

AD-A031 509

CODEX CORP NEWTON MASS
CODEX SPEECH DIGITIZER ADVANCED DEVELOPMENT MODEL.(U)
JUN 76 G D FORNEY, S QURESHI

F/G 17/2

DCA100-76-C-0026

UNCLASSIFIED

NL

1 OF 2
ADA031509



CODEX

AD A031509

FG
?

DEFENSE COMMUNICATIONS AGENCY

DCA100-76-C-0026

CODEX SPEECH DIGITIZER

ADVANCED DEVELOPMENT MODEL

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

DDC
RECEIVED
NOV 3 1976
D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

codex
corporation

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DCA100-76-C-0026	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CODEX SPEECH DIGITIZER ADVANCED DEVELOPMENT MODEL.	5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT. June 1975 - June 1976.	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) G. D. Forney S. Qureshi	8. CONTRACT OR GRANT NUMBER(s) DCA100-76-C-0026 NEW	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Codex Corporation 15 Riverdale Ave. Newton, Massachusetts 02195 <i>Notation</i>	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS P.E. NO. 33126K Task 15101	
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Washington, DC 20305	12. REPORT DATE June 1976	13. NUMBER OF PAGES 110
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) <i>12171p.</i>	15. SECURITY CLASS. (of this report) Unclassified	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Unlimited Public Distribution		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Speech Coding Digital Voice Terminal Delta Modulation Residual Coding		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the Codex Speech Digitizer Advanced Development Model. The speech coder operates at rates of 16 and 9.6 kbps using three selectable techniques: Adaptive Residual Coding (ARC), Continuously Variable Slope Delta (CVSD) modulation, and Adaptive Delta Modulation (ADM).		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

387 208

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

4B

ACCESSION for

NTIS White

DDC Only Section

UNANNOUNCED

JUSTIFICATION _____

BY _____

DISTRIBUTION/AVAILABILITY CODES _____

Dist. AVAIL. and/or SPECIAL

A		
---	--	--

codex
corporation

FINAL REPORT

CODEX SPEECH DIGITIZER
ADVANCED DEVELOPMENT MODEL

DDC
RECEIVED
NOV 3 1976
RECEIVED
D

Codex Corporation
15 Riverdale Avenue
Newton, Massachusetts 02195

DCA100-76-C-0026

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

JUNE 1976

SECTION 1
INTRODUCTION

The Codex Speech Digitizer advanced development models, supplied under Contract DCA100-76-C-0026, code speech at rates of 16 and 9.6 kbps using three selectable techniques: adaptive residual coding (ARC), digitally implemented continuously variable slope modulation (CVSD), and adaptive delta modulation (ADM).

The design is implemented on two circuit assemblies and is contained within a standard attache case along with attendant power supply and handset. An optional rack-mounting kit is provided.

Contents of this Final Report are summarized:

- Section 1 Introduction
- Section 2 Theory of Operation
(Includes design algorithms)
- Section 3 Hardware Design
(Includes circuit descriptions and programs)
- Section 4 Operation
(For operator/installer)
- Section 5 Acceptance Test Procedure
- Section 6 Parts Listings

TABLE 1-1. INPUT/OUTPUT SPECIFICATIONS

Input Impedance	600 ohms
Maximum Input Level	3V, p-p
Output Impedance	600 ohms
Maximum Output Level	3V, p-p
Frequency Cut-off	3000 Hz, Transmit and Receive

SECTION 2

THEORY OF OPERATION

2.1 GENERAL

This section summarizes the theory of operation and describes the algorithms implemented in the Codex speech digitizer. For a more detailed description of the theoretical background refer to section 2 of our first quarterly report¹ on contract no. DCA 100-74-C-0053.

The Codex speech digitizer is capable of coding speech into a bit stream at 9.6 and 16 kb/s using one of the following three techniques: adaptive residual coding (ARC), continuous variable-slope delta modulation (CVSD) and adaptive delta modulation (ADM).

The adaptive residual coder uses the type of waveform coding generally known as adaptive differential pulse code modulation 2^{-4} (ADPCM). However, it combines non-uniform multilevel adaptive quantization with a two-loop step-size adaptation algorithm and variable-length coding to achieve a subjective quality better than previous schemes at lower transmission rates. All elements of the system are self synchronizing and robust against high channel error rates.

CVSD and ADM are variants of delta modulation, using a two-level (binary) quantizer and a sampling rate equal to the bit rate, with two different algorithms for quantizer step size adaptation.

A functional description of the system and a simple block diagram illustrate the basic operations performed in the speech digitizer.

Section 2.3 deals with the adaptive quantizer, which is the most important system element. The two-loop update procedure used in the ARC mode allows the step size to expand rapidly on the onslaught of voiced sounds and pitch pulses, while minimizing granular noise during stationary intervals.

Variable-length codes for efficient transmission of the quantized residual at 9.6 and 16 kb/s are given in Section 2.4, together with a buffer management scheme to cope with the potential problems that arise in the use of variable-length codes.

The algorithms used for CVSD and ADM are presented in Section 2.5.

2.2 SYSTEM STRUCTURE

The block diagram of the speech digitizer for adaptive residual coding is shown in Figure 2-1. Input speech from a telephone handset or microphone passes through a low-pass filter and is sampled at a rate twice the bandwidth of the filter. Linear predictions p_k based on N previous reconstructed samples s_{k-i}' , $i=1, \dots, N$ are subtracted from the input speech samples s_k to give the residuals e_k . The quantizer outputs are the normalized levels l_k , representing quantized residuals $e_k' = T_k l_k$, where T_k is the quantizer step size. This step size is updated for each sample in the quantizer control circuitry on the basis of the previous quantizer output. The reconstructed speech samples are just the sum of the quantized residuals and the predictions. Finally, quantizer outputs l_k are encoded in a variable-length coder and placed in a first-in first-out (FIFO) buffer, whence they are clocked out onto the line at 9.6 or 16 kb/s. At the receiver, received bits are clocked into a FIFO buffer, whence they are removed and decoded into quantizer levels l_k . Reconstructed samples s_k' are computed from the levels as in the transmitter, and are passed through a D/A and low-pass filter to a telephone handset or speaker.

Since the speech digitization method consists of coding adaptively quantized residuals, it has been called the adaptive residual coder^{5,6} (ARC).

Our earlier work on adaptive predictors^{1,7} showed that while such a predictor would substantially increase the complexity of the ARC, the gain in signal-to-quantization noise ratio it provided was only 1 to 1.5 dB. Moreover, informal listening tests of simulated systems with adaptive and fixed predictors indicated little difference in subjective speech quality.

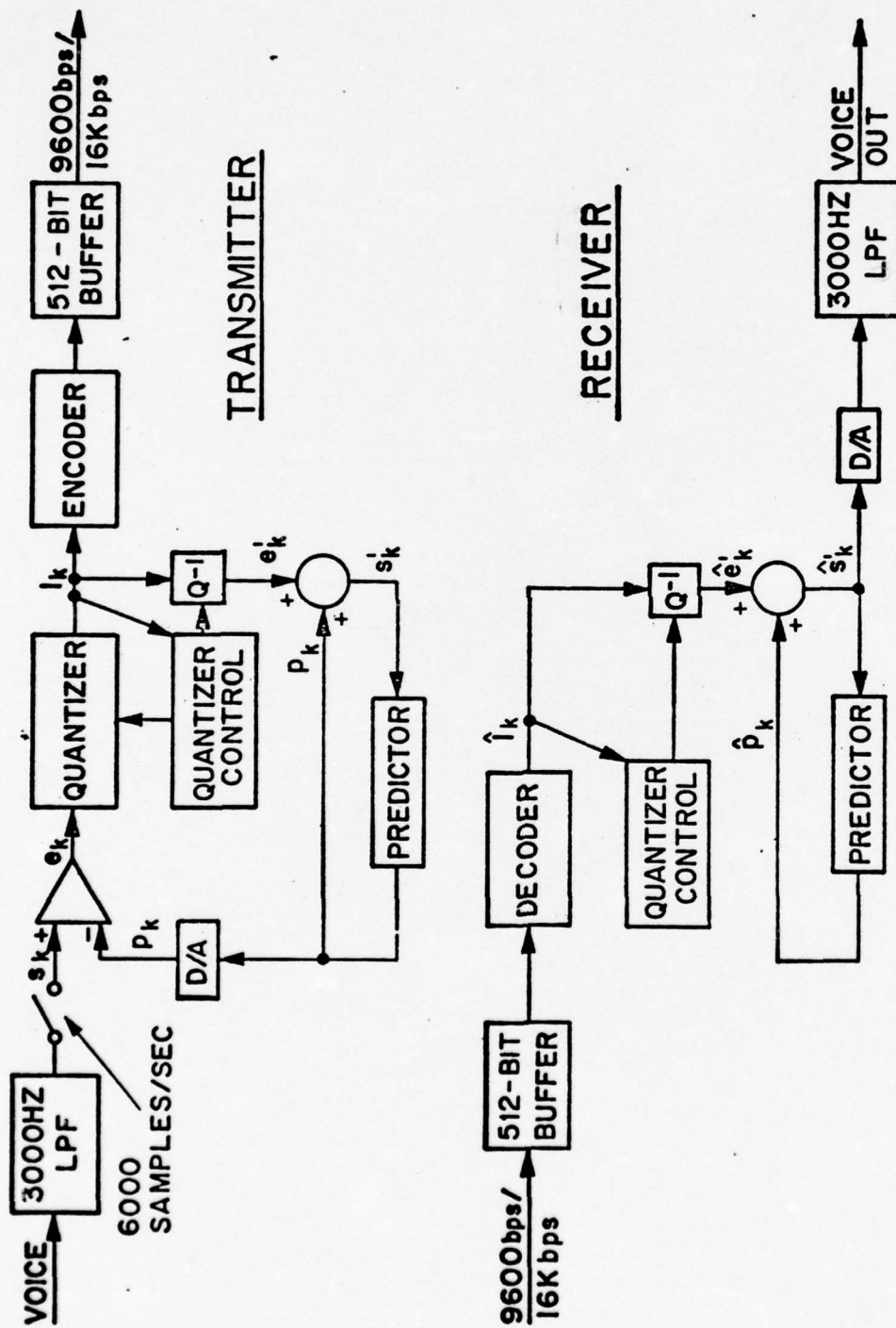


FIGURE 2-1
ADAPTIVE RESIDUAL CODER

Therefore, the following simple third order fixed predictor was implemented in the Codex speech digitizer. The prediction p_k is given by

$$p_k = B_1 s'_{k-1} + B_2 s'_{k-2} + B_3 s'_{k-3}.$$

The particular values of B_1 , B_2 and B_3 are based on the average correlation coefficients of speech at a sampling rate around 6000 samples/sec and have been rounded off for ease of implementation, as explained in section 3.

Since the average correlation coefficients of speech are dominated by the higher-level voiced sounds, the fixed predictor coefficients B_1 , B_2 and B_3 are far from optimum for low-level unvoiced sounds, such as fricatives. In fact, for these sounds the variance of the residual (or error signal e_k) may be higher than the input signal variance resulting in greater quantization noise. We have attempted to overcome this drawback by effectively forcing B_1 , B_2 and B_3 and hence p_k to zero for low-level sounds and background noise indicated by a small quantizer step size.

2.3 ADAPTIVE QUANTIZER

Figure 2-2 illustrates typical parameters for 5- and 7- level quantizers used for 9.6 and 16 kb/s ARC systems, respectively. The slicing levels or thresholds are represented by short vertical lines whereas stars indicate the quantizer output levels. T_k is the current quantizer scale factor, or step size.

The quantizers are nonuniform to take advantage of the amplitude distribution of the residual. We chose a quantizer with an odd number of levels for both rates to ensure that the zero level was included. The presence of the zero level tends to eliminate the idle channel noise characteristic of two-level quantizers used in delta-modulation systems. Another feature of the quantizers of Figure 2-2 is the inclusion of two 'extra' outer levels which are relatively farther removed from the other levels. This was originally proposed by Cohn and Melsa⁶, who have called the technique PCQ (pitch compensated quantizer).

