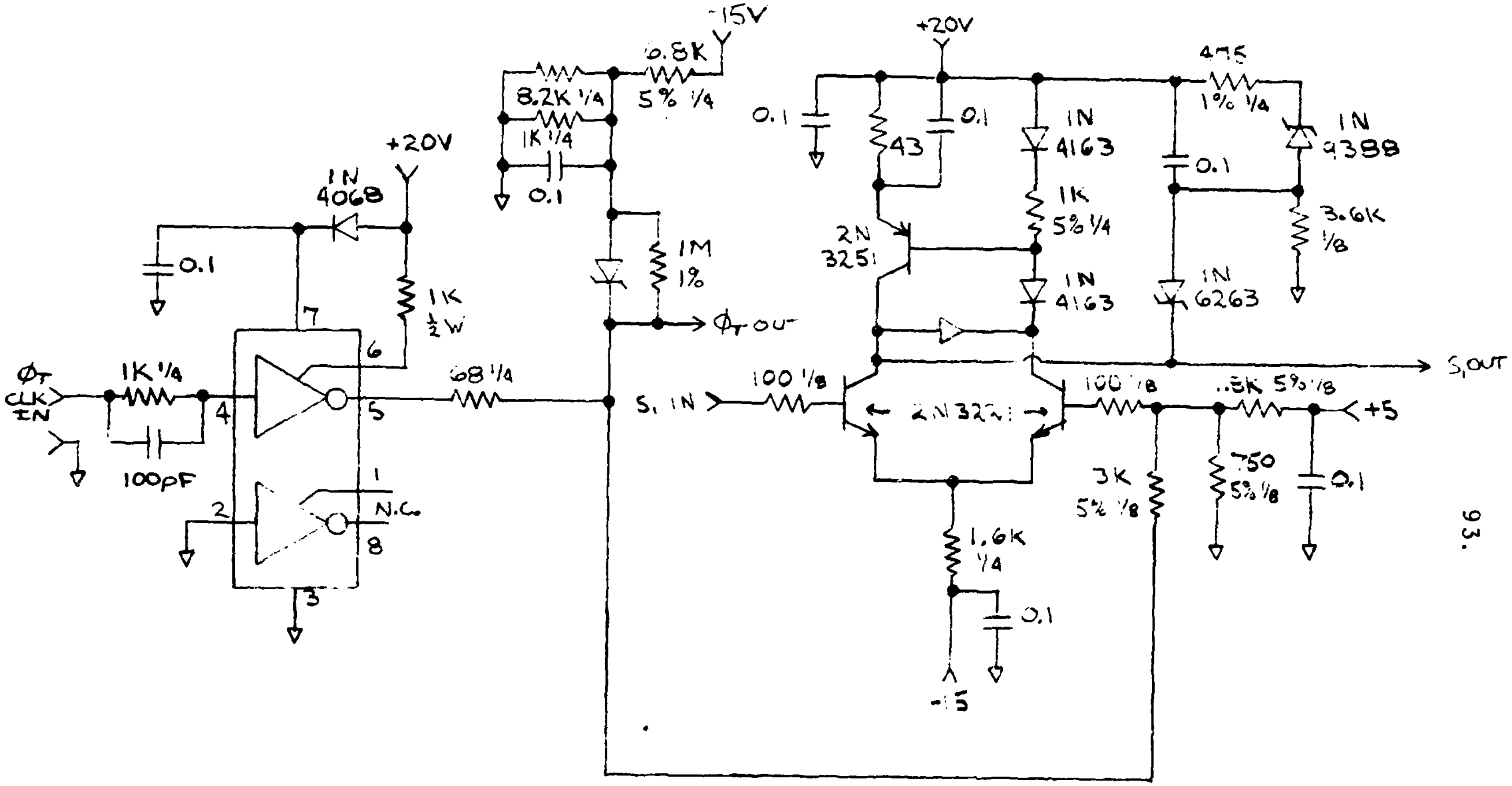


Figure 4.6a Driver Circuit for ϕ_T and ϕ_{SI}

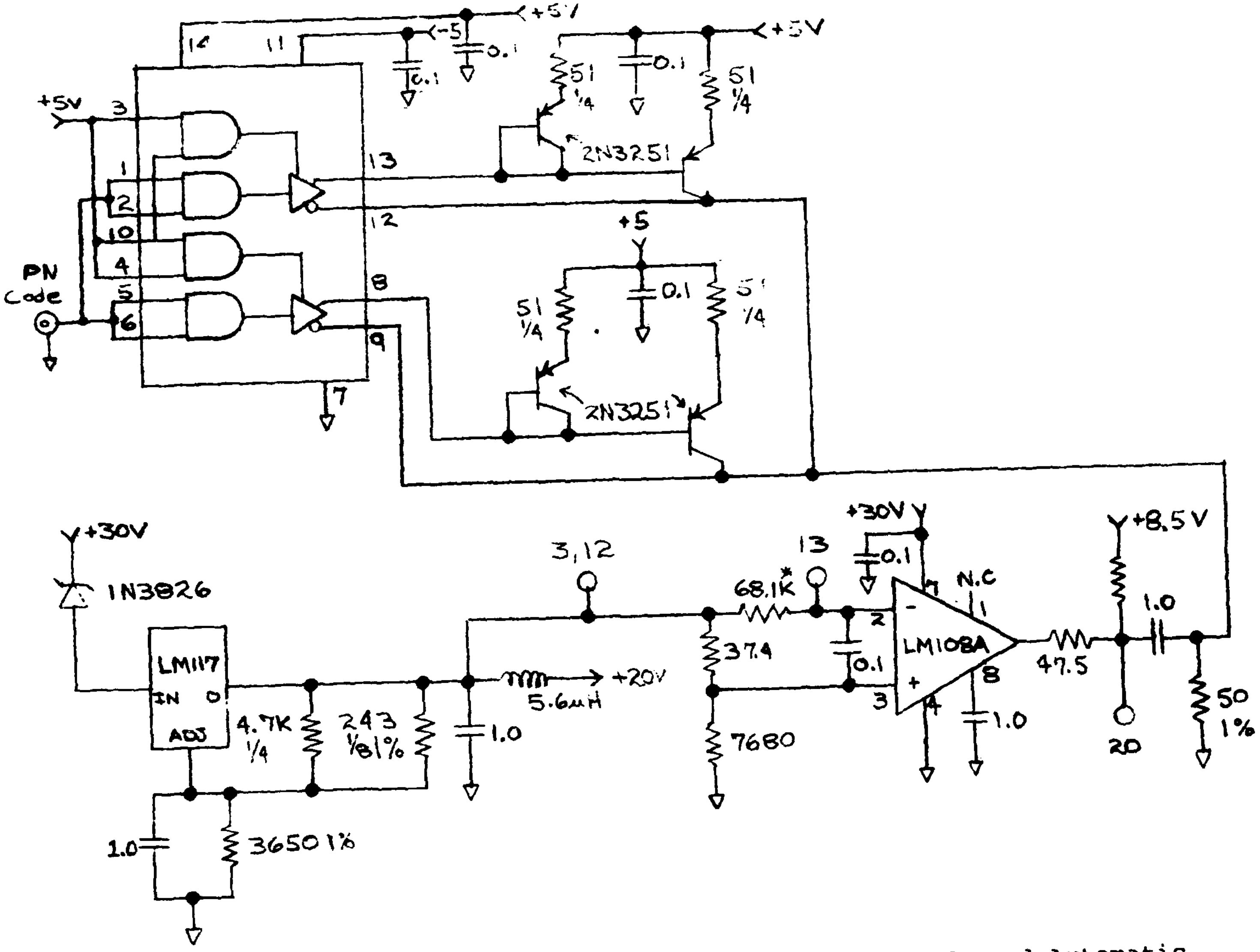


Figure 4.6b Driver Circuit for Input PN Code and Automatic Bias Correcting Circuitry for Fat-Zero

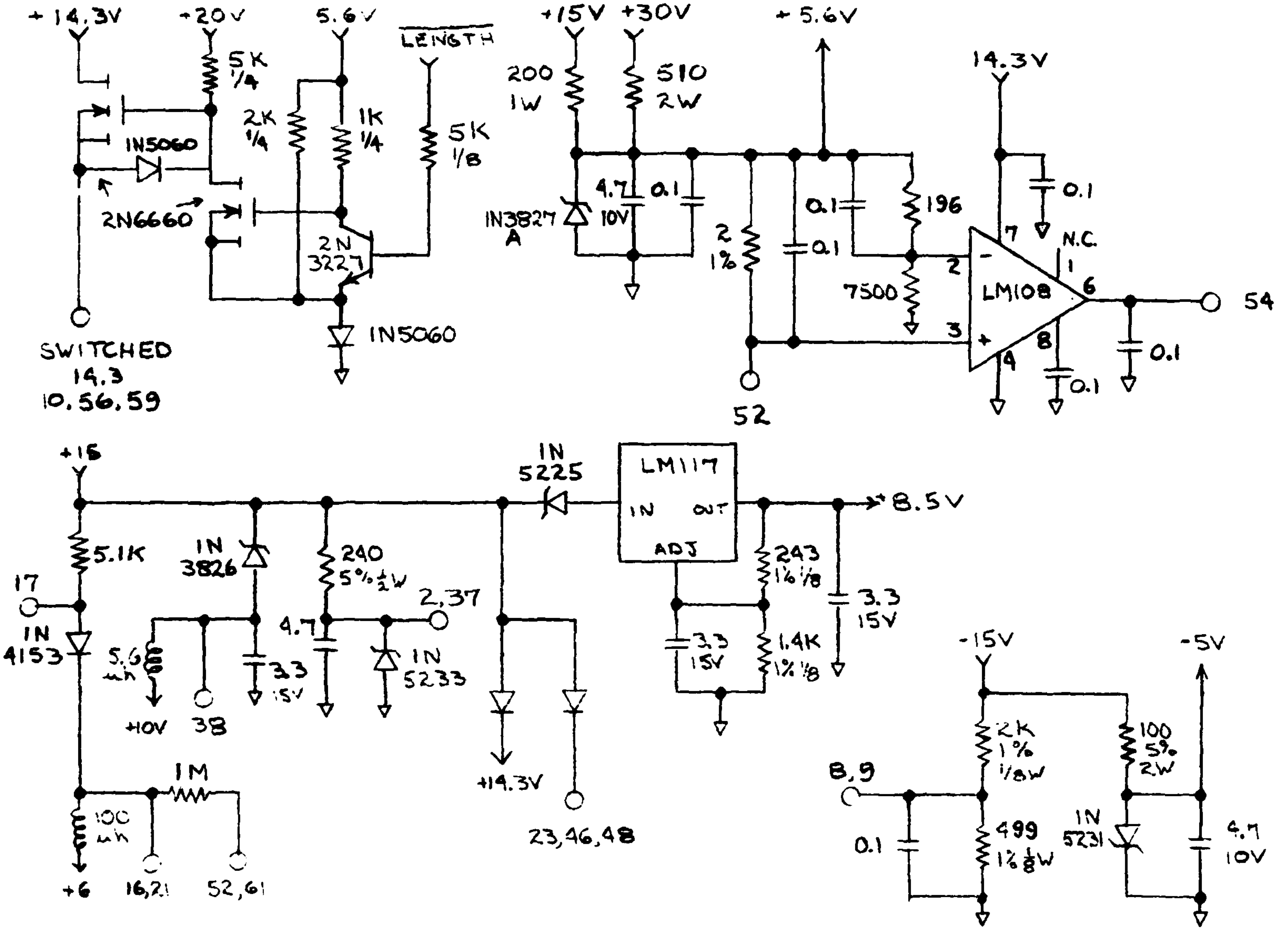


Figure 4.6c Chip Bias Circuits and Automatic Bias Correcting Circuitry for TAP FET's