The occurrence of the PCQ levels is used as a cue to overload distortion and we respond by rapidly expanding the quantizer step size T_k . At other times T_k is adapted at a syllabic rate in response to changes in the average signal level, as explained below.

Syllabic adaptation can be achieved^{1,7} by choice of expansion/contraction factors $M(\cdot)$ in the following update algorithm proposed by Jayant⁸:

$$T_{k+1} = T_k M(|\ell_k|).$$

For an ADPCM system Jayant selected the multipliers $M(\cdot)$ corresponding to a small update time constant to allow the quantizer to respond rapidly to changes in the input level. Thus, severe overload distortion is avoided, at the expense of substantial granular noise. On the other hand, if multipliers close to unity are used to achieve syllabic companding, there is a noticeable though somewhat less annoying degradation due to overload distortion. We use a Jayant loop of the latter type, but compensate for this distortion by increasing T_k through a second correction factor which is rapidly increased on the occurrence of a PCQ level and decays to unity with a small time constant. Thus, with this two-loop algorithm we permit the quantizer to track short term increases in the input level while keeping the granular noise during stationary sounds to a minimum.

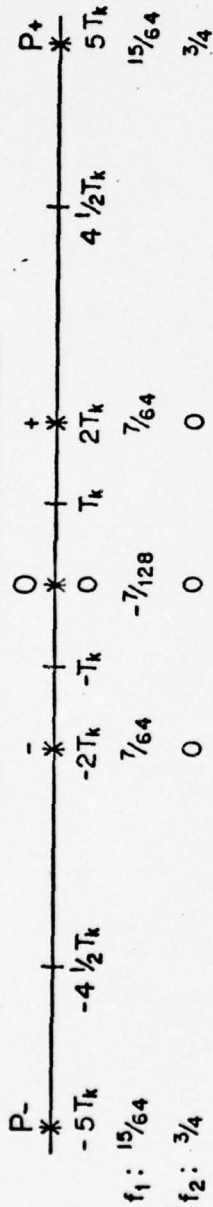
Another important aspect of any adaptive quantizer is the minimum or bottom of the step size. The bottom must be small enough to faithfully reproduce low-level unvoiced sounds but large enough to block out noise during silence intervals. Thus a compromise value must be selected.

We now present the quantizer adaptation algorithm. To avoid multiplications in the algorithm we define a logarithmic step-size parameter

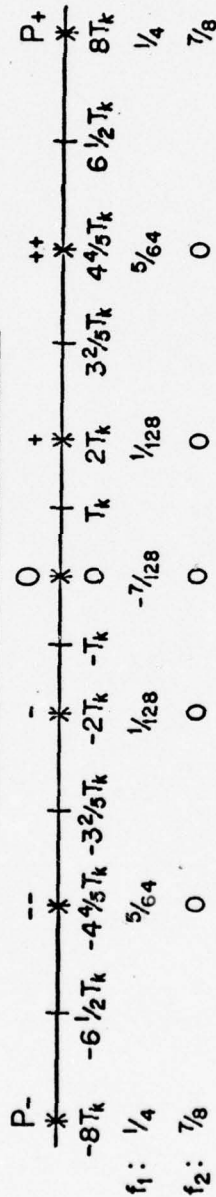
$$G_k = \log_2 T_k$$

and set $G_k = G'_k + C'_k + G_{\min}$.

9600BPS: 5-LEVEL QUANTIZER



16KBPS: 7-LEVEL QUANTIZER



ADAPTATION LOGIC

$T_k = 2^{G_k}$
 $G_k = G_k' + C_k + G_{min}$
 $G_{k+1} = \zeta G_k' + f_1(I_k) \geq 0$
 $C_{k+1} = \gamma C_k + f_2(I_k)$

PARAMETERS

$G_{min} = 1$ (16KBPS), 2 (9.6KBPS)
 $\zeta = 127/128$
 $\gamma = 3/4$

FIGURE 2-2
ADAPTIVE QUANTIZER
USED FOR ADAPTIVE RESIDUAL CODING

The constant G_{\min} determines the quantizer bottom. The term G_k is updated at a syllabic rate according to

$$G_{k+1} = \zeta G_k + f_1(|l_k|) \geq 0$$

where $f_1(\cdot)$ are small as indicated by the typical values shown in Figure 2-2. The correction term C_k is updated according to

$$C_{k+1} = \gamma C_k + f_2(|l_k|)$$

where $f_2(\cdot)$ is zero for all but the outermost levels for which it is relatively large. With $\gamma \approx 3/4$ ($1 - 2^{-m_2}$ in the notation of section 3), C_k decays to zero with a time constant of $2^{m_2} = 4$ samples.

To make the algorithm robust against channel errors ζ is selected as $127/128$ ($1 - 2^{-m_1}$ in the notation of section 3), which ensures that the effect of an incorrect update at the receiver due to a channel error will decay exponentially with a time constant of $1/(1 - \zeta) = 2^{m_1} = 128$ samples.

Further, a small constant T_{\min} is added to T_k to achieve more freedom in selecting the minimum value of the quantizer step size (see section 3.).

2.4 VARIABLE-LENGTH CODES AND BUFFER MANAGEMENT

Having set up an adaptive quantizer, attention can now be focused on coding the quantizer output levels for transmission to the receiver.

Table 2-1 lists the typical probabilities obtained in the adaptive residual coder for the 7- and 5- level quantizers omitting silence intervals. The source entropies, H , for these quantizers are 2.42 and 1.53 bits per sample (see Table 2-1). Thus, transmission rates of 16 kb/s and 9.6 kb/s can be achieved by properly designed variable-length codes at sampling rates of 6400 and 6000 samples/s, respectively.

Variable-length codes imply the use of a buffer, which necessitates a buffer management scheme to handle initial synchronization, underflow and overflow. To

this end we have chosen self-synchronizing codes; that is, the receiver is able to properly partition the received bit sequence into code words, both initially and after channel errors. Moreover, we provide a filler word in the codes to prevent transmit buffer underflow.

At 16 kb/s, prefix Huffman codes turn out to be reasonably efficient. The one given in Table 2-1 (average code word length $\bar{l} = 2.47$) was selected for its good synchronization properties. The code is complete, so the decoder can parse any bit stream into code words. The synchronization sequence is 010; that is, the receiver can start parsing at any point and as soon as 010 occurs it will be in the same state as the transmitter, namely at the end of the word. Since the code for the '0' level is 10, the synch sequence appears frequently.

At 9.6 kb/s the PCQ levels P_+ and P_- which have small probabilities of occurrence, make efficient encoding difficult. A normal prefix Huffman code leads to an average codeword length about 0.12 bits greater than the source entropy, making it impossible to achieve a transmission rate of 9.6 kb/s at 6000 samples/s. Consequently the variable-input-variable-output code shown as the two codes 1 and 2 in Table 2-1 has been devised. Code 1 is used at all times, except that after a '-' level has been sent in code 1, code 2 is used on the following time; in other words a 0 is inserted after every pair of '-' levels. Since 0 occurs only at the end of code words, the receiver automatically synchronizes itself every time a 0 is received. The code is uniquely decipherable, and decoding is simply a matter of counting the number of ones before the 0 that terminates every codeword except the filler. Encoding is also straightforward.

The buffer management scheme is simply the following. At the transmitter the filler word is inserted whenever the buffer is empty. If the buffer fills up, the drastic step of transmitting the code word corresponding to the zero level at 9.6 kb/s is taken. At 16 kb/s the quantizer need only be restricted to the three innermost levels, since the length of the code words corresponding to these levels is

TABLE 2-1

VARIABLE LENGTH CODES

LEVEL	16KBPS			9600 BPS		
	PROB	CODE	PROB	CODE H	CODE 1	CODE 2
P+	0.03	01110	0.02	1110	111110	111110
++	0.10	0110	-	-	-	-
+	0.24	00	0.19	10	10	10
0	0.28	10	0.60	0	0	0
-	0.23	11	0.18	110	11	110
--	0.10	010	-	-	-	-
P-	0.02	011110	0.01	11110	1111110	1111110
FILLER	-	011111	-	11111	11111111	-
	H=2.42	$\bar{l}=2.47$	H=1.53	$\bar{l}=1.65$	$\bar{l}=1.56$	
		PREFIX HUFFMAN SELF-SYN (SS=010)		PREFIX HUFFMAN SELF-SYN (SS=0)		PREFIX VARIABLE-TO-VARIABLE SELF-SYN (SS=0)

BUFFER MANAGEMENT

- TRANSMIT FILLER WORD WHEN TRANSMIT BUFFER EMPTY
- TRANSMIT 0 WORD WHEN TRANSMIT BUFFER FULL
- ENSURE RECEIVE BUFFER FULL WHEN FILLER WORD RECEIVED (IF NOT, SUPPRESS BUFFER OUTPUT AND SUBSTITUTE 0 WORDS UNTIL FULL)

less than the channel transmission rate of $2 \frac{1}{2}$ or $2 \frac{2}{3}$ bits per sample

At the receiver, initial synchronization is achieved by ensuring that the buffer is 'full' when the filler word is received. If the buffer is not 'full' and a filler is received, the regular decoding cycle (buffer unloading) is suppressed and the zero quantizer level is substituted. This tends to fill the receiver buffer to the proper depth at startup, and is the only synchronization method provided. Subsequently, receive buffer overflows or underflows should not occur in the absence of channel errors. If the receive buffer underflows after channel errors the quantizer level is forced to zero. In the event of receive buffer overflow, no special action is taken and the affected samples are simply deleted or garbled.

Thus, with very little sacrifice in code efficiency (due to the addition of the filler word) this simple buffer management procedure avoids any special initial synchronization sequence or handshaking.

2.5 DELTA MODULATION

The block diagram for delta modulation is similar to the ARC system diagram shown in Fig. 2-1. The main differences are that the sampling rate is now 9.6 or 16 kb/s depending on the selected bit rate, the quantizer is binary (two-level) and, therefore, the encoder and decoder are no longer required. Moreover, a first-order predictor (single integrator in analog implementations) is usually employed.

In the following sections we present the algorithms for CVSD and ADM.

2.5.1 Continuous Variable-Slope Delta Modulation (CVSD)

CVSD has evolved from the work of Greefkes and DeJager⁹ and others. The quantizer step size in CVSD is adapted smoothly in time with a time constant of the order of 5 ms resulting in robustness against channel errors.

Conventional analog implementations of CVSD suffer from the drawback of sensitivity to component tolerances and DC offsets leading to the selection of a

smaller than desirable value for the ratio of the maximum to minimum step size. The Codex speech digitizer provides a digital implementation of CVSD with an improved choice of parameters.

As in ARC, the prediction p_k , based on the previous reconstructed speech sample s'_{k-1} , is subtracted from the input speech sample to find the error signal e_k . Thus,

$$e_k = s_k - p_k,$$

where

$$p_k = B_1 s'_{k-1}$$

with B_1 a constant of the form $1-2^{-n}$ close to 1. Thus, the time constant of integration is 2^n sample intervals. A single bit, $b_k=0$ or 1, based on the sign of the residual e_k is transmitted. The reconstructed speech sample s'_k is simply the sum of the quantized residual e'_k and the prediction p_k , that is

$$s'_k = e'_k + p_k$$

with

$$e'_k = \text{sgn}(e_k) (\ell T_k + T_{\min})$$

where the constant ℓ is the normalized quantizer output level and T_k is the quantizer step size at time k and T_{\min} is a constant.

A run of three bits is used as a criterion for the detection of slope overload leading to the following step size update strategy:

$$T_{k+1} = \zeta T_k + f(b_k, b_{k-1}, b_{k-2})$$

where ζ is again a constant of the form $1-2^{-m1}$ and $f(b_k, b_{k-1}, b_{k-2})$ is nonzero only when $b_k = b_{k-1} = b_{k-2}$. Clearly, the time constant of adaptation is $2^{m1} K$ where

$$f(b_k, b_{k-1}, b_{k-2}) = \begin{cases} K & , \text{ if } b_k = b_{k-1} = b_{k-2} \\ 0 & , \text{ otherwise.} \end{cases}$$

2.5.2 Adaptive Delta Modulation (ADM)

The price paid for robustness against channel errors in the CVSD quantizer adaptation algorithm is the significant amount of slope-overload distortion present in the reconstructed speech. This leads to a certain loss of "crispiness" as well as considerable degradation in speech quality when two or more CVSD links are connected in tandem.