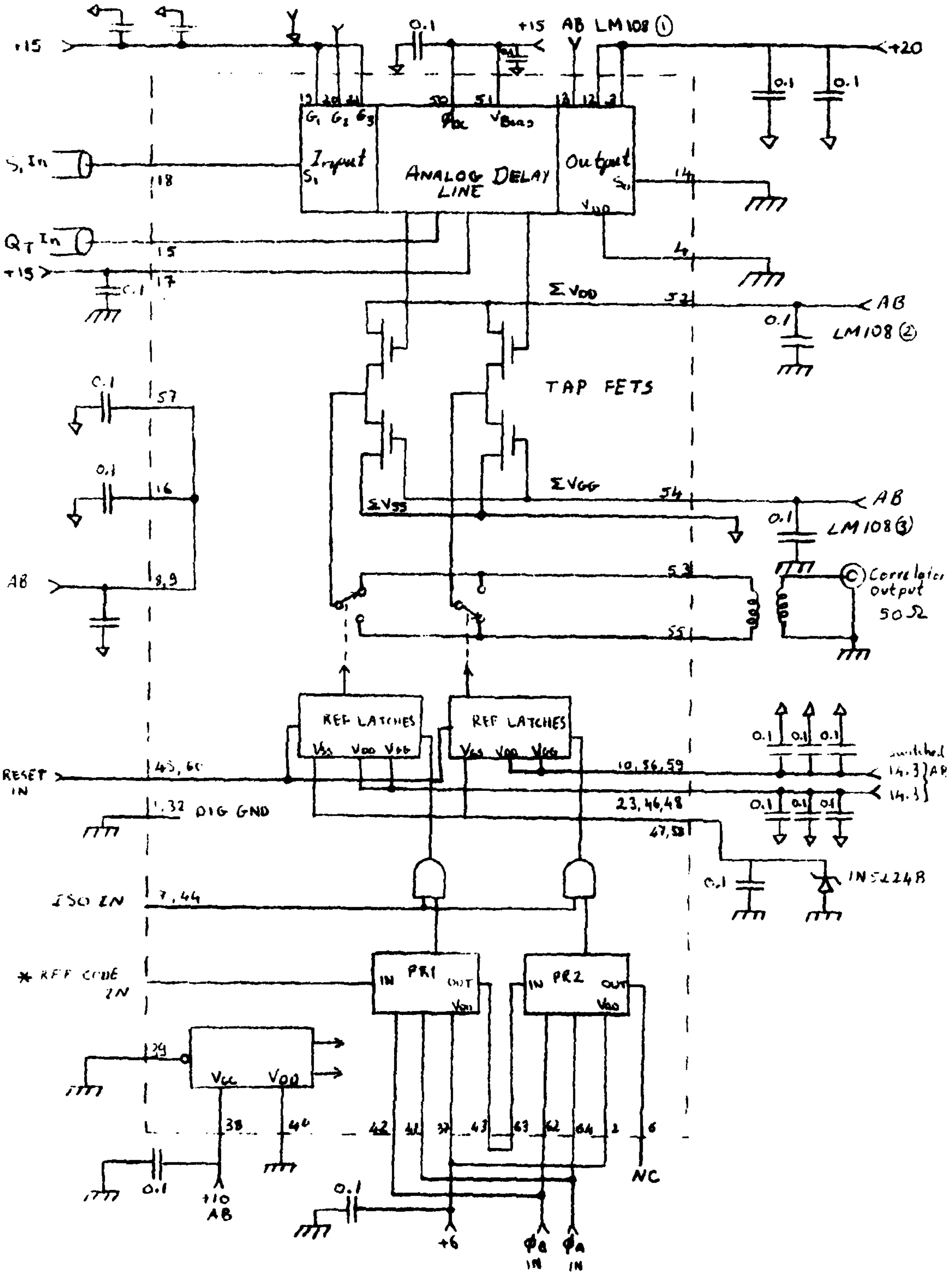


Figure 4.7 Simplified Representation of the ABC Chip

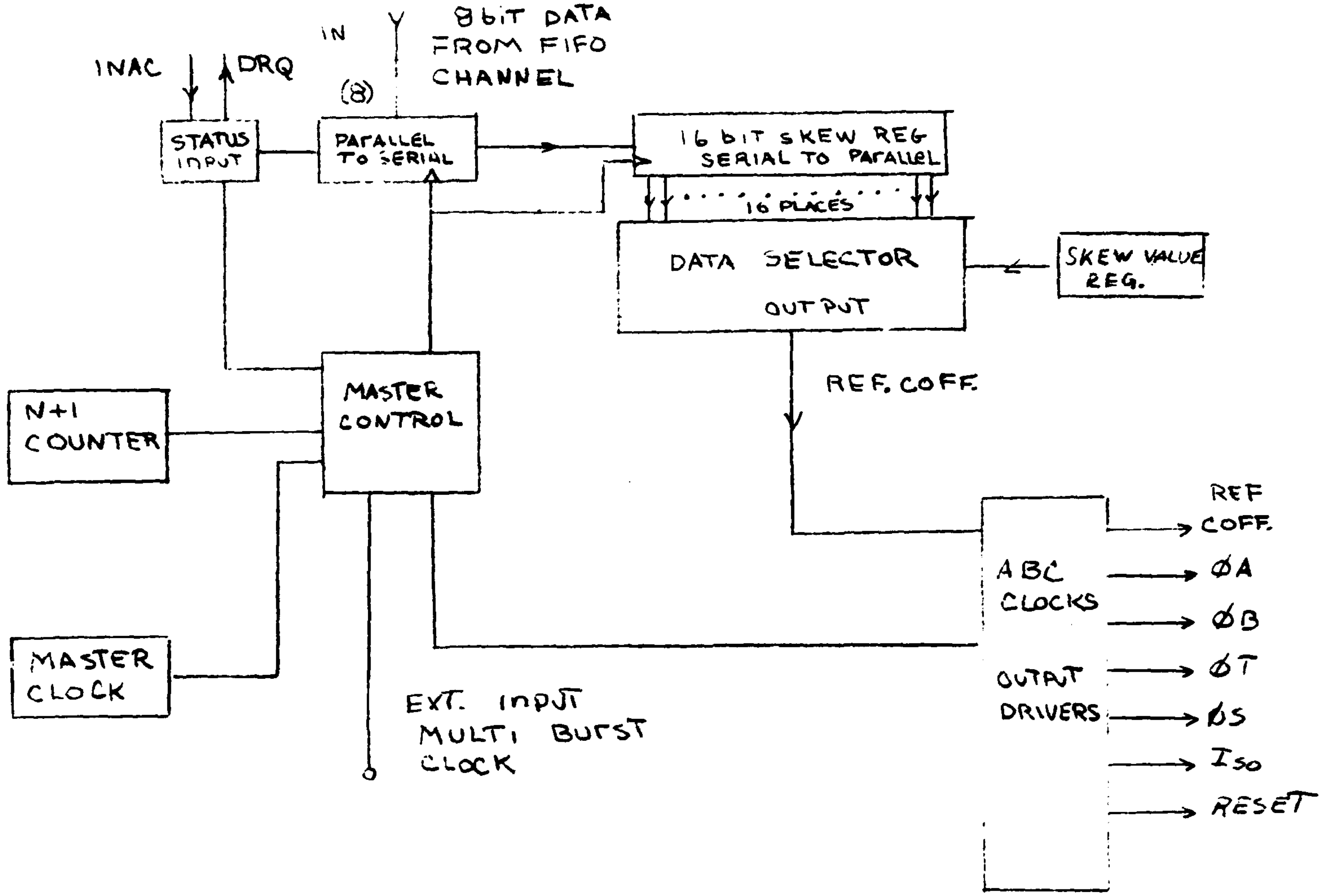


Figure 4.8 Microprocessor-ABC Interface

operation, the procedure for loading the coefficients is as follows.

A reset vector is sent to clear the system. The number of coefficients (bits) to be sent to the ABC is loaded into the N+1 register. The skew value to be used by the data selector is sent to the skew register. The start of data transfer can begin by loading the control flag register. To check the status of the system, the flag register has a start bit, which when set, shows the system is busy.

Once the system is "started," a request is sent to the FIFO channel for coefficient data. The eight bits of data is received by an input buffer. This is then converted from parallel to serial. The serial coefficient data is held by a 16-bit skew register. The skew value is sent to the data selector which taps the coefficient bit from the data stream. The coefficient signal is conditioned and sent to the ABC.

The N+1 counter is used to control the number of coefficient bits sent to the ABC. In turn, this is controlled by the value of the N+1 register. An external multi-burst clock is used to control the rate at which the block of N+1 data bits is repeatedly sent to the ABC.

The master control generates ϕ_A, ϕ_B, ϕ_T and ϕ_S from the master clock. These signals are used to synchronize the coefficient data on the ABC board. All signals sent to the ABC are conditioned to the appropriate drive levels.

There are four eight-bit registers used in this interface.

These registers, shown in Figure 4.9, are as follows:

R_0, R_1: These two registers hold the value of the N+1 counter. This is the number of coefficient bits to be sent to the ABC. We note that the value sent to these registers is N.

R_2: This register holds the skew value sent to the data selector. The value ranges from 0 through F_H.

R_3: This is the control flag register.

D_0 = RESET: Resets the entire system.

D_1 = START: When set, data transfer begins.

D_2 = BURST: When set, the system operates in the single bit mode and will move N+1 data bits out. When cleared, the system operates in a multi-mode and will transfer N+1 data bits out on each burst clock pulse.

D_3 = BRESET: Burst mode reset will stop the burst mode of operation (set the B_2 bit).

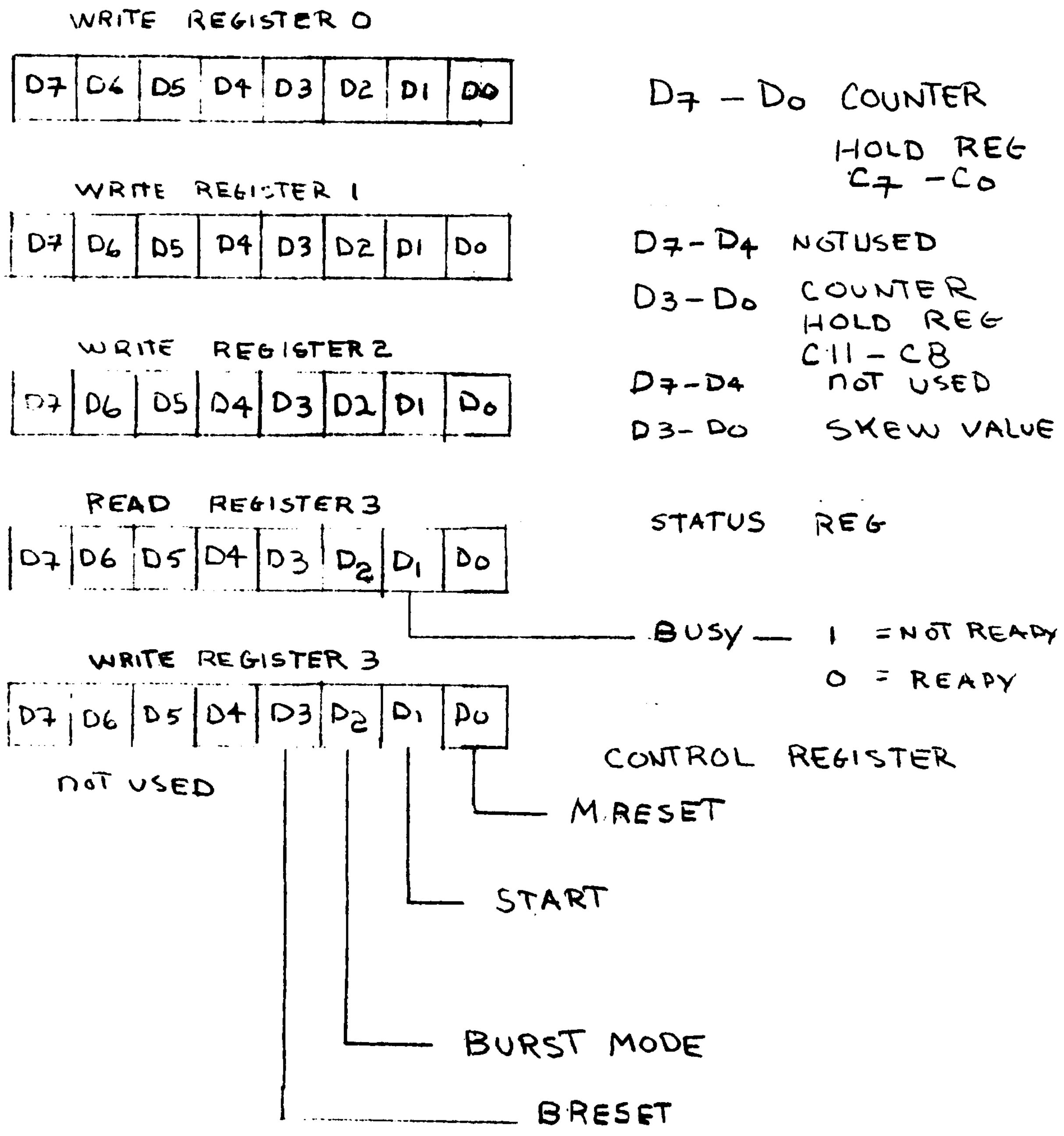


Figure 4.9 The Four Registers in the Microprocessor-ABC Interface

5. Programmable Transversal Filter

Introduction

The PTF is a 1,024-stage CCD transversal filter with electronically programmable tap weights. Each of the weighting coefficients is decomposed into a binary representation, +1 or 0 weighting, which can be loaded into a static shift register. The coefficient code in the static shift register then determines the relative timing of charge transfer in the four-phase CCD register. If the charge is transferred early there is a contribution to the integrator output.

The PTF incorporates on chip analog and digital support circuitry. This monolithic analog signal processing system has the flexibility to be operated in nine programmable configurations, from 1,024 stages by 1-bit to 128 stages by 8-bits as indicated in Table 1.1 (page 15). Figure 5.1 is a simplified block diagram of the architecture of the PTF. As indicated in Figure 5.1, there are eight 128-stage dual-differential four-phase CCD's, scaling circuitry for achieving multi-bit accuracy, i.e., signal weighting at either the input or output by factors of 1, $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{16}$, switches for selection of filter configuration, unity gain charge amplifiers (CVAMP's) for linking the output of one 128-stage CCD to the input of the next and differential current integration (DCI's). Sixteen static binary registers of 64-bits each, two for each of the dual-differential 128-stage CCD's, are not indicated in Figure 5.3.

An on-chip four-bit to nine-line decode circuit is used to externally select any one of the nine possible filter

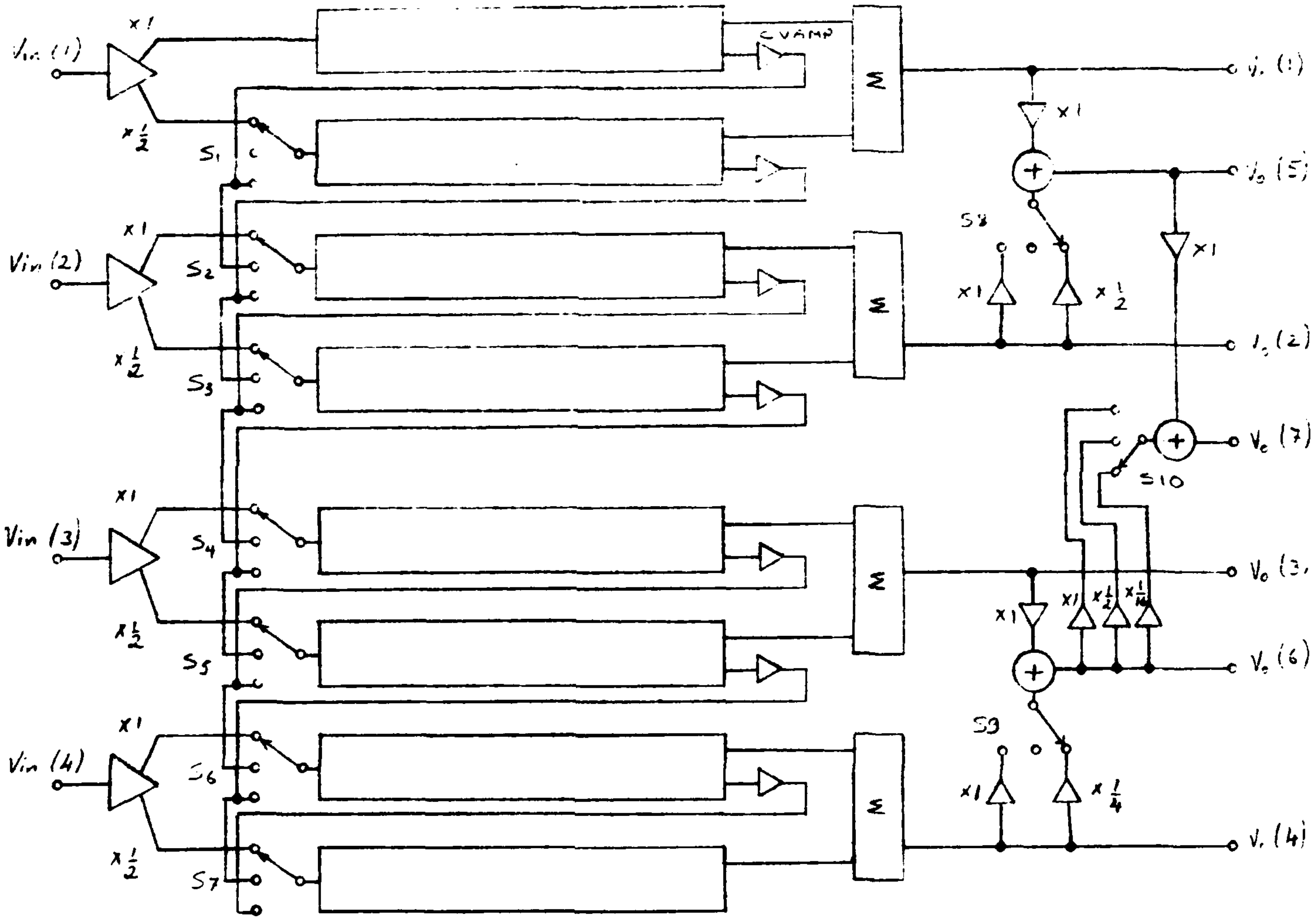


Figure 5.1 Block Diagram of P[illegible] 128-Stage by 2-Bit Filter

configurations. Table 1.1 (page 15) indicates the input word code for each filter.

On chip interface circuitry has been incorporated to permit microprocessor loading of the static shift registers. An asynchronous strobe pulse from the microprocessor indicates it is ready to send new data. A data acknowledge from the PTF enables the microprocessor and new data is loaded into the static shift register serially. In the absence of a strobe pulse the previous data is recirculated in the register.

The PTF chip requires 13 clocks, all of which are derived on-chip from a single master clock input that operates at four times the system frequency. The maximum system frequency of 1 MHz is limited primarily by the power dissipation of the drivers.

A more detailed discussion of the physical construction and operation of the PTF can be found in References [5] and [6].

In the course of the design and packaging of the PTF system the original chip has been modified twice. PTF-1 was the original chip with which we were concerned. Difficulties with loading has resulted in a third generation chip, PTF-3. Packaging for PTF-1 and PTF-3 as well as changes in the circuitry will be discussed in the following sections.

PTF-1

The block diagram of Figure 5.2 indicates the basic architecture of the peripheral support for PTF-1. The physical system consists of two PC boards in an eight-inch rack-mountable shielded cage and a TM 506 power module with Option-02. The TM 506 power

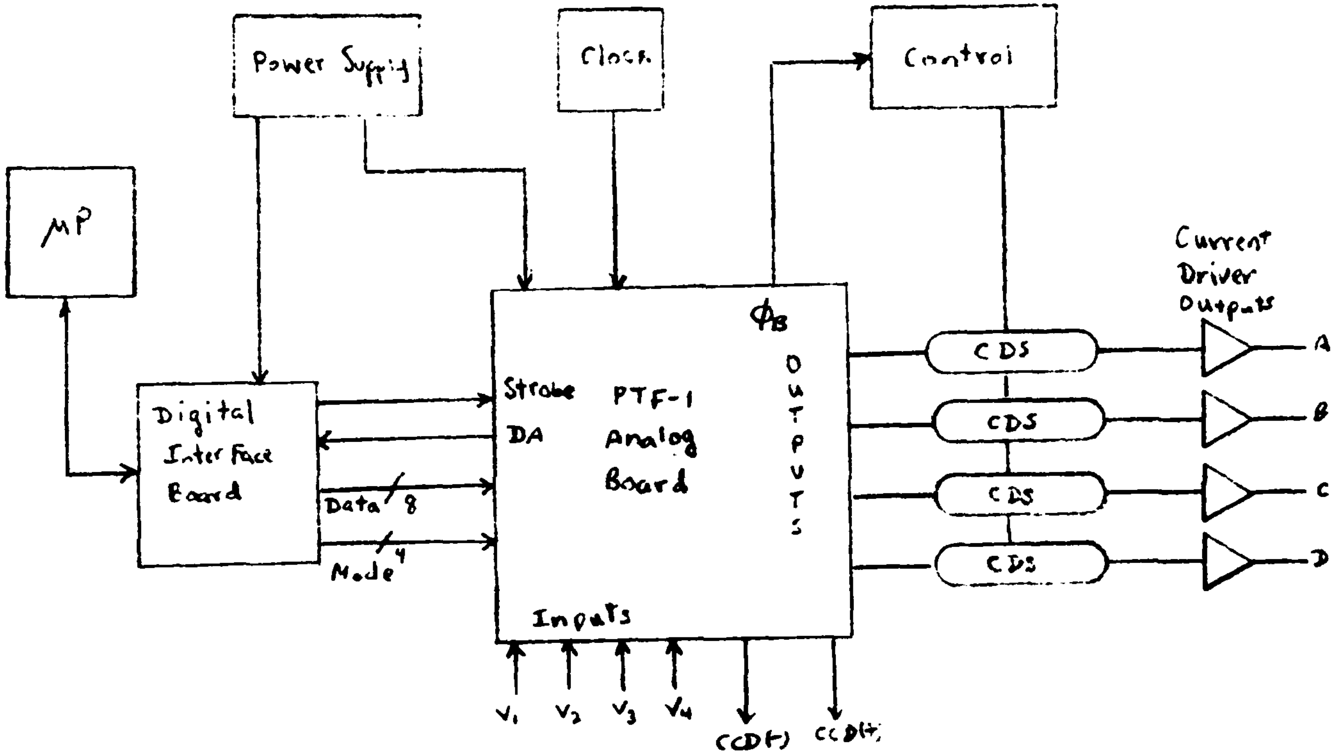


Figure 5.2 Block Diagram of PTF-1 Peripheral Support Circuitry

module houses a power supply, PTF master clock and four bins for the four correlated-double-sampling (CDS) outputs. One PC board is the digital interface board between the microprocessor and the PTF system. A second analog PC board contains the chip, coefficient amplifiers, bit driver buffer amplifiers, modė buffer amplifiers and bias supplies.

The master clock circuitry is shown in Figure 5.3. The digital interface board circuitry is shown in Figure 5.4. In the interest of flexibility this is a plug-in PC board which can be modified or replaced readily. This board contains four 67401 FIFO's which are used to load the PTF with new weights from the microprocessor. The PTF loading rate is dependent on the clock rate so a self-loading technique is used such that the PTF loading rate is independent of the microprocessor load rate.

The interface board is made up of three sections: an input loading section, FIFO storage and a PTF self-loading clock. Data is loaded into the FIFO's by sending 128 bytes in the proper format which is controlled by the Mode word. The FIFO's are each capable of 64x4 bits and are interconnected to provide a 128x8 capability of 1,024 bits.

The first $\bar{W}_R$ pulse disables the output. The $\bar{W}_R$ pulse triggers a one-shot that serves the purpose of centering the $\bar{W}_R$ pulse within the valid data time, thus providing the requisite set-up and hold-time for the FIFO's. Once the FIFO's are fully loaded the IR outputs will stay low, indicating that the FIFO's may now be enabled to load the PTF.