In this section we present an improved algorithm for quantizer step size adaptation which results in better subjective speech quality without a significant increase in sensitivity to channel errors.

A desirable approach in updating the quantizer step size is to attempt to set the step size proportional to a moving estimate of the standard deviation of the quantizer input. We have previously shown^{1,7} that Jayant's update algorithm

$$T_{k+1} = T_k M(|e_k|)$$

approximates an estimator of the above form. The algorithm is quite general in the sense that the speed of adaptation and the steady-state fluctuation of T_k for stationary portions of the signal is determined by the function $M(\cdot)$.

In the case of a two-level quantizer it is not possible to estimate the standard deviation of the input based on only one quantizer output. Thus, for delta modulation the algorithm needs to be modified so that

$$T_{k+1} = T_k M(b_k, b_{k-1}, \dots)$$

i.e. $M(\cdot)$ becomes a function of two or more quantizer outputs. There is, of course, some delay in adjusting T_k , e.g., severe overload is detected when there have been three or more successive 1's or 0's at the quantizer output. This is compensated to some extent by the higher sampling rate of delta modulation systems.

Jayant has suggested⁴ a value of $M = 1.5$ when $b_k = b_{k-1}$ and $M = 1/1.5$ when $b_k \neq b_{k-1}$. This rule leads to small overload distortion at the expense of greater granular noise and increased sensitivity to channel errors.

We have implemented the following update algorithm which is similar to the one used for ARC:

$$T_k = \log_2 G_k$$

$$G_k = G'_k + G_{\min}$$

where $G'_{k+1} = \zeta G'_k + f(b_k, b_{k-1}, b_{k-2})$

with the function $f(\cdot)$ of the following form:

$b_k \oplus b_{k-1}, b_k \oplus b_{k-2}$	$f(b_k, b_{k-1}, b_{k-2})$
0 0	3/16 (expand rapidly)
0 1	5/64 (expand)
1 0	-5/128 (contract)
1 1	0

Note that the quantizer step size is expanded on the first indication of overload i.e., $b_k = b_{k-1}$. If overload persists, i.e., $b_k = b_{k-1} = b_{k-2}$, the step size is expanded rapidly. To make the algorithm resistant to channel errors we have incorporated the factor $\zeta = 1 - 2^{-m1} = 127/128$ which ensures convergence of the quantizer step size with a time constant of $2^{m1} = 128$ sample intervals.

Informal listening tests indicate that ADM results in 'crisper' reconstructed speech with less granular noise than CVSD. The main improvement in quality is due to the fact that the step size update is multiplicative (additions in the logarithmic domain in ADM) rather than additive (RC filtering in CVSD).

To take advantage of the additional processing time available at 9.6 kb/s a third-order predictor is used instead of a first-order predictor for adaptive delta modulation (see microprogram given in Table 3-7).

2.6 PARAMETER SELECTION

2.6.1 16 Kb/s CVSD Parameters

1. Time constant of integrator = 8 ms.
2. Time constant of step size filter = 2 ms.
3. Minimum step size = 20 mV.
4. Compression ratio = 166.
5. Effective magnitude of pulse applied to step size filter whenever three successive transmission bits are identical = 280 mV.

Starting from the initial recommended parameter values, the following adjustments were made. Informal listening tests indicated that integrator time constant could be reduced from 16 to 8 ms without any noticeable loss in speech quality. This reduction, however, makes the system more robust in the presence of channel errors. The time constant of the step size filter was chosen to be 2 ms to permit a reasonable value for the effective gain of this filter without an arithmetic overflow problem. This in combination with a compression ratio of 166 achieved by keeping the minimum step size as small as 20 mV results in "livelier" reconstructed speech with very small idle channel noise.

REFERENCES

1. "9.6/16 kb/s adaptive gradient speech signal coder," First Quarterly Report, Codex Corporation, Newton, Massachusetts, Contr. No. DCA 100-74-C-0053, October 1974.
2. P. Cumiskey, N. S. Jayant and J. L. Flanagan, "Adaptive quantization in differential PCM coding of speech," Bell System Technical Journal, volume 52, pp. 1105-1118, September 1973.
3. P. Noll, "Non-adaptive and adaptive DPCM of speech signals," Overdruk uit Polytech. Tijdschr. Ed. Elektrotech./Electron. (The Netherlands), no. 19, 1972
4. N. S. Jayant, "Digital coding of speech waveforms: PCM, DPCM, and DM quantizers," Proc. IEEE, vol. 62, pp. 611-632, May 1974.
5. J. D. Gibson, S. K. Jones and J. L. Melsa, "Sequentially adaptive prediction and coding of speech signals," IEEE Trans. Commun., vol. COM-22, pp. 1789-1797, November 1974.
6. D. L. Cohn and J. L. Melsa, "The residual encoder - an improved ADPCM system for speech digitization," Conf. Rec. Int'l Conf. on Commun. Vol. II, pp. 30-26 to 30-30, June 1975.
7. S. U. H. Qureshi & G. D. Forney, Jr., "A 9.6/16 kb/s speech digitizer," Conf. Rec. International Conf. on Commun. Vol. II, pp. 30-31 to 30-36, June 1975.
8. N. S. Jayant, "Adaptive quantization with a one-word memory," Bell System Tech. J. vol. 52, pp. 1119-1144, Sept. 1973.
9. J. A. Greefkes and F. DeJager, "Continuous delta modulator for telephone transmission," Philips Res. Rep., vol. 23/2, pp. 233-246, 1968.

SECTION 3

HARDWARE DESIGN

3.1 GENERAL

The CODEX Speech Digitizer is capable of coding speech at 16 and 9.6 kb/s using any of the following three techniques: adaptive residual coding (ARC), digitally implemented continuous variable-slope delta modulation (CVSD) and adaptive delta modulation (ADM). In addition, a different set of ARC parameters may be stored for operation at another desired rate (such as 14.4 or 19.2 kb/s) using a modem-supplied bit-rate clock.

In this section we describe the essential features of the hardware design of various system components and include detailed schematics for the two circuit cards.

As shown in Table 3-1 the mode of operation is controlled by the signals M0, M1 and M2 (see also Fig. 3-9). The first three rocker dip switches mounted in position A6 on the digital card control M0, M1 and M2, respectively.

The nominal bandwidth of the coder is 3000 Hz in all modes (2820 Hz 3 dB bandwidth). The sampling rate is 6000 samples/s except for CVSD and ADM (M1 = 1) for which the sampling rate is equal to the bit rate.

3.2 BLOCK DIAGRAM

In section 2.2 we presented a simple block diagram which shows the interconnection of the major functional components of the system. Figure 3-1 illustrates the system hardware architecture for a transmit-receive pair. A microprocessor-like structure is apparent in the top half of Figure 3-1. This 12-bit arithmetic processor, which is built mostly out of series 7400 low-power Schottky TTL circuits, handles all the basic arithmetic operations required in adaptive residual coding, CVSD and adaptive delta modulation (see Figure 2-1). These operations include the formation of the second-order prediction p_k , the quantized residual e_k^q , the

TABLE 3-1
MODES OF OPERATION

M0	M1	M2	
0	0	0	16 kb/s ARC
0	0	1	14.4 kb/s ARC
0	1	0	16 kb/s CVSD
0	1	1	16 kb/s ADM
1	0	0	9.6 kb/s ARC
1	0	1	Unspecified
1	1	0	9.6 kb/s CVSD
1	1	1	9.6 kb/s ADM

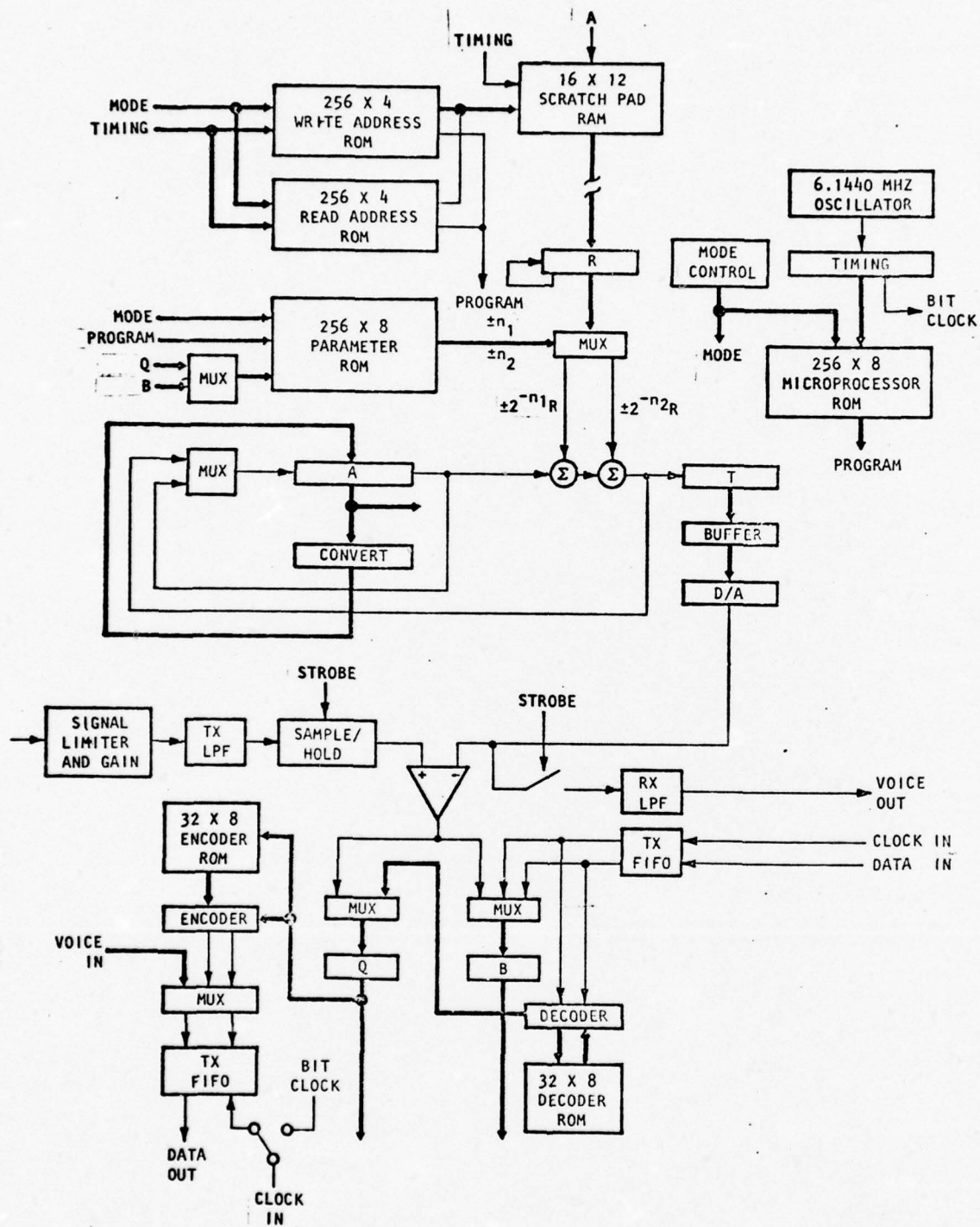


Figure 3-1. Block Diagram

reconstructed speech samples s_k' and the update of the quantizer step size. The operation and microprogramming of the processor are explained in Section 3.4.

The analog circuits shown in Figure 3-1 (the signal limiter, the transmitter low-pass filter (LPF), the sample-and-hold circuit, the comparator, the D/A converter and the receiver LPF) are discussed in Section 3.6.

As explained in the next section, the timing circuits are designed to divide each sample interval into two halves: The transmit and receive cycles. This arrangement permits time-sharing of the processor, the D/A converter and the quantizer register Q or bit register B between the transmitter and the receiver.

In the ARC mode, during the transmit cycle, the prediction p_k is computed and loaded into the test register T whose output is D/A converted and compared with the input speech sample. After a series of (successive-approximation) comparisons with the quantizer level corresponding to the quantized residual e_k' appears in the Q register. This quantizer level is encoded and loaded into the transmit first-in first-out (FIFO) buffer, whose output is the transmit data. Receive data are loaded into the receive FIFO and eventually decoded, with the decoded level being placed in the Q register. The output of the Q register is also used to address the read-only memory (ROM) where the quantizer parameters are stored. These parameters are used by the processor to update the transmit and receive quantizer step sizes during the corresponding cycles. The operation of the encoder, decoder and the FIFO's is explained in Section 3.5.

When the speech coder is in the CVSD or ADM mode encoding and decoding are no longer required. During the transmit cycle, the binary quantized residual (output bit) which is available after a single comparison, is serially loaded into a B register. The output bit is then fed into the transmit FIFO which in this case serves only to decouple the speech coder timing from the modem clock. Similarly, the received data are loaded into the receive FIFO and eventually loaded into the B register, which stores the present and previous two output and input bits in inter-

leaved form. These three (output/input) bits are used to address the parameter ROM.

3.3 TIMING AND CLOCK RATES

The transmitter and receiver sections are both run from a common high-frequency clock (YCL) derived from a 6.1440 MHz crystal-controlled oscillator to permit time sharing the circuitry mentioned above. The first-in first-out (FIFO) buffers allow decoupling of system timing from modem timing.

As shown in Fig. 3-9, the main divide-down chain consists of three synchronous 4-bit binary counters (3X74LS163). The first of these counters (A3) simply divides the \overline{YCL} clock into groups (words) of 16 pulses (bits) each and produces a carry (WDTIME) during the last of these 16 \overline{YCL} clock periods (see the timing diagram Fig. 3-2). The nomenclature 'words' and 'bits' will become clear in the next section. The arithmetic processor completes one operation (instruction) in one word-time as bits of information are processed and serially loaded into the A or T register at each bit-time. The second counter (A2) produces a carry (GPTIME) every 16th word time in the ARC mode ($M1 = 0$) independent of the bit rate. However, in the CVSD or ADM mode ($M1 = 1$) the counter acts as a divide by 6 at 16 kb/s ($M0 = 0$) or as a divide by 10 at 9.6 kb/s ($M1 = 1$). The third counter (A1) produces a SAMPLETIME pulse for every fourth GPTIME pulse. We can summarize the breakdown of the sample interval as follows:

1 sample interval = 4 group times

$$1 \text{ group time} = \begin{cases} 16 \text{ word times (ARC)} \\ 10 \text{ word times (CVSD/ADM at 9.6 kb/s)} \\ 6 \text{ word times (CVSD/ADM at 16 kb/s)} \end{cases}$$

1 word time = 16 bit times (6.144 MHz clock periods)

The timing signal W6 (a square wave at the sample rate) divides the sample interval into the transmit ($W6 = 0$) and receive ($W6 = 1$) halves. The word times during each of these halves is indexed by the 5-bit word $W5, W4, \dots, W1$. In the

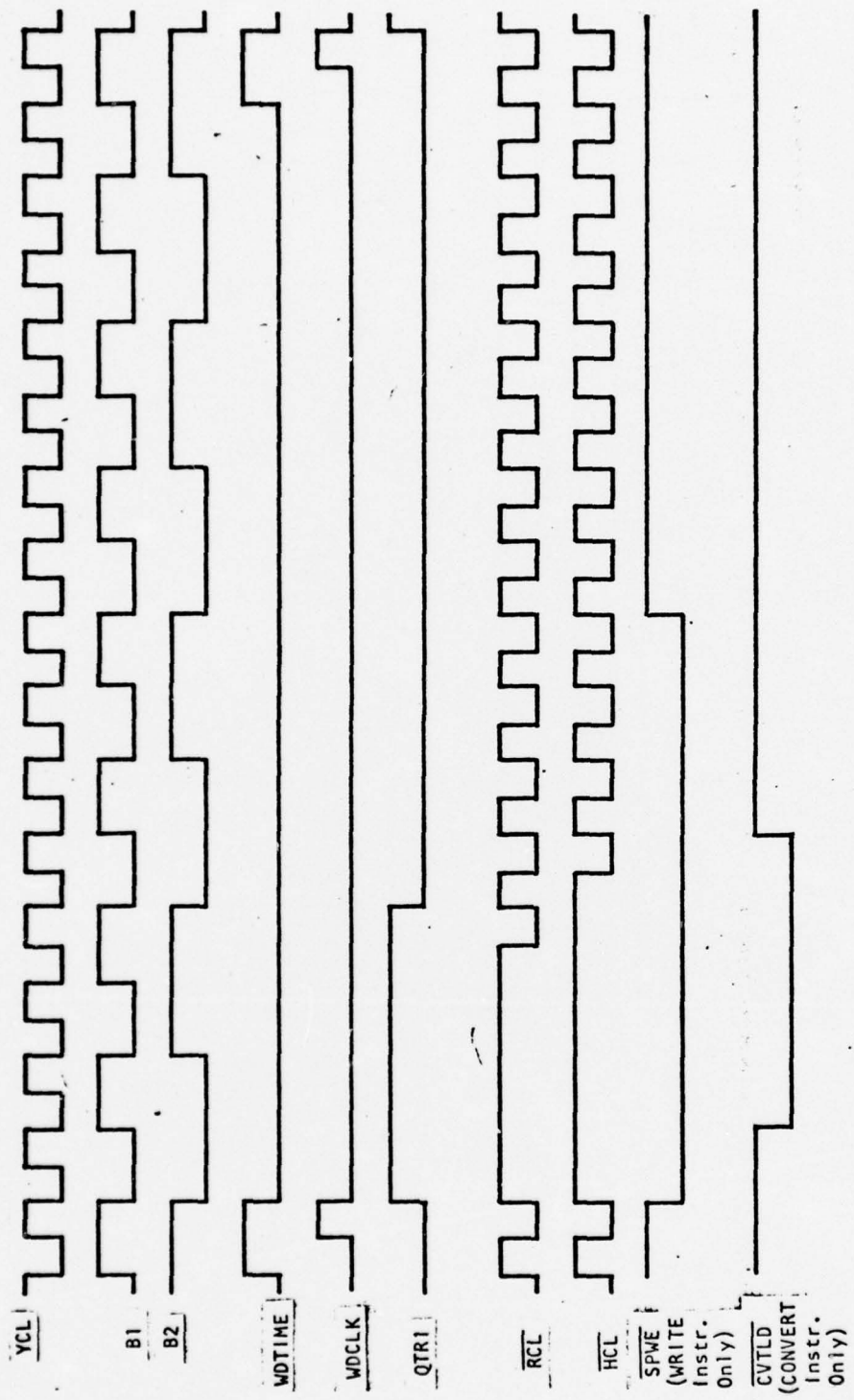


Figure 3-2. Timing Diagram

ARC mode the count is simply 0 to 31, while in the CVSD/ADM mode the count proceeds as follows:

CVSD/ADM at 16 kb/s 10, ..., 15, 26, ..., 31

CVSD/ADM at 9.6 kb/s 6, ..., 15, 22, ..., 31

As explained in the next section, W5, ..., W1 are used as the program counter for the microprogram.

The timing circuits also generate a clock BITCLKX2 at twice the 16 or 9.6 kb/s bit rate. As shown in Figure 3-9 BITCLKX2 is W1 divided by 6 when M0 = 0 and is W2 divided by 10 when M0 = 1.

The 4-bit shift register 74LS194 (A9) generates the 25% duty cycle clocks QTR1 to QTR4, which as the names imply, are high during the respective quarters of a word time. Other timing signals shown in Figures 3-2 and 3-3 are \overline{RCL} (\overline{YCL} with the first 3 clock pulses suppressed), \overline{HCL} (\overline{YCL} with the first 4 clock pulses suppressed) and WDCLK (\overline{YCL} with only the last clock pulse enabled each word time), \overline{SPWE} (scratchpad write enable) and \overline{CVTLD} (convert load).

As explained in Section 4.2, transmit output timing is selectable (via dip switch C21 on the digital card) from internal or external (modem supplied) bit rate clock (DB in EIA notation). The former is either 9.6 or 16 kb/s, while the latter may be arbitrary in the ARC mode provided that the system parameters are chosen so that the transmit buffer does not overflow at that output rate. Similarly the receive clock and data may be at an arbitrary rate. Thus in the ARC mode any bit rate can be used by providing the appropriate external clock. Note that a tracker circuit for slaving the internal timing to an external bit rate clock (transmit or receive) is not required since the buffer management strategy is not sensitive to timing slips. Thus, timing slips affect the system performance in the same way as channel errors.

3.4 ARITHMETIC PROCESSOR AND QUANTIZER

The microprogrammed processor executes a set of 12, 20, or 32 instructions twice per sample interval to perform the transmit and receive arithmetic functions.

To avoid full 12-bit multiplications in the processor, we use the quantizer adaptation rule given in Section 2.3. In ARC and ADM modes, the step size T_k is obtained through the following piecewise linear approximation of 2^{G_k} :

$$T_k = \left[2^{\lfloor G_k \rfloor} (1 + G_k - \lfloor G_k \rfloor) \right]$$

where $\lfloor \cdot \rfloor$ means "integer part of". This operation is performed by the CONVERT circuitry shown in Figure 3-1. Further, we constrain all multiplicative parameters to be of the form $\pm 2^{-n_1} \pm 2^{-n_2}$. Then all "multiplications" can be performed by two adders (see Figure 3-1).

The flow of information and the processor function is controlled by the contents of the microprogram ROM and the write-and read-address ROM's.

A 256x8 bit programmable read-only memory (74S471) controls most of the processor functions. Two 256x4 bit programmable ROM's (2x74S287) whose outputs are tied together to form a tristate bus provide part of the address for the scratchpad (see Figure 3-1) and perform the remaining control functions. The main storage elements of the processor are a 16x12 bit scratchpad RAM (3x74C89), a 256x8 bit parameter ROM (74S471), and three 12-bit registers: a RAM output (or read) register R (3x74LS194), an accumulator register A (3x74LS194), and a test register T (74LS164 + 74LS195). The basic arithmetic operation performed by the processor is of the form $A \pm 2^{-n_1} R \pm 2^{-n_2} R$. The result of this operation can be serially loaded in A or T or both. The 'multiplier' $\pm 2^{-n_1} \pm 2^{-n_2}$ is supplied by the parameter ROM.

The contents of the accumulator A can be stored (written into) the scratchpad RAM and later loaded into the R register, which performs a right shift with sign extension. Two one-of-eight multiplexers (2x74LS151) and a pair of true-complement (exclusive or) gates (1/2x74LS86) provide the shifted and/or negated serial words

$\pm 2^{-n1} R$ and $\pm 2^{-n2} R$. These words are serially added to A in a pair of high-speed one-bit full adders (74H183). The accumulator A can be cleared, serially loaded from the adder output or its contents can be recirculated. During the CONVERT operation the A register is parallel loaded with the converted step size T_k .

At the appropriate instants the contents of the test register T are loaded into the D/A buffer register (2x74LS174 located on the analog card) for quantizer comparisons during the transmit cycle and for reconstructed speech sample output during the receive cycle.

During the transmit cycle in the ARC mode the quantizer output level $\pm i$, $i=0,1,2,3$ is determined by making three tests to locate the residual e_k in one of seven regions symmetrical about zero. During these tests the T register is loaded with p_k , $p_k \pm a_2 T_k$ and $p_k \pm a_1 T_k$, $i=1$ or 3 , successively where a_i are the quantizer thresholds and T_k is the quantizer scale factor (step size) at time k . After D/A conversion these test values are compared to the input speech sample s_k and the one-bit result of each comparison is loaded into the quantizer register Q (74LS163) after some logic. The final quantizer level eventually appears in the Q register, where it controls the variable-length encoder.

During the receive half of the sample interval the Q register is loaded with the quantizer level directly from the decoder.

As explained earlier in Section 3.2, in ADM and CVSD modes a separate B register (74LS164) stores the present and previous two bits in interleaved form. The B register is serially loaded under program control once during the transmit cycle (from the comparator) and once during the receive cycle (from the receive FIFO).