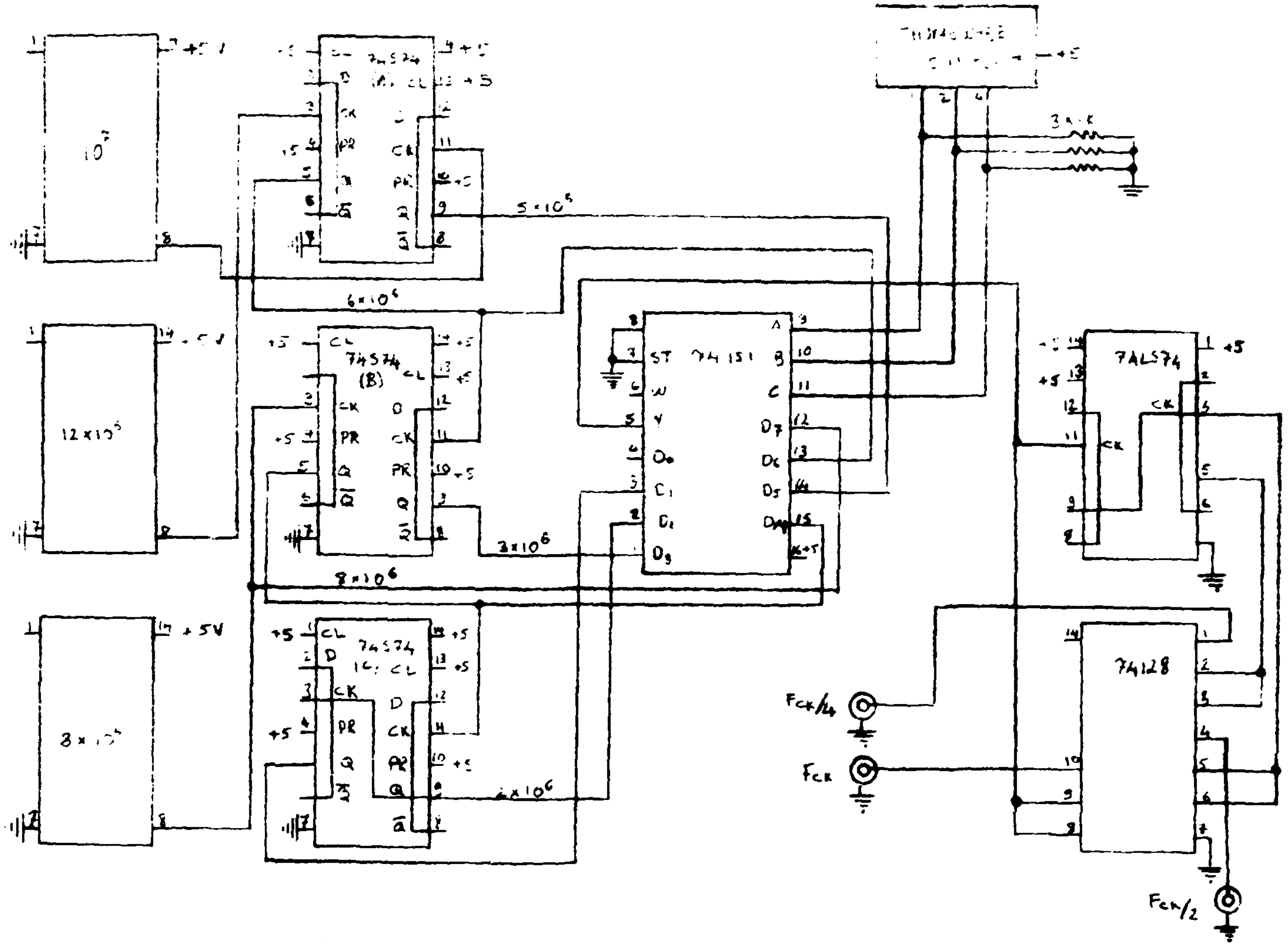


Figure 5.3 PTF Master Clock Circuit

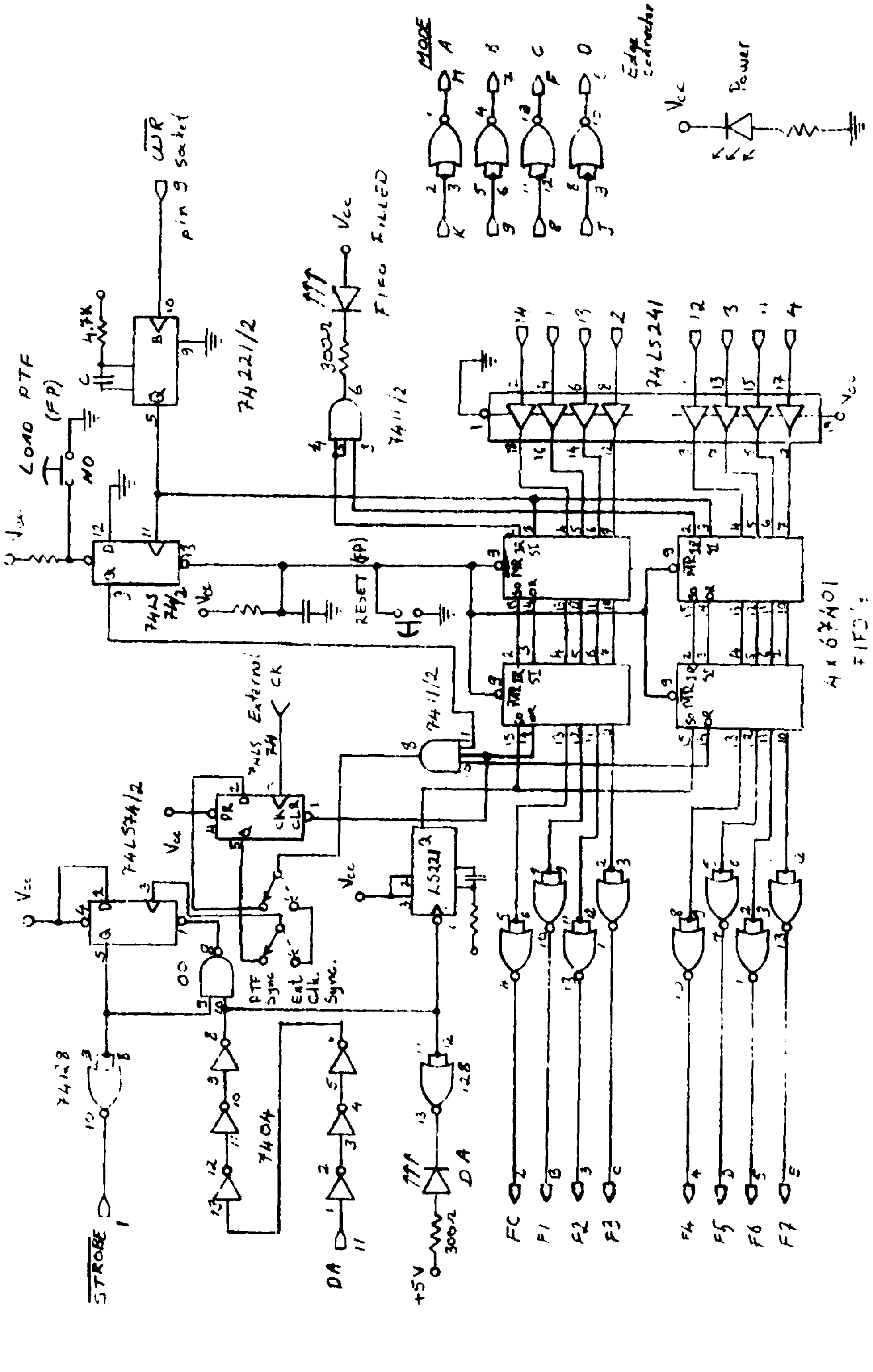


Figure 5.4 PTF Interface with DMA

The PTF clock section is made up of a D-latch and additional logic gates to control the interaction between the $\overline{\text{strobe}}$ and DA signals associated with the on-chip microprocessor support circuitry. When the DA signal from the chip is low the $\overline{\text{strobe}}$ signal will load data into the PTF. When DA is high the $\overline{\text{strobe}}$ is inactive and the FIFO's are triggered to output the next byte.

An optional sync circuit, controlled by a switch on the board, may be switched into the clock circuit. A byte is now loaded only on the rising edge of the external clock. This allows for a slower load rate and aids in debugging any system timing errors.

The PTF chip pin-out diagram is shown in Figure 5.5. Figures 5.6(a) and 5.6(b) show the peripheral support circuitry on the chip analog PC board. Mode control signals, originating on the interface board, are coupled into the chip by means of MH0026 driver amplifiers as indicated in Figure 5.6(a). Also shown in Figure 5.6(a) is one of four coefficient amplifier circuits with one of four analog signal input buffer amplifiers. Four MH0026 driver amplifiers are used for inputing the bit signals to the chip. The MH0026 IC contains two amplifiers with a common power input. Only a typical pair for the various bits is shown in Figure 5.6(b). SD210 MOS FET's in the source follower configuration are used for buffering the four output signals, and the CCD(-) and CCD(+) current outputs.