Two output bits of the Q or B register address the parameter ROM together with mode control and timing information to select the appropriate "multiplier" $\pm 2^{-n1} \pm 2^{-n2}$. The sign bit Q_s or B_s determines whether the "product" $\pm 2^{-n1} R \pm 2^{-n2} R$ is to be added to or subtracted from A.

For the 9.6 kb/s ARC system the number of quantizer levels is reduced to 5 by simply setting $a_2=a_3$, $l(2)=l(3)$, $f_1(2)=f_1(3)$ and $f_2(2)=f_2(3)$ in the parameter ROM.

Having described the capabilities of the components of the processor, we are now in a position to discuss the processor control through the microprogram and give a detailed description of the circuits.

3.4.1 MICROPROGRAM

The microprogram may consist of up to 32 12-bit instructions (control words) for each mode of operation. There are no branch instructions; the program counter is simply the 5-bit word count W_5, \dots, W_1 , generated by the timing circuits. As mentioned in Section 3.3 the count W_5, \dots, W_1 , takes only 12 of the 32 possible values in the CVSD/ADM mode at 16 kb/s and 20 values at 9.6 kb/s. The microprogram is identical for the transmit and receive cycles. However, the timing signal, W_6 , is used to select slightly different behavior, e.g., to select one half of the scratchpad RAM for the transmit variables and the other half for the receive variables, and to select different sources for loading the Q or B registers.

From the preceding functional description we can identify three basic types of instructions: ADD, WRITE and CONVERT. In the ADD instruction, the R, A and T register are shifted right and a rounded 12-bit serial add is performed (through the two adders) during the last 13 bit times of the particular word time. The sum is always serially loaded into the T register and it may also be loaded into the A register. If the A register is not to be loaded from the adder output, its contents are simply recirculated. Table 3-2 shows the specification of the ADD instruction in terms of the microprogram word (control signals P1 through P12).

In the WRITE instruction, the A and R registers are not serially shifted. The contents of the A register are written into the scratchpad RAM during the first 4 bit times. Later, at the last bit time the A register may be cleared. Since the

scratchpad writing is completed during the first half of the particular word time, the contents of the same or another scratchpad location may be loaded into the R register at the last bit time of the same word. Under program control the A register contents may be cleared before writing into the scratchpad, if the most significant bit (MSB) of the T register is 1. The specification of the WRITE instruction appears in Table 3-3.

The CONVERT instruction is a special instruction that replaces the contents of the A register by $2^{\lfloor A \rfloor + k} (1 + A - \lfloor A \rfloor)$ to obtain approximate binary exponentiation as explained earlier. The constant k corresponds to G_{\min} (see Section 2.3) which determines the bottom (minimum value) of the quantizer step size. The CONVERT instruction is specified by P1=P2=1 in the microprogram word. P3 to P8 are the same as in the WRITE instruction and the four bits P9 to P12 specify the constant k mentioned above (see Table 3-4).

In the ADD instructions the R register is shifted right with sign extension (arithmetic right shift) during the last 13 bit times of each word. However, at the last bit time the right shift may be suppressed and instead the register may be parallel loaded from the scratchpad RAM. In the WRITE and CONVERT instructions shifting of the R register is inhibited but parallel loading is permitted.

The quantizer register Q may be loaded during any instruction at the last bit time. During the WRITE instruction the Q register input may be held to a zero under program control depending on the state of the transmit FIFO (explained further in Section 3.4.6).

To refresh the value '1' (=000100000000) in the scratchpad we may clear A and then write its contents into the scratchpad with the 4th MSB inverted.

Note that with P1 through P12 all zero a null instruction (NOP) is executed during which the contents of the A register are recirculated, the adder output is loaded into the T register and the R register is shifted right.

TABLE 3-2
ADD INSTRUCTION

- P1 : = 0.
- P2 : Selects input to A register from the adder output (P2=1) or from A register output (P2=0).
- P3 : Addresses the parameter ROM: parameters are independent of the quantizer level if P3=1.
- P4 : Forms part of the address of the parameter ROM if P3=1.
- P5 : The parameters are multiplied by +1 or -1 depending on the sign bit QS/BS if P5=1.
- P6 : The D/A buffer register is loaded at the last bit time if P6=1.
- P7,P8 : Form part of the address of the parameter ROM.
- P9 : The R register is loaded at the last bit time if P9=1 (R register shifts right at other bit times).
- P10-P12: Form part of the address for reading from the scratchpad RAM if P9=1. Initiate encoding/decoding and strobe (D/A output into receive low-pass filter) if P10=1 and P9=0.

TABLE 3-3
WRITE INSTRUCTION

- P1 : =1
- P2 : =0
- P3 : Clear A register at the last bit time if P3=1.
- P4 : Enter new value into Q register or B register if P4=1.
- P5 : Clear A register during bit times 3 and 4 if the MSB of T register is 1 and P5=1.
- P6 : Hold the Q register input to 0 (low) if P6=1 and the transmit FIFO is full.
- P7,P8 : Unused (don't care).
- P9 : During the first 8 bit times:
Invert the 4th MSB of A register before writing into scratchpad if P9=1.

During the last 8 bit times:
Load R register at the last bit time if P9=1. (R register will be stable at other bit times)
- P10-P12: During the first 8 bit times:
Form part of the address for writing into the scratchpad RAM.

During the last 8 bit times:
Form part of the address for reading from the scratchpad RAM.

TABLE 3-4
CONVERT INSTRUCTION

P1 : = 1

P2 : = 1

P3-P8 : Same as in WRITE instruction (Table 3-3).

P9-P12: Specify the constant k which determines the minimum quantizer step size.

Table 3-5 specifies the addresses of the state variables stored in the scratchpad in each mode of operation. Note that W6 is the most significant address bit permitting a different set of variables to be stored for the transmitter and receiver.

Table 3-6 specifies the addresses for the quantizer and predictor parameters. The first 6 bits of the 8-bit address of the parameter ROM are the three mode control bits and program bits P3, P7 and P8. The last two bits are given by the contents of the Q or B register or by program bit P4 as explained in Section 3.4.2.

The listings of the microprogram for each mode of operation appear in Tables 3-7, 3-8 and 3-9. These listings specify the contents of the program, write- and read-address ROM's, respectively, and explain the functions performed by each instruction. The source programs were written in a form suitable for the Motorola M6800 assembler and stored on disk in the Codex Prime Time Shared Computer System. Thus, binary paper tapes can be automatically generated for programming the programmable read-only memories and changes to the microprogram can be conveniently made.

The first two columns of each table give the ROM address and contents, respectively, in hexadecimal notation. The fifth column specifies the program word in binary format with bits P1 through P8 in Table 3-7 and bits P9 through P12 in Tables 3-8 and 3-9.

TABLE 3-5
SCRATCHPAD RAM ADDRESS TABLE

$W_6, P_{10}, P_{11}, P_{12}$	ARC	CVSD	ADM
X000	'1'	'1'	'1'
X001	s'_{k-1}	s'_{k-1}	s'_{k-1}
X010	T_k	T_k	T_k
X011	s'_{k-3}		s'_{k-3}
X100	G'_k		G'_k
X101	s'_{k-2}		s'_{k-2}
X110	C_k		
X111	spare	spare	spare

TABLE 3-6
PARAMETER ROM ADDRESS TABLE

M0, M1, M2, P3, P7, P8, G, H	ARC	CVSD	ADM
XXX 000 XX	$a_{(i)}$		
001 XX	$l_{(i)}$	l	l
010 XX	$f_1(Q)$	$f(B)$	$f(B)$
011 XX	$f_2(Q)$		
100 00	B_2		B_2
100 10	$(1-2^{-m_2})$	$(1-2^{-m_2})$	
101 00	B_1		B_1
101 10	1	1	1
110 00	B_3		B_3
110 10		B_1	
111 00	T_{min}		
111 10	$(1-2^{-m_1})$	T_{min}	$(1-2^{-m_1})$

TABLE 3-7. MICROPROGRAM

(0001)	*	MICROPROGRAM	
(0002)	*	THIS LISTING SPECIFIES THE FIRST 8 BITS (P1-P8) OF THE PROGRAM	
(0003)	*	WORD FOR THE CODEX SPEECH DIGITIZER	
(0004)	*		
(0005)	*		
(0006)	*	LIST OF SYMBOLS	
(0007)	*		
(0008)	*	A	CONTENTS OF THE A (ACCUMULATOR) REGISTER
(0009)	*	R	CONTENTS OF THE R (READ) REGISTER
(0010)	*	T	CONTENTS OF THE T (TEST) REGISTER
(0011)	*	Q	QUANTIZER REGISTER USED IN THE ARC MODE
(0012)	*	B	BIT REGISTER USED IN THE CVSD AND ADM MODES
(0013)	*	S*(K)	CURRENT RECONSTRUCTED SPEECH SAMPLE
(0014)	*	P(K)	CURRENT VALUE OF THE PREDICTION
(0015)	*	S*(K-1)	PREVIOUS RECONSTRUCTED SPEECH SAMPLE
(0016)	*	S*(K-2)	SECOND PREVIOUS RECONSTRUCTED SPEECH SAMPLE
(0017)	*	S*(K-3)	THIRD PREVIOUS RECONSTRUCTED SPEECH SAMPLE
(0018)	*	B1,B2,B3	PREDICTOR COEFFICIENTS
(0019)	*	A(I)	QUANTIZER THRESHOLDS IN THE ARC MODE
(0020)	*	L(I)	QUANTIZER OUTPUT LEVELS IN THE ARC MODE
(0021)	*	L	QUANTIZER OUTPUT LEVEL IN CVSD AND ADM MODES
(0022)	*	F1(I)	FIRST LOGARITHMIC EXPANSION/CONTRACTION FACTOR FOR THE QUANTIZER STEP SIZE (CORRESPONDING TO QUANTIZER OUTPUT LEVEL L(I))
(0023)	*	F2(I)	SECOND LOGARITHMIC EXPANSION/CONTRACTION FACTOR FOR THE QUANTIZER STEP SIZE (CORRESPONDING TO QUANTIZER OUTPUT LEVEL L(I))
(0024)	*	F(B)	QUANTIZER EXPANSION/CONTRACTION FACTOR IN THE CVSD AND ADM MODES
(0025)	*	G*(K)	CURRENT VALUE OF SLOWLY VARYING PART OF LOGARITHMIC QUANTIZER STEP SIZE
(0026)	*	G*(K+1)	NEXT VALUE OF G*(K)
(0027)	*	M1	2**M1 SAMPLES IS THE TIME CONSTANT OF G*(K)
(0028)	*	C*(K)	CURRENT VALUE OF RAPIDLY VARYING PART OF LOGARITHMIC QUANTIZER STEP SIZE
(0029)	*	C*(K+1)	NEXT VALUE OF C*(K)
(0030)	*	M2	2**M2 SAMPLES IS THE TIME CONSTANT OF C*(K) IN THE ARC MODE AND OF T(K) IN THE CVSD MODE
(0031)	*	T*(K)	PIECEWISE LINEAR APPROXIMATION OF 2***(G*(K)+C*(K)+C-5)
(0032)	*	C	CONSTANT WHICH DETERMINES THE MINIMUM QUANTIZER STEP SIZE
(0033)	*	TMIN	CONSTANT ADDED TO T*(K)
(0034)	*	T(K)	CURRENT VALUE OF QUANTIZER STEP SIZE, T(K)=T*(K)+TMIN IN THE ARC MODE
(0035)	*		
(0036)	*		
(0037)	*		
(0038)	*		
(0039)	*		
(0040)	*		
(0041)	*		
(0042)	*		
(0043)	*	16 KBPS ARC PROGRAM	
(0044)	*		
0000:	A0	(0045) FCB X10100000	00: WRITE: REFRESH '1', LOAD S*(K-3) IN R, CLEAR A
0001:	62	(0046) FCB X01100010	01: ADD: LOAD A AND T WITH B3.S*(K-3), LOAD S*(K-2) IN R
0002:	60	(0047) FCB X01100000	02: ADD: LOAD A AND T WITH B2.S*(K-2)+A, LOAD S*(K-1) IN R
0003:	65	(0048) FCB X01100101	03: ADD: LOAD A AND T WITH B1.S*(K-1)+A+P(K), LOAD T INTO D/A
0004:	00	(0049) FCB X00000000	04: NOP
0005:	00	(0050) FCB X00000000	05: NOP
0006:	90	(0051) FCB X10010000	06: WRITE: A INTO 'SPARE', LOAD Q, LOAD T(K) IN R
0007:	0C	(0052) FCB X00001100	07: ADD: LOAD T WITH +-A(2).R+A, RECIRCULATE A, LOAD T INTO D/A
0008:	00	(0053) FCB X00000000	08: NOP
0009:	00	(0054) FCB X00000000	09: NOP
000A:	94	(0055) FCB X10010100	10: WRITE: A INTO 'SPARE', LOAD Q (LOAD ZERO IF TXFIFO FULL), LOAD T(K) IN R
000B:	0C	(0056) FCB X00001100	11: ADD: LOAD T WITH +-A(1).R+A, RECIRCULATE A, LOAD T INTO D/A
000C:	00	(0057) FCB X00000000	12: NOP
000D:	00	(0058) FCB X00000000	13: NOP
000E:	90	(0059) FCB X10010000	14: WRITE: A INTO 'SPARE', LOAD Q, LOAD T(K) IN R
000F:	40	(0060) FCB X01001101	15: ADD: LOAD A AND T WITH +-L(I).R+A+S*(K), LOAD S*(K-1) IN R, LOAD T INTO D/A
0010:	A0	(0061) FCB X10100000	16: WRITE: A INTO S*(K-1), CLEAR A
0011:	71	(0062) FCB X01110001	17: ADD: LOAD A AND T WITH 1.R, LOAD S*(K-2) IN R
0012:	A0	(0063) FCB X10100000	18: WRITE: A INTO S*(K-2), CLEAR A
0013:	71	(0064) FCB X01110001	19: ADD: LOAD A AND T WITH 1.R, INITIATE ENCODING/DECODING AND STROBE
0014:	A0	(0065) FCB X10100000	20: WRITE: A INTO S*(K-3), CLEAR A, LOAD G*(K) IN R
0015:	73	(0066) FCB X01110011	21: ADD: LOAD A AND T WITH (1-2**(-M1)).R, LOAD '1' IN R
0016:	42	(0067) FCB X01000010	22: ADD: LOAD A AND T WITH F1(I).R+A+G*(K+1)
0017:	88	(0068) FCB X10001000	23: WRITE: A INTO G*(K), CLEAR A BEFORE WRITING IF T IS NEGATIVE
0018:	A0	(0069) FCB X10100000	24: WRITE: A INTO 'SPARE', CLEAR A, LOAD C*(K) IN R
0019:	70	(0070) FCB X01110000	25: ADD: LOAD A AND T WITH (1-2**(-M2)).R, LOAD '1' IN R
001A:	43	(0071) FCB X01000011	26: ADD: LOAD A AND T WITH F2(I).R+A+C*(K+1)
001B:	80	(0072) FCB X10000000	27: WRITE: A INTO C*(K), LOAD G*(K) IN R
001C:	71	(0073) FCB X01110001	28: ADD: LOAD A AND T WITH R.1+A
001D:	C0	(0074) FCB X11000000	29: CONVERT: LOAD T*(K) IN A, LOAD '1' IN R
001E:	63	(0075) FCB X01100011	30: ADD: LOAD A AND T WITH TMIN'.R+A+T(K)
001F:	A0	(0076) FCB X10100000	31: WRITE: A INTO T(K), CLEAR A

TABLE 3-7. MICROPROGRAM (Cont)

(0077) *			
(0078) * 14.4 KBPS ARC PROGRAM			
(0079) *			
0020:	A0	(0080) FCB X10100000	00: WRITE: REFRESH '1', LOAD S'(K-3) IN R, CLEAR A
0021:	52	(0081) FCB X01100010	01: ADD: LOAD A AND T WITH B3.S'(K-3), LOAD S'(K-2) IN R
0022:	60	(0082) FCB X01100000	02: ADD: LOAD A AND T WITH B2.S'(K-2)+A, LOAD S'(K-1) IN R
0023:	65	(0083) FCB X01100101	03: ADD: LOAD A AND T WITH B1.S'(K-1)+A+P(K), LOAD T INTO D/A
0024:	00	(0084) FCB X00000000	04: NOP
0025:	00	(0085) FCB X00000000	05: NOP
0026:	90	(0086) FCB X10010000	06: WRITE: A INTO 'SPARE', LOAD 0, LOAD T(K) IN R
0027:	0C	(0087) FCB X00001100	07: ADD: LOAD T WITH +-A(2).R+A, RECIRCULATE A, LOAD T INTO D/A
0028:	00	(0088) FCB X00000000	08: NOP
0029:	00	(0089) FCB X00000000	09: NOP
002A:	94	(0090) FCB X10010100	10: WRITE: A INTO 'SPARE', LOAD 0 (LOAD ZERO IF TXFIFO FULL), LOAD T(K) IN R
002B:	0C	(0091) FCB X00001100	11: ADD: LOAD T WITH +-A(1).R+A, RECIRCULATE A, LOAD T INTO D/A
002C:	00	(0092) FCB X00000000	12: NOP
002D:	00	(0093) FCB X00000000	13: NOP
002E:	90	(0094) FCB X10010000	14: WRITE: A INTO 'SPARE', LOAD 0, LOAD T(K) IN R
002F:	4D	(0095) FCB X01001101	15: ADD: LOAD A AND T WITH +-L(1).R+A+S'(K), LOAD S'(K-1) IN R, LOAD T INTO D/A
0030:	A0	(0096) FCB X10100000	16: WRITE: A INTO S'(K-1), CLEAR A
0031:	71	(0097) FCB X01110001	17: ADD: LOAD A AND T WITH 1.R, LOAD S'(K-2) IN R
0032:	A0	(0098) FCB X10100000	18: WRITE: A INTO S'(K-2), CLEAR A
0033:	71	(0099) FCB X01110001	19: ADD: LOAD A AND T WITH 1.R, INITIATE ENCODING/DECODING AND STROBE
0034:	A0	(0100) FCB X10100000	20: WRITE: A INTO S'(K-3), CLEAR A, LOAD G'(K) IN R
0035:	73	(0101) FCB X01110011	21: ADD: LOAD A AND T WITH (1-2**(-M1)).R, LOAD '1' IN R
0036:	42	(0102) FCB X01000010	22: ADD: LOAD A AND T WITH F1(1).R+A+G'(K+1)
0037:	88	(0103) FCB X10001000	23: WRITE: A INTO G'(K), CLEAR A BEFORE WRITING IF T IS NEGATIVE
0038:	A0	(0104) FCB X10100000	24: WRITE: A INTO 'SPARE', CLEAR A, LOAD C'(K) IN R
0039:	70	(0105) FCB X01110000	25: ADD: LOAD A AND T WITH (1-2**(-M2)).R, LOAD '1' IN R
003A:	43	(0106) FCB X01000011	26: ADD: LOAD A AND T WITH F2(1).R+A+C'(K+1)
003B:	80	(0107) FCB X10000000	27: WRITE: A INTO C'(K), LOAD G'(K) IN R
003C:	71	(0108) FCB X01110001	28: ADD: LOAD A AND T WITH R.1+A
003D:	C0	(0109) FCB X11000000	29: CONVERT: LOAD T'(K) IN A, LOAD '1' IN R
003E:	63	(0110) FCB X01100011	30: ADD: LOAD A AND T WITH TMIN'.R+A+T(K)
003F:	A0	(0111) FCB X10100000	31: WRITE: A INTO T(K), CLEAR A
(0112) *			
(0113) * 16 KBPS CVSD PROGRAM			
(0114) *			
0040:	00	(0115) FCB \$00	00
0041:	00	(0116) FCB \$00	01
0042:	00	(0117) FCB \$00	02
0043:	00	(0118) FCB \$00	03
0044:	00	(0119) FCB \$00	04
0045:	00	(0120) FCB \$00	05
0046:	00	(0121) FCB \$00	06
0047:	00	(0122) FCB \$00	07
0048:	00	(0123) FCB \$00	08
0049:	00	(0124) FCB \$00	09
004A:	A0	(0125) FCB X10100000	10: WRITE: REFRESH '1', LOAD S'(K-1) IN R, CLEAR A
004B:	76	(0126) FCB X01110110	11: ADD: LOAD A AND T WITH B1.S'(K-1)+P(K), LOAD T INTO D/A
004C:	00	(0127) FCB X00000000	12: NOP
004D:	00	(0128) FCB X00000000	13: NOP
004E:	90	(0129) FCB X10010000	14: WRITE: A INTO 'SPARE', LOAD B, LOAD T(K) IN R.
004F:	43	(0130) FCB X01001001	15: ADD: LOAD A AND T WITH +-L.T(K)+A, LOAD '1' IN R.
0050:	00	(0131) FCB \$00	16
0051:	00	(0132) FCB \$00	17
0052:	00	(0133) FCB \$00	18
0053:	00	(0134) FCB \$00	19
0054:	00	(0135) FCB \$00	20
0055:	00	(0136) FCB \$00	21
0056:	00	(0137) FCB \$00	22
0057:	00	(0138) FCB \$00	23
0058:	00	(0139) FCB \$00	24
0059:	00	(0140) FCB \$00	25
005A:	7F	(0141) FCB X01111111	26: ADD: LOAD A AND T WITH +-R.TMIN+A+S'(K), LOAD T INTO D/A
005B:	A0	(0142) FCB X10100000	27: WRITE: A INTO S'(K-1), CLEAR A, LOAD T(K) IN R.
005C:	70	(0143) FCB X01110000	28: ADD: LOAD A AND T WITH (1-2**(-M2)).R, INITIATE STROBE
005D:	80	(0144) FCB X10000000	29: WRITE: A INTO 'SPARE', LOAD '1' IN R.
005E:	42	(0145) FCB X01000010	30: ADD: LOAD A AND T WITH F(B).R+A+T(K+1).
005F:	A8	(0146) FCB X10101000	31: WRITE: A INTO T(K), CLEAR A BEFORE WRITING IF T IS NEGATIVE, CLEAR A