A correlated-double-sampled (CDS) technique is used to

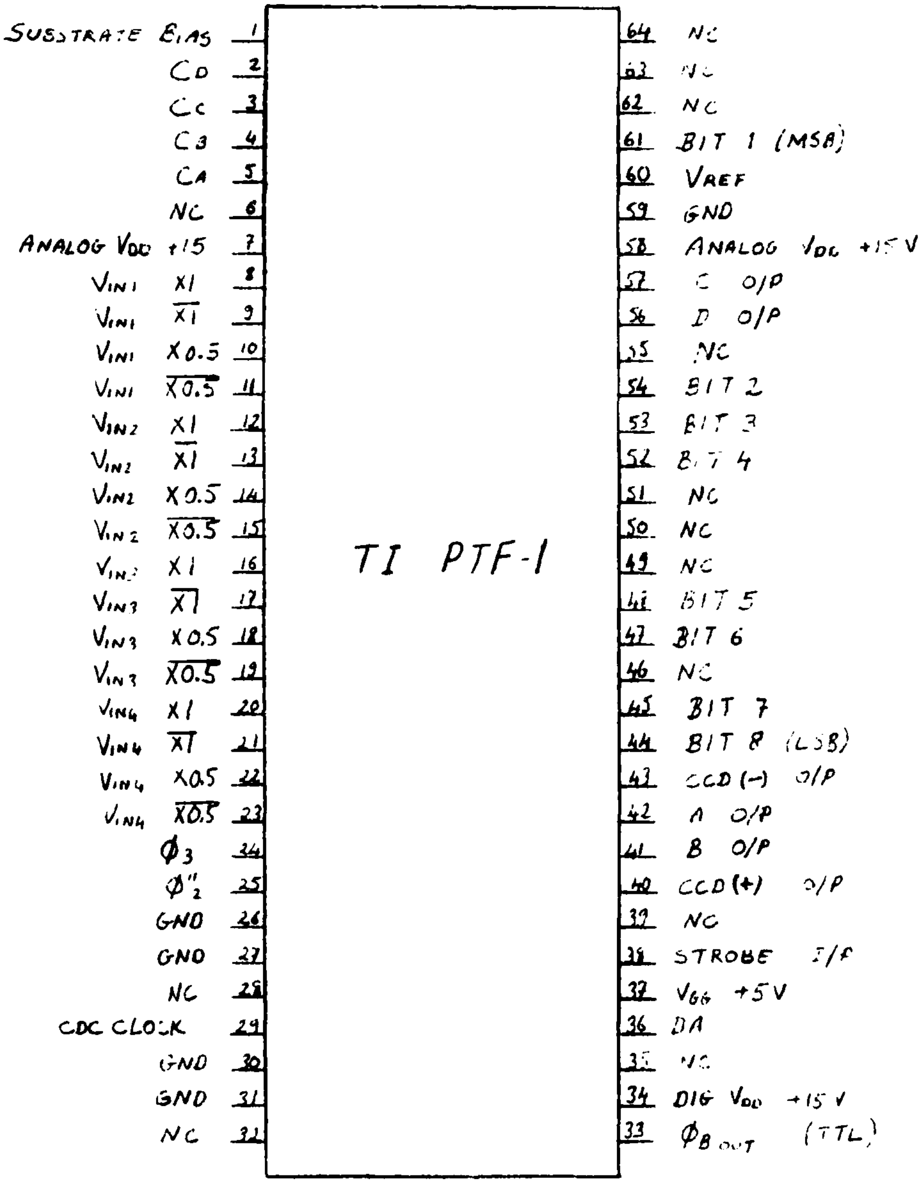


Figure 5.5 PTF Chip Pinout Diagram

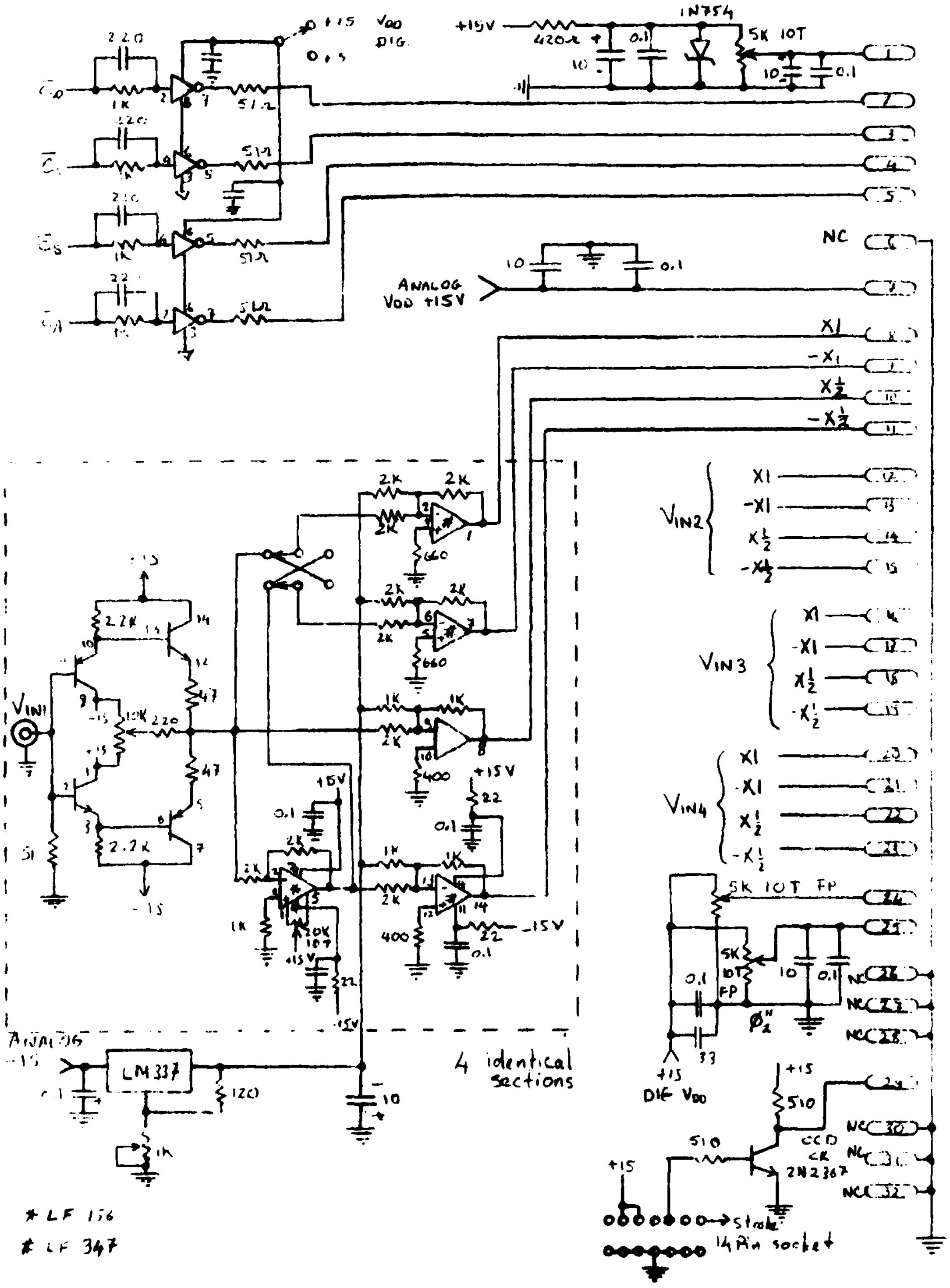


Figure 5.6a PTF Chip DC Board Peripheral Circuitry

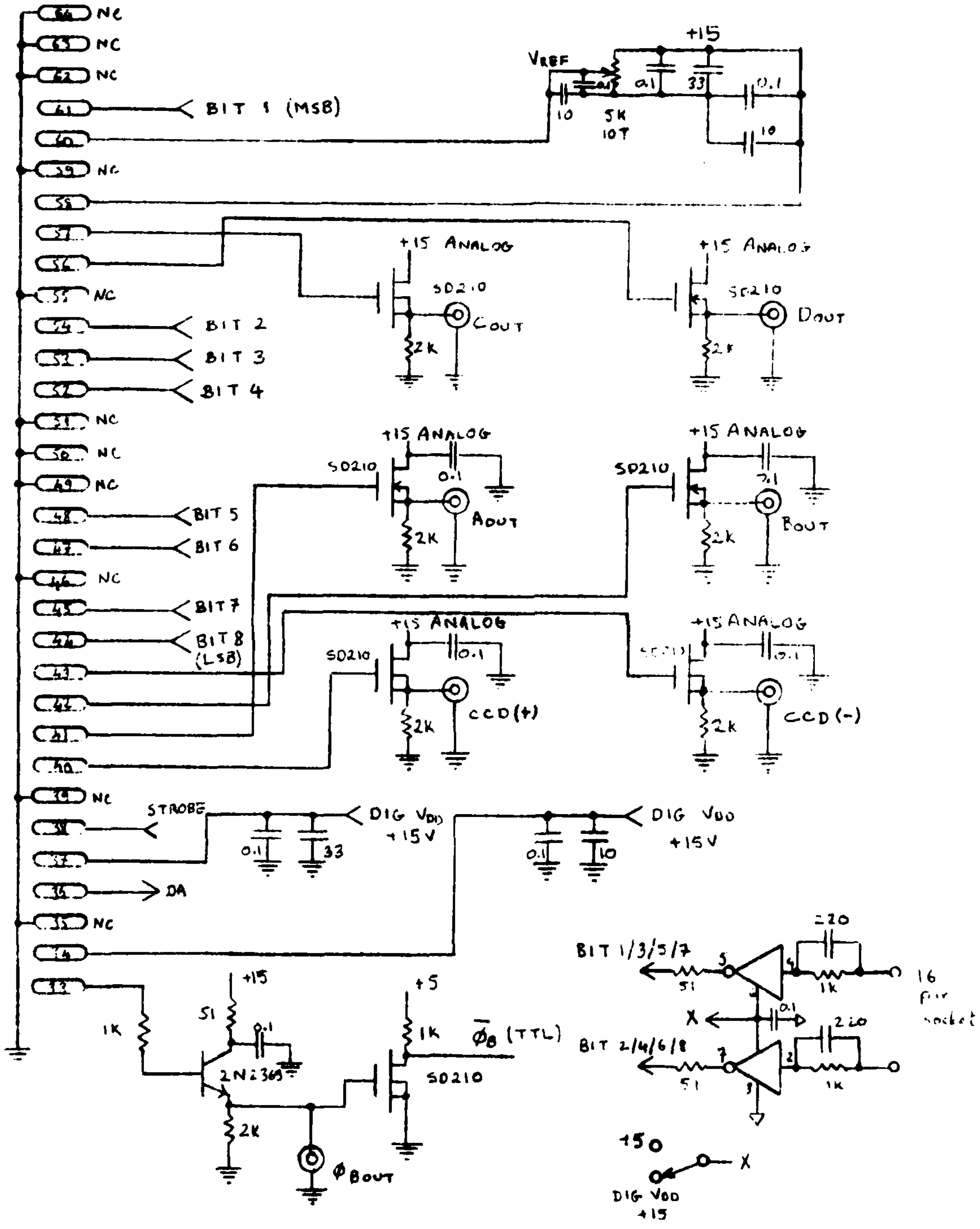


Figure 5.6b PTF Chip DC Board Peripheral Circuitry

obtain the final correlation output. A simplified version of the CDS technique is shown in Figure 5.7. Closure of SW_1 results in an initial charge on C_1 of V_{REC} which can be used to eliminate any undesirable offset at the CCD output. With SW_1 open and SW_2 closed, the CCD signal is coupled to the input of A_2 and C_2, the hold capacitor.

The CDS circuitry used for each of the PTF outputs is shown in Figure 5.8. Clock voltage ϕ_B, obtained from the chip [see Figure 5.6(b)] is used with 2N74221 monostable circuits to obtain the necessary timing for the sample-hold operation. The input switch of Figure 5.7 consists of an SD210 MOS FET driven by an MH0026 driver amplifier. An Analog Devices THC-0300 is used for the out S/H function followed by a current driver output amplifier LH0002.

Third-Generation PTF's (PTF-3)

Two difficulties were encountered with the PTF-1's. One, a technical difficulty, was uncertainty about satisfactory loading of the coefficient code into the static shift register. The second, an availability problem, was that fully functionable PTF-1 chips were not obtainable. That is, the chips that were available were not sufficiently functionable to perform other than the most simple filter functions.

Third-generation chips, PTF-3's, fully functional, did become available with modifications enhancing the loading capability. Other changes involved on-chip scaling circuitry and reduced complexity with respect to peripheral support. These

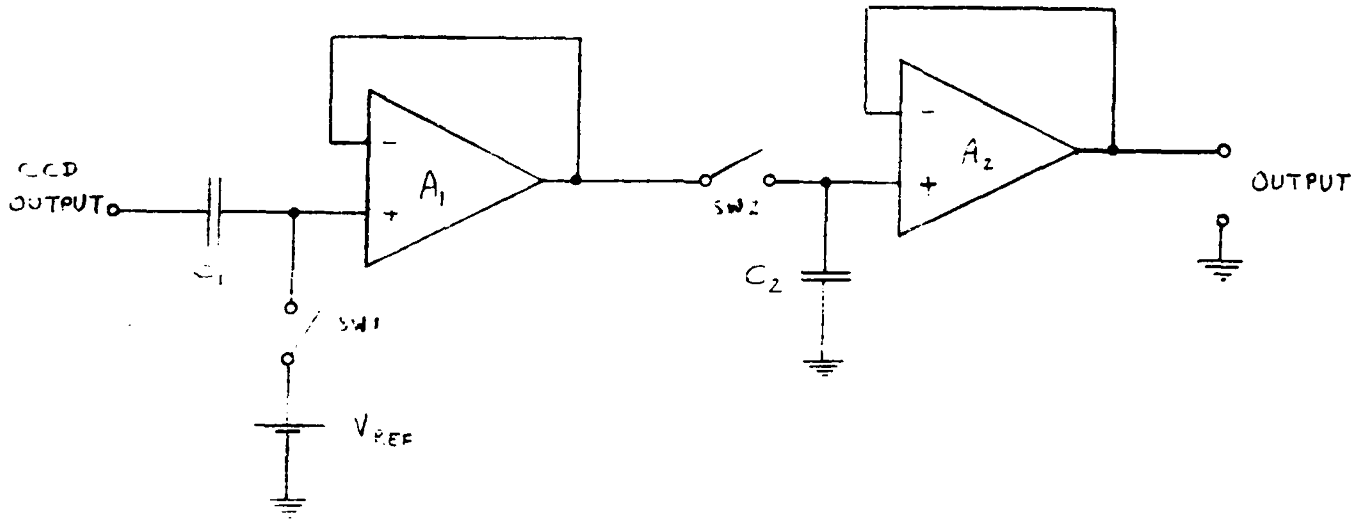


Figure 5.7 Simplified CDS System

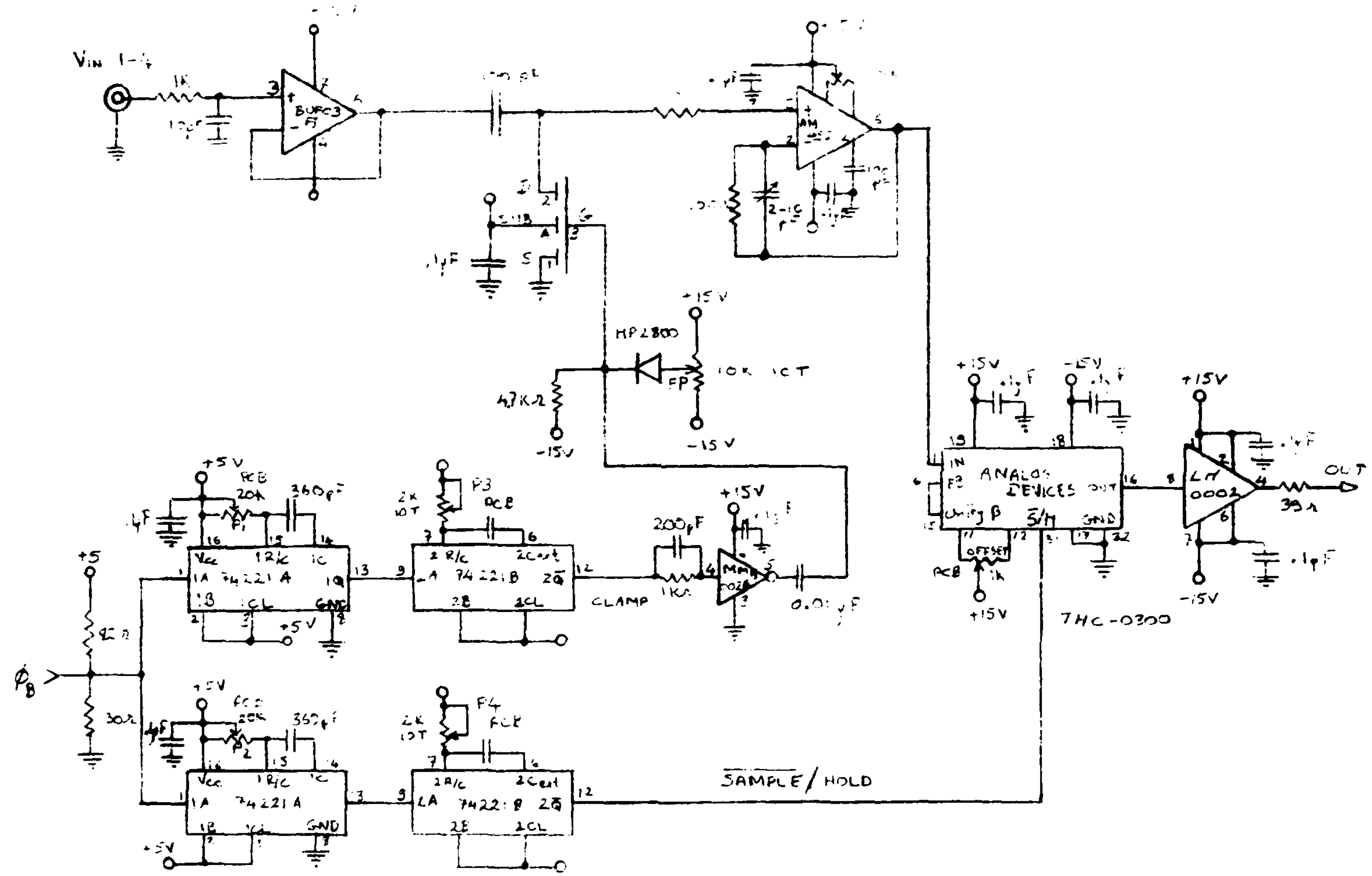


Figure 5.8 PTF CDS Circuitry

changes led to a significant reduction in the size of the physical system. For example, the PTF-3 chip, all four CDS units and the circuitry for the original chip analog boards has been fitted into a Tektronix single bin. The entire system, including power and clock circuitry, is housed in a single TM 504 module.

6. Signal Generator

The signal generator is driven by the microprocessor through the high-speed FIFO port on the interface board. The device is basically a digital-to-analog converter (DAC). The output of the signal generator is an analog signal which becomes the input to the CCD modules.

Figure 6.1 illustrates the circuit diagram for the signal generator. The corresponding timing diagram is shown in Figure 6.2. The operation of the circuit is as follows. The master clock triggers a negative-going pulse from a digital one shot. On the falling edge A the data from the position scaler is clocked into the "B" latch and is now available for the DAC to use (see G in Figure 6.2). On the positive edge of the wave, the address counter is clocked to the next state for the PROM.

The output of the PROM (D_1, D_2, D_3) is routed to the position scaler. The input of the position scaler can be shifted zero to seven bits. The three most significant bits from the PROM are used to control an "AND" mask. This mask is used to force the least significant bits from the position scaler low. A varying number of least significant bits to the D/A converter have to be forced low, depending on the mode of operation, e.g. one, two, and four-bit modes. A LOAD/CLR signal (D_0) is routed to the address counter and a second one shot is enabled.

At the end of the byte cycle when data is needed, the one shot is enabled by the LOAD/CLR line B. On the next clock pulse (output from first one shot), the second one shot is fired (see C in Figure 6.2). This positive-going pulse clocks new input data to the "A" latch from the FIFO (see F, Figure 6.2). This

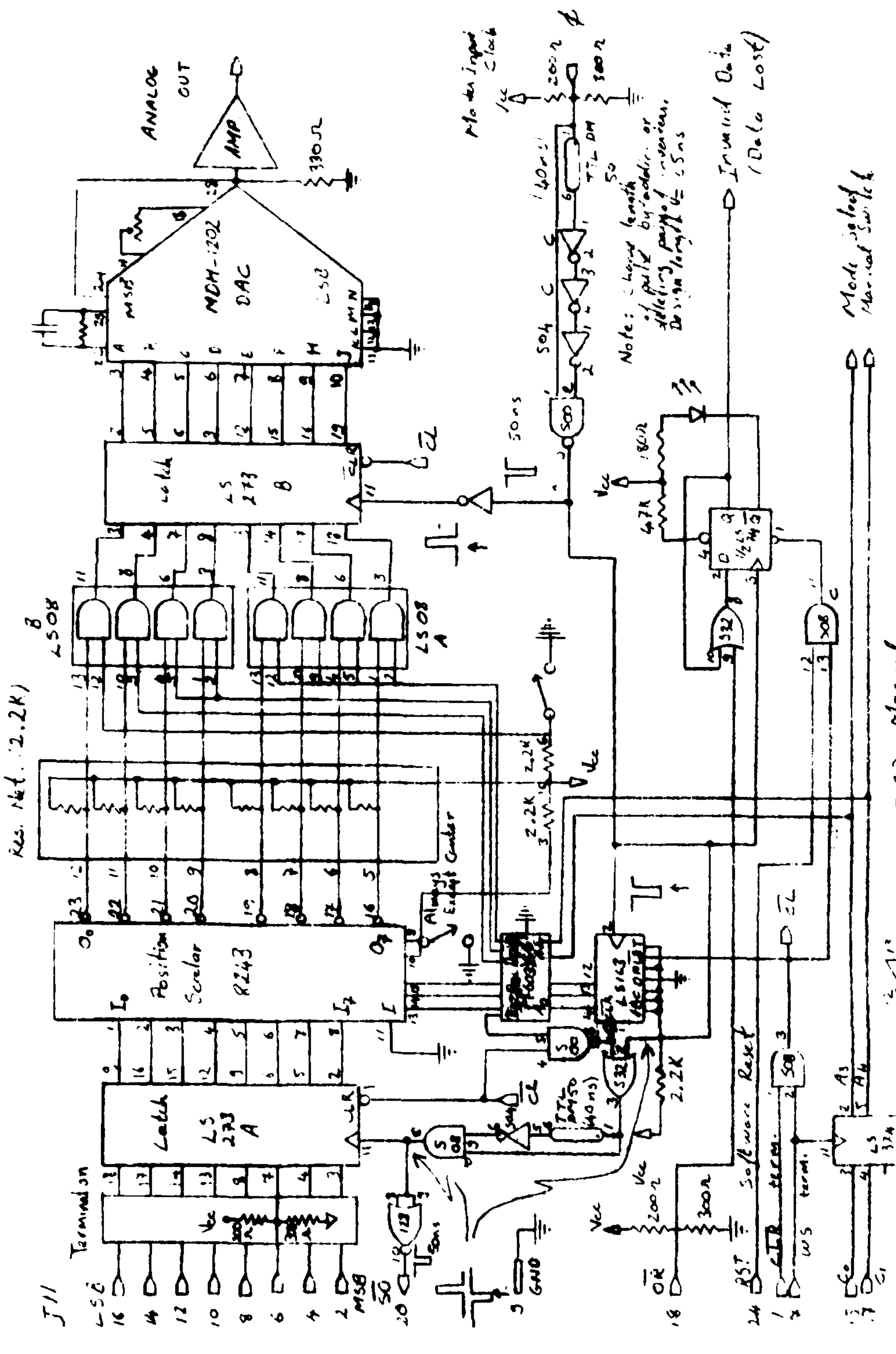


Figure 6.1 Circuit Diagram for the Signal Generator

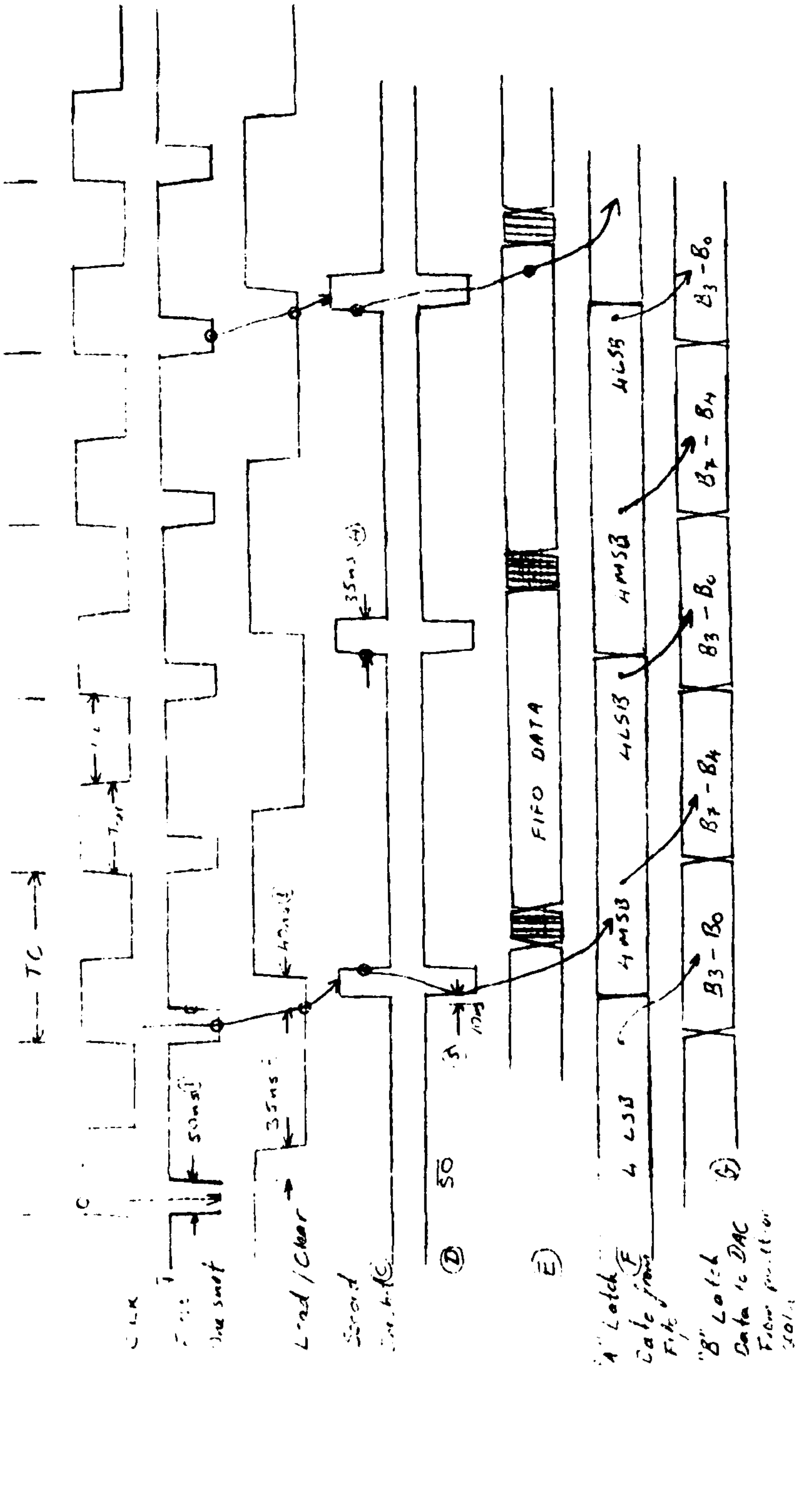


Figure 6.2 Timing Diagram for the Signal Generator Circuitry

pulse is complemented to provide a $\overline{SO}$ (shift out) signal to the FIFO (see D, Figure 6.2). The rising edge of the $\overline{SO}$ will strobe new data out of the FIFO (see E, Figure 6.2).

7. Design of Two-Dimensional DFT

In this section we describe the design of a module for performing a two-dimensional discrete Fourier transform (DFT). The module consists of two CCD's, each of which performs a DFT via the chirp-z transform algorithm, and two corner turning memories (CTM) for reformatting the two-dimensional data.

If d(r,c) represents an NxN real-valued data array where r denotes the row index and c the column index, the two-dimensional DFT is

$$D(k,\ell) = \sum_{r=o}^{N-1} \sum_{c=o}^{N-1} d(r,c)\, e^{-j(2\pi/N)rk}\, e^{-j(2\pi/N)c\ell}$$

$$= \sum_{r=o}^{N-1} D_1(r,\ell)\, e^{-j(2\pi/N)rk} \quad , \quad \begin{array}{l} k = 0,1,\ldots,N-1 \\ \ell = 1,2,\ldots,N-1 \end{array}$$

$D_1(r,\ell)$ represents the DFT for the r^{th} row of d(r,c) and it is, in general, complex. It can be computed using the chirp-z transform, according to the relation

$$D_1(r,\ell) = \sum_{c=o}^{N-1} d(r,c)\, e^{-j(2\pi/N)c\ell}$$

$$= \left\{ \sum_{c=o}^{N-1} [d(r,c)\, e^{-j(\pi/N)c^2}]\, e^{j(\pi/N)(\ell-c)^2} \right\} e^{-j(\pi/N)\ell^2} ,$$

$$\ell = 0,1,\ldots,N-1$$

The operations involved in this computation are premultiplication of d(r,c) by the chirp signal $\exp(-j\pi c^2/N)$, convolution of the product sequence with the chirp filter having the impulse response $\exp(j\pi c^2/N)$ and post-multiplication of the filter output by $\exp(-j\pi \ell^2/N)$. Similarly, $D(k,\ell)$ is computed from $D_1(r,\ell)$ by means of the chirp-z transform relation

$$D(k,\ell) = \left\{ \sum_{r=o}^{N-1} [D_1(r,\ell)\; e^{-j\pi r^2/N}]\; e^{j(\pi/N)(k-r)^2} \right\} e^{-j(\pi/N)k^2}$$

The block diagram in Figure 7.1 illustrates the computation of $D(k,\ell)$.