TABLE 3-7. MICROPROGRAM (Cont)

```

(0147) *
(0148) * 16 KOPS ADM PROGRAM
(0149) *
0060: 00 (0150) FCB $00 -00
0061: 00 (0151) FCB $00 01
0062: 00 (0152) FCB $00 02
0063: 00 (0153) FCB $00 03
0064: 00 (0154) FCB $00 04
0065: 00 (0155) FCB $00 05
0066: 00 (0156) FCB $00 06
0067: 00 (0157) FCB $00 07
0068: 00 (0158) FCB $00 08
0069: 00 (0159) FCB $00 09
006A: A0 (0160) FCB $10100000 10: WRITE: REFRESH '1'. LOAD S'(K-1) IN R. CLEAR A
006B: 65 (0161) FCB $01100101 11: ADD: LOAD A AND T WITH B1.S'(K-1)+P(K). LOAD T INTO D/A
006C: 00 (0162) FCB $00000000 12: NOP
006D: 00 (0163) FCB $00000000 13: NOP
006E: 90 (0164) FCB $10010000 14: WRITE: A INTO 'SPARE'. LOAD B. LOAD T(K) IN R.
006F: 4D (0165) FCB $01001101 15: ADD: LOAD A AND T WITH +-L.T(K)+A=S'(K). LOAD T INTO D/A.
0070: 00 (0166) FCB $00 16
0071: 00 (0167) FCB $00 17
0072: 00 (0168) FCB $00 18
0073: 00 (0169) FCB $00 19
0074: 00 (0170) FCB $00 20
0075: 00 (0171) FCB $00 21
0076: 00 (0172) FCB $00 22
0077: 00 (0173) FCB $00 23
0078: 00 (0174) FCB $00 24
0079: 00 (0175) FCB $00 25
007A: A0 (0176) FCB $10100000 26: WRITE: A INTO S'(K-1). CLEAR A. LOAD G'(K) IN R
007B: 73 (0177) FCB $01110011 27: ADD: LOAD A AND T WITH (1-2**(-M1)).R. LOAD '1' IN R
007C: 42 (0178) FCB $01000010 28: ADD: LOAD A AND T WITH F(B).R+A=G'(K+1). INITIATE STROBE
007D: 88 (0179) FCB $10001000 29: WRITE: A INTO G'(K). CLEAR A BEFORE WRITING IF T IS NEGATIVE
007E: C0 (0180) FCB $11000000 30: CONVERT: LOAD A WITH T(K)
007F: A0 (0181) FCB $10100000 31: WRITE: A INTO T(K). CLEAR A
(0182) *
(0183) * 9.6 KOPS ARC PROGRAM
(0184) *
0080: A0 (0185) FCB $10100000 00: WRITE: REFRESH '1'. LOAD S'(K-3) IN R. CLEAR A
0081: 62 (0186) FCB $01100010 01: ADD: LOAD A AND T WITH B3.S'(K-3). LOAD S'(K-2) IN R
0082: 60 (0187) FCB $01100000 02: ADD: LOAD A AND T WITH B2.S'(K-2)+A. LOAD S'(K-1) IN R
0083: 65 (0188) FCB $01100101 03: ADD: LOAD A AND T WITH B1.S'(K-1)+A=P(K). LOAD T INTO D/A
0084: 00 (0189) FCB $00000000 04: NOP
0085: 00 (0190) FCB $00000000 05: NOP
0086: 90 (0191) FCB $10010000 06: WRITE: A INTO 'SPARE'. LOAD Q. LOAD T(K) IN R
0087: 0C (0192) FCB $00001100 07: ADD: LOAD T WITH +-A(2).R+A. RECIRCULATE A. LOAD T INTO D/A
0088: 00 (0193) FCB $00000000 08: NOP
0089: 00 (0194) FCB $00000000 09: NOP
008A: 90 (0195) FCB $10010000 10: WRITE: A INTO 'SPARE'. LOAD Q. LOAD T(K) IN R
008B: 0C (0196) FCB $00001100 11: ADD: LOAD T WITH +-A(1).R+A. RECIRCULATE A. LOAD T INTO D/A
008C: 00 (0197) FCB $00000000 12: NOP
008D: 00 (0198) FCB $00000000 13: NOP
008E: 90 (0199) FCB $10010000 14: WRITE: A INTO 'SPARE'. LOAD Q. LOAD T(K) IN R
008F: 4D (0200) FCB $01001101 15: ADD: LOAD A AND T WITH +-L(I).R+A=S'(K). LOAD S'(K-1) IN R. LOAD T INTO D/A
0090: A0 (0201) FCB $10100000 16: WRITE: A INTO S'(K-1). CLEAR A
0091: 71 (0202) FCB $01110001 17: ADD: LOAD A AND T WITH 1.R. LOAD S'(K-2) IN R
0092: A0 (0203) FCB $10100000 18: WRITE: A INTO S'(K-2). CLEAR A
0093: 71 (0204) FCB $01110001 19: ADD: LOAD A AND T WITH 1.R. INITIATE ENCODING/DECODING AND STROBE
0094: A0 (0205) FCB $10100000 20: WRITE: A INTO S'(K-3). CLEAR A. LOAD G'(K) IN R
0095: 73 (0206) FCB $02061101 21: ADD: LOAD A AND T WITH (1-2**(-M1)).R. LOAD '1' IN R
0096: 42 (0207) FCB $01000010 22: ADD: LOAD A AND T WITH F1(I).R+A=G'(K+1)
0097: 88 (0208) FCB $10001000 23: WRITE: A INTO G'(K). CLEAR A BEFORE WRITING IF T IS NEGATIVE
0098: A0 (0209) FCB $10100000 24: WRITE: A INTO 'SPARE'. CLEAR A. LOAD C'(K) IN R
0099: 70 (0210) FCB $01110000 25: ADD: LOAD A AND T WITH (1-2**(-M2)).R. LOAD '1' IN R
009A: 43 (0211) FCB $01000011 26: ADD: LOAD A AND T WITH F2(I).R+A=C'(K+1)
009B: 80 (0212) FCB $10000000 27: WRITE: A INTO C'(K). LOAD G'(K) IN R
009C: 71 (0213) FCB $01110001 28: ADD: LOAD A AND T WITH R.1+A
009D: C0 (0214) FCB $11000000 29: CONVERT: LOAD T'(K) IN A. LOAD '1' IN R
009E: 63 (0215) FCB $01100011 30: ADD: LOAD A AND T WITH TMIN'.R+A=T(K)
009F: A0 (0216) FCB $10100000 31: WRITE: A INTO T(K). CLEAR A

```

TABLE 3-7. MICROPROGRAM (Cont)

	(0217) *		
	(0210) * UNSPECIFIED		
	(0219) *		
00A0:	00 (0220) FCB \$00	00	
00A1:	00 (0221) FCB \$00	01	
00A2:	00 (0222) FCB \$00	02	
00A3:	00 (0223) FCB \$00	03	
00A4:	00 (0224) FCB \$00	04	
00A5:	00 (0225) FCB \$00	05	
00A6:	00 (0226) FCB \$00	06	
00A7:	00 (0227) FCB \$00	07	
00A8:	00 (0228) FCB \$00	08	
00A9:	00 (0229) FCB \$00	09	
00AA:	00 (0230) FCB \$00	10	
00AB:	00 (0231) FCB \$00	11	
00AC:	00 (0232) FCB \$00	12	
00AD:	00 (0233) FCB \$00	13	
00AE:	00 (0234) FCB \$00	14	
00AF:	00 (0235) FCB \$00	15	
00B0:	00 (0236) FCB \$00	16	
00B1:	00 (0237) FCB \$00	17	
00B2:	00 (0238) FCB \$00	18	
00B3:	00 (0239) FCB \$00	19	
00B4:	00 (0240) FCB \$00	20	
00B5:	00 (0241) FCB \$00	21	
00B6:	00 (0242) FCB \$00	22	
00B7:	00 (0243) FCB \$00	23	
00B8:	00 (0244) FCB \$00	24	
00B9:	00 (0245) FCB \$00	25	
00BA:	00 (0246) FCB \$00	26	
00BB:	00 (0247) FCB \$00	27	
00BC:	00 (0248) FCB \$00	28	
00BD:	00 (0249) FCB \$00	29	
00BE:	00 (0250) FCB \$00	30	
00BF:	00 (0251) FCB \$00	31	
	(0252) *		
	(0253) * 9.6 KBPS CVSD PROGRAM		
	(0254) *		
00C0:	00 (0255) FCB \$00	00	
00C1:	00 (0256) FCB \$00	01	
00C2:	00 (0257) FCB \$00	02	
00C3:	00 (0258) FCB \$00	03	
00C4:	00 (0259) FCB \$00	04	
00C5:	00 (0260) FCB \$00	05	
00C6:	00 (0261) FCB %00000000	06: NOP	
00C7:	00 (0262) FCB %00000000	07: NOP	
00C8:	00 (0263) FCB %00000000	08: NOP	
00C9:	00 (0264) FCB %00000000	09: NOP	
00CA:	00 (0265) FCB %10100000	10: WRITE: REFRESH '1'. LOAD S'(K-1) IN R. CLEAR A	
00CB:	76 (0266) FCB %01110110	11: ADD: LOAD A AND T WITH S'(K-1)+P(K). LOAD T INTO D/A	
00CC:	00 (0267) FCB %00000000	12: NOP	
00CD:	00 (0268) FCB %00000000	13: NOP	
00CE:	90 (0269) FCB %10010000	14: WRITE: A INTO 'SPARE'. LOAD B. LOAD T(K) IN R.	
00CF:	49 (0270) FCB %01001001	15: ADD: LOAD A AND T WITH +-L.T(K)+A. LOAD '1' IN R.	
00D0:	00 (0271) FCB \$00	16	
00D1:	00 (0272) FCB \$00	17	
00D2:	00 (0273) FCB \$00	18	
00D3:	00 (0274) FCB \$00	19	
00D4:	00 (0275) FCB \$00	20	
00D5:	00 (0276) FCB \$00	21	
00D6:	7F (0277) FCB %01111111	22: ADD: LOAD A AND T WITH +-R.T(I)+A+S'(K). LOAD T INTO D/A	
00D7:	00 (0278) FCB %10100000	23: WRITE: A INTO S'(K-1). CLEAR A	
00D8:	A0 (0279) FCB %10100000	24: WRITE: A INTO 'SPARE'. CLEAR A. LOAD T(K) IN R	
00D9:	70 (0280) FCB %01110000	25: ADD: LOAD A AND T WITH (1-2**(-M2)).R. INITIATE STROBE	
00DA:	80 (0281) FCB %10000000	26: WRITE: A INTO 'SPARE'. LOAD '1' IN R	
00DB:	42 (0282) FCB %01000010	27: ADD: LOAD A AND T WITH F(B).R+A+T(K+1)	
00DC:	A8 (0283) FCB %10101000	28: WRITE: A INTO T(K). CLEAR A BEFORE WRITING IF T IS NEGATIVE. CLEAR A	
00DD:	00 (0284) FCB %00000000	29: NOP	
00DE:	00 (0285) FCB %00000000	30: NOP	
00DF:	00 (0286) FCB %00000000	31: NOP	
	(0287) *		
	(0288) * 3.6 KBPS ADM PROGRAM		

TABLE 3-7. MICROPROGRAM (Cont)

	(0289) *			
00E0:	00	(0290)	FCB \$00	00
00E1:	00	(0291)	FCB \$00	01
00E2:	00	(0292)	FCB \$00	02
00E3:	00	(0293)	FCB \$00	03
00E4:	00	(0294)	FCB \$00	04
00E5:	00	(0295)	FCB \$00	05
00E6:	A0	(0296)	FCB X10100000	06: WRITE: REFRESH '1'. LOAD S'(K-3) IN R. CLEAR A
00E7:	62	(0297)	FCB X01100010	07: ADD: LOAD A AND T WITH B3.S'(K-3). LOAD S'(K-2) IN R
00E8:	60	(0298)	FCB X01100000	08: ADD: LOAD A AND T WITH B2.S'(K-2)+A. LOAD S'(K-1) IN R
00E9:	65	(0299)	FCB X01100101	09: ADD: LOAD A AND T WITH B1.S'(K-1)+A+P(K).LOAD T INTO D/A
00EA:	00	(0300)	FCB X00000000	10: NOP
00EB:	00	(0301)	FCB X00000000	11: NOP
00EC:	90	(0302)	FCB X10010000	12: WRITE: A INTO 'SPARE', LOAD B. LOAD T(K) IN R
00ED:	40	(0303)	FCB X01001101	13: ADD: LOAD A AND T WITH +-L.T(K)+A+S'(K). LOAD T INTO D/A. LOAD S'(K-1) IN R
00EE:	A0	(0304)	FCB X10100000	14: WRITE: A INTO S'(K-1). CLEAR A
00EF:	71	(0305)	FCB X01110001	15: ADD: LOAD A AND T WITH R.1. LOAD S'(K-2) IN R
00F0:	00	(0306)	FCB \$00	16
00F1:	00	(0307)	FCB \$00	17
00F2:	00	(0308)	FCB \$00	18
00F3:	00	(0309)	FCB \$00	19
00F4:	00	(0310)	FCB \$00	20
00F5:	00	(0311)	FCB \$00	21
00F6:	A6	(0312)	FCB X10100000	22: WRITE: A INTO S'(K-2). CLEAR A
00F7:	71	(0313)	FCB X01110001	23: ADD: LOAD A AND T WITH R.1
00F8:	A0	(0314)	FCB X10100000	24: WRITE: A INTO S'(K-3). CLEAR A. LOAD G'(K) IN R
00F9:	73	(0315)	FCB X01110011	25: ADD: LOAD A AND T WITH (1-2**(-M1)).R. LOAD '1' IN R
00FA:	42	(0316)	FCB X01000010	26: ADD: LOAD A AND T WITH F(B).1+A+G'(K+1). INITIATE STROBE
00FB:	88	(0317)	FCB X10001000	27: WRITE: A INTO G'(K). CLEAR A BEFORE WRITING IF T IS NEGATIVE
00FC:	C0	(0318)	FCB X11000000	28: CONVERT: LOAD T(K) IN A
00FD:	A0	(0319)	FCB X10100000	29: WRITE: A INTO T(K). CLEAR A
00FE:	00	(0320)	FCB X00000000	30: NOP
00FF:	00	(0321)	FCB X00000000	31: NOP
	0100	(0322)	END	