The hardware that we have available for implementing the computation of the two-dimensional DFT consists of two Reticon chirp-z transform boards which perform a 512-point DFT and two 32 x 32 CCD CTM's. Four problems were encountered in attempting to interconnect the Reticon boards and the CTM's.

One problem is the incompatibility in the number of points in the CTM and the chirp-z transform. The problem can be solved by repeating each row of the 32 x 32 data array d(r,c) into the first chirp-z transformer 16 times. Thus, from each 32-point sequence corresponding to a row of data we create a 512-point sequence, which we denote as x(r,m). Its 512-point DFT is

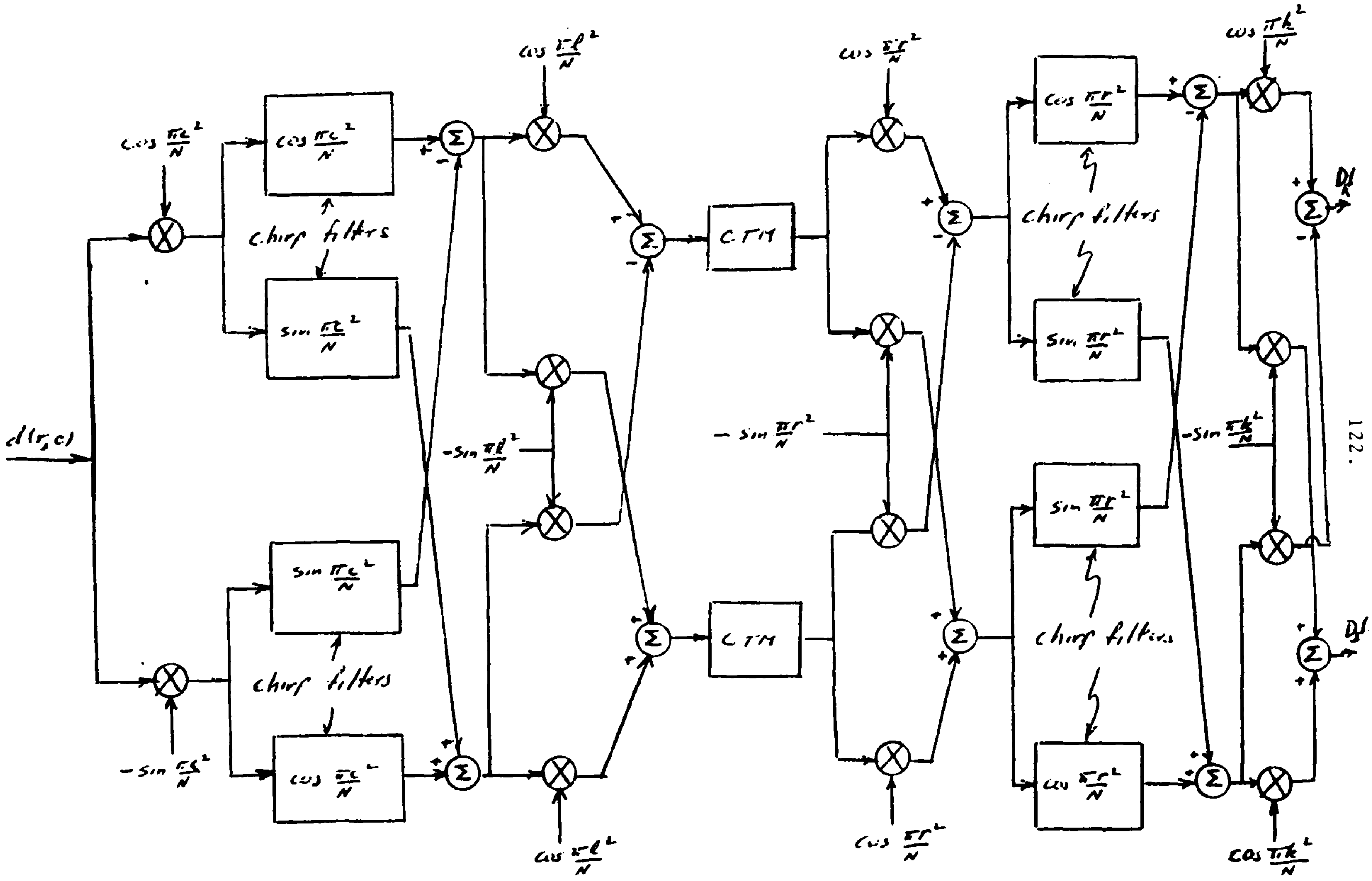


Figure 7.1 Computation of the Two-Dimensional DFT Using Two Chirp-z Transformers and Two CTM's

$$X(r,n) = \sum_{m=o}^{511} x(r,m)\ e^{-j(2\pi/512)nm}$$

$$= \sum_{m=o}^{15} \sum_{c=o}^{31} d(r,c)\ e^{-j(2\pi/512)(c+32m)n}$$

$$= \left[\sum_{m=o}^{15} e^{-j(2\pi/16)mn}\right]\left[\sum_{c=o}^{31} d(r,c)e^{-j(2\pi/512)cn}\right]$$

We note that

$$X(r,16\ell) = 16D_1(r,\ell) \quad , \quad \ell = 0,1,2,\dots,31$$

Thus, the desired 32-point DFT is obtained by selecting every 16th point at the output of the first chirp-z transform (downsampling by a factor of 16). Similarly, each row output of the CTM's must be repeated 16 times to create 512 sequences from 32-point sequences. The resulting sequence is the input to the second chirp-z transformer. The output of the second chirp-z transformer is downsampled by a factor of 16 to obtain the desired 32-point transform. An alternative procedure is to insert 15 zeros after each data point.

The second problem that was encountered is that the Reticon chirp-z transform board does not implement the postmultiplication with the chirp signal but, instead, it computes the magnitude of the DFT. Since the first chirp-z transformer (prior to the CTM) must implement this postmultiplication we have to modify this board by adding four multipliers and two adders as shown in

Figure 7.1.

The third problem is that the Reticon board can only handle real-valued input signals. However, the second chirp-z transformer (following the CTM) must accommodate complex-valued data. This means that the second board must be modified by the addition of two multipliers and two adders.

The fourth problem that was encountered is the incompatibility in the clock rate for the CTM and the chirp-z transformers. The Reticon boards can operate at a maximum sampling rate of approximately 200 kHz while the CTM does not operate very well at clock rates (sampling rates) below 1 MHz. Although the CCD's used in the computation of the chirp-z transform can operate up to about 2 MHz, the support circuitry in the Reticon board limits the speed to below 200 kHz. This problem cannot be solved easily by modifying the support circuitry of the Reticon device to run at a higher speed nor can the CTM be modified to run at a lower clock rate.

For purposes of demonstrating the computation of the two-dimensional DFT the most feasible solution is a digital implementation of the CTM's with clock rates that are compatible with the Reticon chirp-z transformers. Toward this end we designed a digital CTM for performing the column and row interchange required in the two-dimensional DFT. It is called CREM (column-row exchange memory), in order to distinguish it from the CCD CTM.

CREM is composed of five sections as shown in Figures 7.2 and 7.3. The first is the input section which consists of two

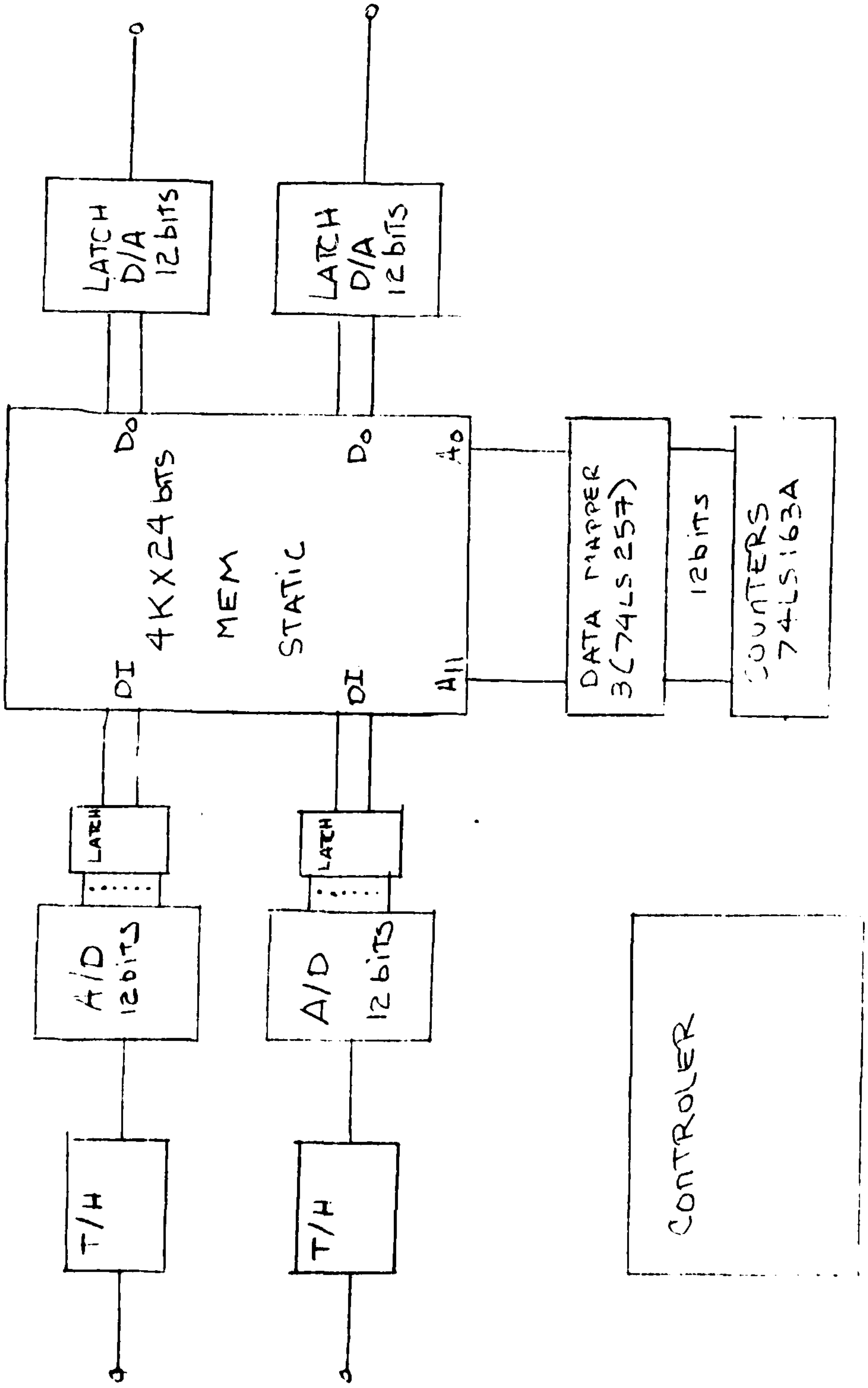


Figure 7.2 General Block Diagram of CREM

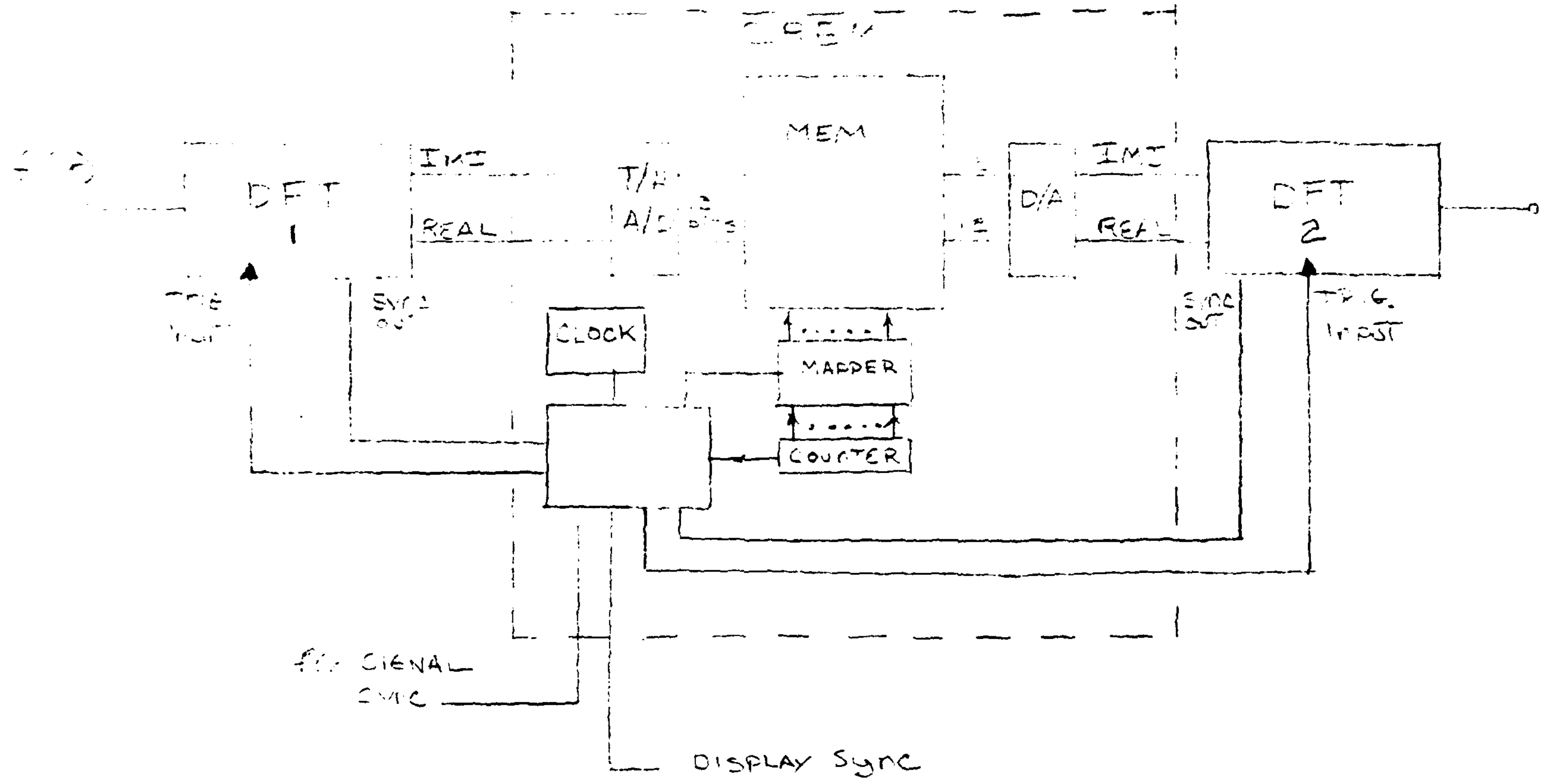


Figure 7.3 Connection of CREM to Reticon Chirp-z Transformers

channels with track and hold input buffers. The A/D converters have a resolution that can be selected between 8 bits and 12 bits with conversion times of five microseconds to eight microseconds, respectively. The maximum sampling rate is approximately 125 kHz for the 12-bit resolution.

The second section is the exchange memory, which is comprised of 2k by 8-bit chips. We chose to implement a 64 x 64 transformation, which implies that we need 4K by 12 bits for each channel. The cycle time of the memory is under 500 nanoseconds, so that interfacing to the A/D converter is easily accommodated.

The output section of the CREM consists of two latch 12-bit D/A converters having the appropriate speed to interface to the second Reticon board. The use of a latch D/A should preclude the need for the sample-and-hold (S/H) units at the analog input to the real and imaginary channels of the second Reticon board.

The address mapper is the fourth section of CREM. It controls the exchange of row data with column data. The standard mode of operation is to read data out of a location, latch it to the D/A, write new data into that location and, then, to increment to the next location. After one frame (64 x 64 points), the mapper will exchange the six least significant bits with the six most significant bits, thus exchanging the rows and columns that will be accessed. At the completion of the frame the mapper is switched out and the cycle repeats. The mapper will switch in and out on alternate frames.

The fifth section is the timing control section. It issues commands to both Reticon boards for input and output synchronization with the track-and-hold buffer and the output D/A. Both Reticon boards are slave to the master timing of the CREM. The control section will output timing to synchronize the input waveform to the first Reticon board and the output of the second Reticon boards. An optional control of timing allows for either a continuous transformation or a one frame and hold. The system timing is shown in Figure 7.4 and the memory timing is shown in Figure 7.5.

The timing need not be changed to increase the number of points in the transform. Only an increase in memory is needed for expansion to the full 512 x 512 points. For the 64 x 64 point two-dimensional DFT, every eighth point is sampled with 12-bit resolution, corresponding to a dynamic range of 72 dB.

Lack of time and manpower prevented us from implementing the design described above.

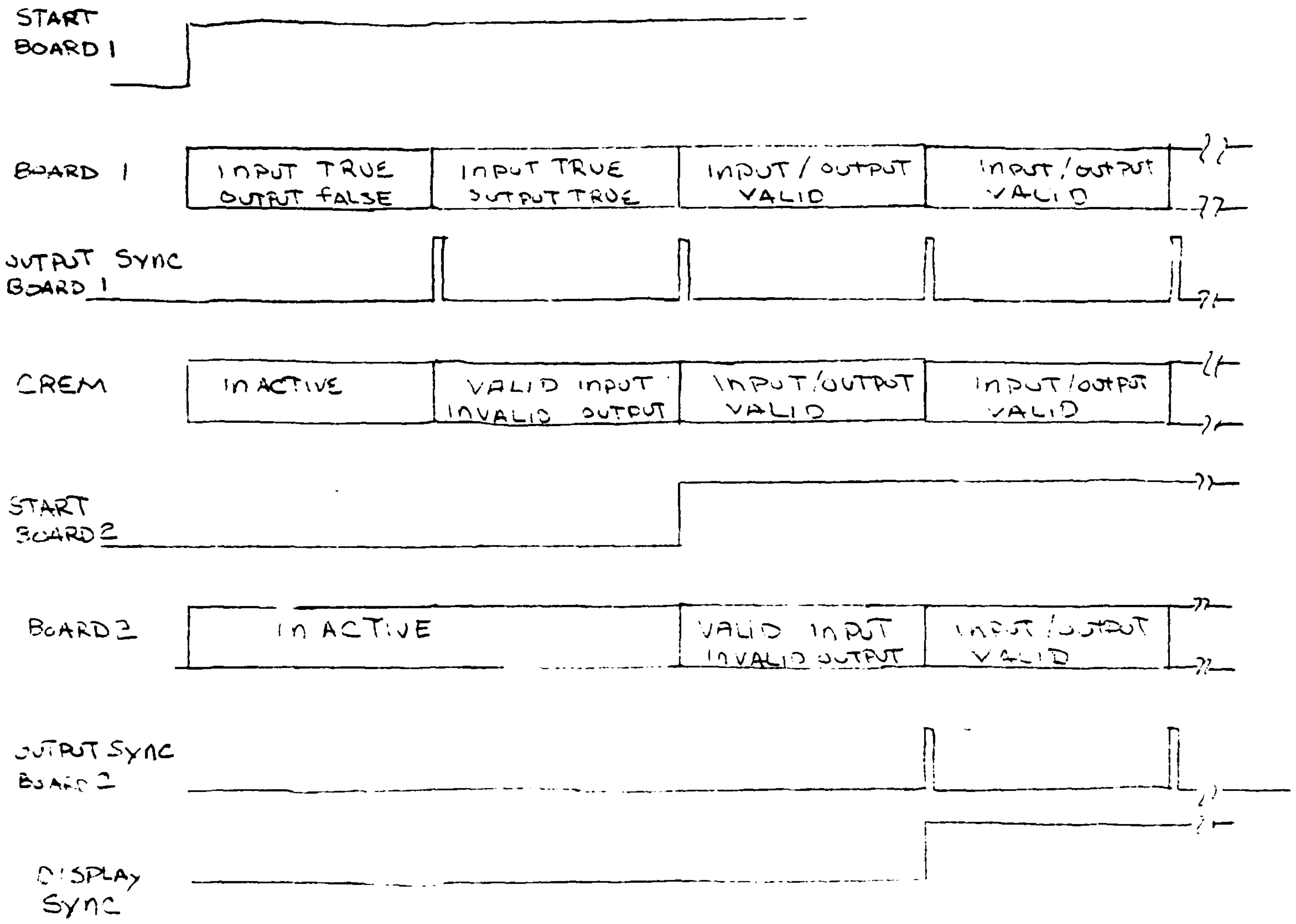


Figure 7.4 System Timing for CREM

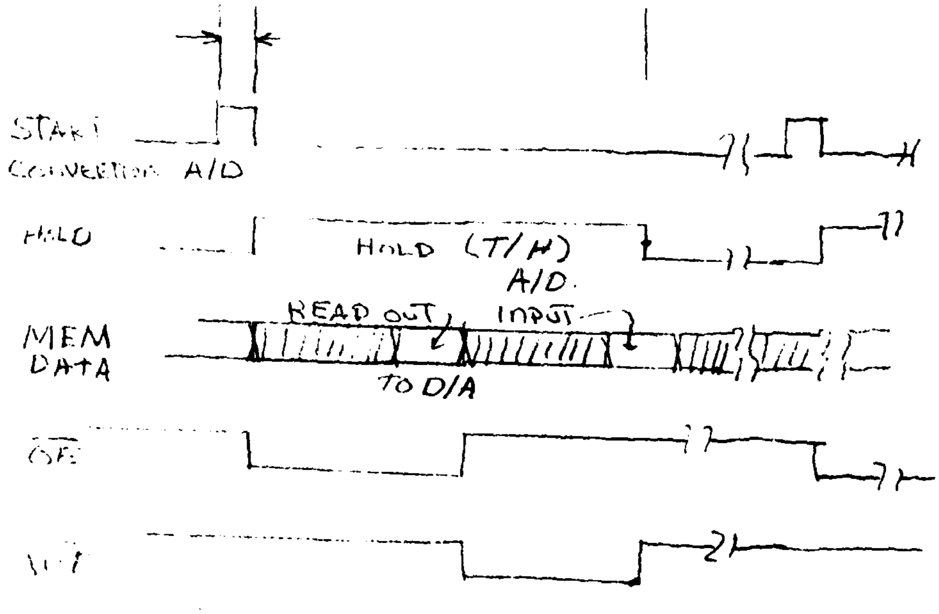


Figure 7.5 Memory Timing for CREM

8. Concluding Remarks and Further Work

Our major efforts on this program were spent on hardware development. Support circuitry was designed and built for the ABC, the PTF and the RSAM. Additional circuitry was designed and constructed in order to interface the microprocessor to the ABC, the PTF and the signal generator for the purpose of controlling the signal processing functions and for passing data. The signal generator was also designed and constructed.

In addition to the hardware developments, a number of software was developed to generate a variety of signals and other data, to pass data from the microprocessor to the ABC, the PTF and the signal generator, and to control the modes of the PTF.

Due to delays involved in the availability of the devices and the great effort expended in the development of the hardware described in this report, there was no time left to actually interconnect the devices. Further work is required to configure the modules described in Section 1 of this report. Hardware should be developed for routing an analog signal from the output of one CCD to the input of another CCD automatically, by means of microprocessor control. In addition, new signal processing algorithms and new software should be developed. This additional work should be aimed toward increasing the flexibility and range of application of the CCD modules and optimizing their performance.

References

1. F. D. Shepherd, Jr., "High Throughput CCD Signal Processing Devices", Proc. Asilomar Conference on Signal Processing (Invited Paper) November 1978.

2. Z80 DMA Technical Manual, December 1980.

3. D. A. Gandolfo, J. R. Tower, "Analog-Binary Programmable Transversal Filter", Interim Report, January 1979.

4. D. A. Gandolfo, J. R. Tower, J. I. Pridgen, S. C. Munroe, "Analog-Binary CCD Correlator: A VLSI Signal Processor", IEEE Journal of Solid-State Circuits, Vol. SC-14, No. 2, April 1979, Pages 518-525.

5. Haken, R. A., Pettengill, R. C., "Binary/Analog CCD Correlator Development", Interim Report, RADC-TR-79-211, September 1979.

6. Haken, R. A., Pettengill, R. C., Hite, L. R., "A General Purpose 1024-state Electronically Programmable Transversal Filter", IEEE Journal of Solid-State Circuits, Vol. SC-15, No. 6, December 1980.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

Printed by
United States Air Force
Hanscom AFB, Mass. 01731

DATE

ILME