TABLE 3-8. WRITE ADDRESS

		(0001) *	WRITE-ADDRESS		
		(0002) *	THIS LISTING SPECIFIES THE LAST 4 BITS (P9-P12) OF THE		
		(0003) *	PROGRAM WORD DURING BIT TIMES 1 TO 8. BITS P10-P12 ARE		
		(0004) *	PART OF THE WRITE ADDRESS OF THE SCRATCHPAD		
		(0005) *			
		(0006) *	16 KBPS ARC PROGRAM		
		(0007) *			
0000:	08	(0008)	FCB %1000	00:	WRITE: '1'
0001:	0D	(0009)	FCB %1101	01:	ADD
0002:	09	(0010)	FCB %1001	02:	ADD
0003:	00	(0011)	FCB %0000	03:	ADD
0004:	00	(0012)	FCB %0000	04:	NOP
0005:	00	(0013)	FCB %0000	05:	NOP
0006:	07	(0014)	FCB %0111	06:	WRITE: SPARE
0007:	00	(0015)	FCB %0000	07:	ADD
0008:	00	(0016)	FCB %0000	08:	NOP
0009:	00	(0017)	FCB %0000	09:	NOP
000A:	07	(0018)	FCB %0111	10:	WRITE: SPARE
000B:	00	(0019)	FCB %0000	11:	ADD
000C:	00	(0020)	FCB %0000	12:	NOP
000D:	00	(0021)	FCB %0000	13:	NOP
000E:	07	(0022)	FCB %0111	14:	WRITE: SPARE
000F:	09	(0023)	FCB %1001	15:	ADD
0010:	01	(0024)	FCB %0001	16:	WRITE: S'(K-1)
0011:	0D	(0025)	FCB %1101	17:	ADD
0012:	05	(0026)	FCB %0101	18:	WRITE: S'(K-2)
0013:	04	(0027)	FCB %0100	19:	ADD: INITIATE ENCODING
					DECODING AND STROBE
0014:	03	(0028)	FCB %0011	20:	WRITE: S'(K-3)
0015:	08	(0029)	FCB %1000	21:	ADD
0016:	00	(0030)	FCB %0000	22:	ADD
0017:	04	(0031)	FCB %0100	23:	WRITE: G'(K)
0018:	07	(0032)	FCB %0111	24:	WRITE: SPARE
0019:	08	(0033)	FCB %1000	25:	ADD
001A:	00	(0034)	FCB %0000	26:	ADD
001B:	06	(0035)	FCB %0110	27:	WRITE: C'(K)
001C:	00	(0036)	FCB %0000	28:	ADD
001D:	08	(0037)	FCB %1000	29:	CONVERT: C=0
001E:	00	(0038)	FCB %0000	30:	ADD
001F:	02	(0039)	FCB %0010	31:	WRITE: T(K)
		(0040) *			
		(0041) *	14.4 KBPS ARC PROGRAM		
		(0042) *			
0020:	05	(0043)	FCB %1000	00:	WRITE: '1'
0021:	0D	(0044)	FCB %1101	01:	ADD
0022:	09	(0045)	FCB %1001	02:	ADD
0023:	00	(0046)	FCB %0000	03:	ADD
0024:	00	(0047)	FCB %0000	04:	NOP
0025:	00	(0048)	FCB %0000	05:	NOP
0026:	07	(0049)	FCB %0111	06:	WRITE: SPARE
0027:	00	(0050)	FCB %0000	07:	ADD
0028:	00	(0051)	FCB %0000	08:	NOP
0029:	00	(0052)	FCB %0000	09:	NOP
002A:	07	(0053)	FCB %0111	10:	WRITE SPARE
002B:	00	(0054)	FCB %0000	11:	ADD
002C:	00	(0055)	FCB %0000	12:	NOP
002D:	00	(0056)	FCB %0000	13:	NOP
002E:	07	(0057)	FCB %0111	14:	WRITE: SPARE

TABLE 3-8. WRITE ADDRESS (Cont)

002F:	09	(0058)	FCB	%1001	15: ADD
0030:	01	(0059)	FCB	%0001	16: WRITE: S'(K-1)
0031:	00	(0060)	FCB	%1101	17: ADD
0032:	05	(0061)	FCB	%0101	18: WRITE: S'(K-2)
0033:	04	(0062)	FCB	%0100	19: ADD: INITIATE ENCODING /DECODING AND STROBE
0034:	03	(0063)	FCB	%0011	20: WRITE: S'(K-3)
0035:	08	(0064)	FCB	%1000	21: ADD
0036:	00	(0065)	FCB	%0000	22: ADD
0037:	04	(0066)	FCB	%0100	23: WRITE: G'(K)
0038:	07	(0067)	FCB	%0111	24: WRITE: SPARE
0039:	08	(0068)	FCB	%1000	25: ADD
003A:	00	(0069)	FCB	%0000	26: ADD
003B:	06	(0070)	FCB	%0110	27: WRITE: C'(K)
003C:	00	(0071)	FCB	%0000	28: ADD
003D:	08	(0072)	FCB	%1000	29: CONVERT: C=8
003E:	00	(0073)	FCB	%0000	30: ADD
003F:	02	(0074)	FCB	%0010	31: WRITE: T(K)
		(0075)	*		
		(0076)	*	16 KBPS CVSD PROGRAM	
		(0077)	*		
0040:	00	(0078)	FCB	\$0	00
0041:	00	(0079)	FCB	\$0	01
0042:	00	(0080)	FCB	\$0	02
0043:	00	(0081)	FCB	\$0	03
0044:	00	(0082)	FCB	\$0	04
0045:	00	(0083)	FCB	\$0	05
0046:	00	(0084)	FCB	\$0	06
0047:	00	(0085)	FCB	\$0	07
0048:	00	(0086)	FCB	\$0	08
0049:	00	(0087)	FCB	\$0	09
004A:	08	(0088)	FCB	%1000	10: WRITE: '1'
004B:	00	(0089)	FCB	%0000	11: ADD
004C:	00	(0090)	FCB	%0000	12: NOP
004D:	00	(0091)	FCB	%0000	13: NOP
004E:	07	(0092)	FCB	%0111	14: WRITE: SPARE
004F:	08	(0093)	FCB	%1000	15: ADD
0050:	00	(0094)	FCB	\$0	16
0051:	00	(0095)	FCB	\$0	17
0052:	00	(0096)	FCB	\$0	18
0053:	00	(0097)	FCB	\$0	19
0054:	00	(0098)	FCB	\$0	20
0055:	00	(0099)	FCB	\$0	21
0056:	00	(0100)	FCB	\$0	22
0057:	00	(0101)	FCB	\$0	23
0058:	00	(0102)	FCB	\$0	24
0059:	00	(0103)	FCB	\$0	25
005A:	00	(0104)	FCB	%0000	26: ADD
005B:	01	(0105)	FCB	%0001	27: WRITE: S'(K-1)
005C:	04	(0106)	FCB	%0100	28: ADD: INITIATE STROBE
005D:	07	(0107)	FCB	%0111	29: WRITE: SPARE
005E:	00	(0108)	FCB	%0000	30: ADD
005F:	02	(0109)	FCB	%0010	31: WRITE: T(K)
		(0110)	*		

TABLE 3-8. WRITE ADDRESS (Cont)

		(0111) * 16 KBFS ADM PROGRAM		
0060:	00	(0112) FCB \$0	00	
0061:	00	(0113) FCB \$0	01	
0062:	00	(0114) FCB \$0	02	
0063:	00	(0115) FCB \$0	03	
0064:	00	(0116) FCB \$0	04	
0065:	00	(0117) FCB \$0	05	
0066:	00	(0118) FCB \$0	06	
0067:	00	(0119) FCB \$0	07	
0068:	00	(0120) FCB \$0	08	
0069:	00	(0121) FCB \$0	09	
006A:	08	(0122) FCB %1000	10:	WRITE: '1'
006B:	00	(0123) FCB %0000	11:	ADD
006C:	00	(0124) FCB %0000	12:	NOP
006D:	00	(0125) FCB %0000	13:	NOP
006E:	07	(0126) FCB %0111	14:	WRITE: SPARE
006F:	00	(0127) FCB %0000	15:	ADD
0070:	00	(0128) FCB \$0	16	
0071:	00	(0129) FCB \$0	17	
0072:	00	(0130) FCB \$0	18	
0073:	00	(0131) FCB \$0	19	
0074:	00	(0132) FCB \$0	20	
0075:	00	(0133) FCB \$0	21	
0076:	00	(0134) FCB \$0	22	
0077:	00	(0135) FCB \$0	23	
0078:	00	(0136) FCB \$0	24	
0079:	00	(0137) FCB \$0	25	
007A:	01	(0138) FCB %0001	26:	WRITE: S'(K-1)
007B:	08	(0139) FCB %1000	27:	ADD
007C:	04	(0140) FCB %0100	28:	ADD: INITIATE STROBE
007D:	04	(0141) FCB %0100	29:	WRITE: G'(K)
007E:	07	(0142) FCB %0111	30:	CONVERT: C=7
007F:	02	(0143) FCB %0010	31:	WRITE: T(K)
		(0144) *		
		(0145) * 9.6 KBFS ARC PROGRAM		
		(0146) *		
0080:	03	(0147) FCB %1001	00:	WRITE: '1'
0081:	0D	(0148) FCB %1101	01:	ADD
0082:	09	(0149) FCB %1001	02:	ADD
0083:	00	(0150) FCB %0000	03:	ADD
0084:	00	(0151) FCB %0000	04:	NOP
0085:	00	(0152) FCB %0000	05:	NOP
0086:	07	(0153) FCB %0111	06:	WRITE: SPARE
0087:	00	(0154) FCB %0000	07:	ADD
0088:	00	(0155) FCB %0000	08:	NOP
0089:	00	(0156) FCB %0000	09:	NOP
008A:	07	(0157) FCB %0111	10:	WRITE SPARE
008B:	00	(0158) FCB %0000	11:	ADD
008C:	00	(0159) FCB %0000	12:	NOP
008D:	00	(0160) FCB %0000	13:	NOP
008E:	07	(0161) FCB %0111	14:	WRITE: SPARE
008F:	09	(0162) FCB %1001	15:	ADD
0090:	00	(0163) FCB %0000	16:	WRITE: S'(K-1)
0091:	0D	(0164) FCB %1101	17:	ADD
0092:	05	(0165) FCB %0101	18:	WRITE: S'(K-2)
0093:	04	(0166) FCB %0100	19:	ADD: INITIATE ENCODING /DECODING AND STROBE

TABLE 3-8. WRITE ADDRESS (Cont)

0094:	03	(0167)	FCB	%0011	20: WRITE: S'(K-3)
0095:	00	(0168)	FCB	%1000	21: ADD
0096:	00	(0169)	FCB	%0000	22: ADD
0097:	04	(0170)	FCB	%0100	23: WRITE: G'(K)
0098:	07	(0171)	FCB	%0111	24: WRITE: SPARE
0099:	06	(0172)	FCB	%1000	25: ADD
009A:	00	(0173)	FCB	%0000	26: ADD
009B:	06	(0174)	FCB	%0110	27: WRITE: C'(K)
009C:	00	(0175)	FCB	%0000	28: ADD
009D:	09	(0176)	FCB	%1001	29: CONVERT: C=9
009E:	00	(0177)	FCB	%0000	30: ADD
009F:	02	(0178)	FCB	%0010	31: WRITE: T(K)
		(0179)	*		
		(0180)	*	UNSPECIFIED	
		(0181)	*		
00A0:	00	(0182)	FCB	\$0	00
00A1:	00	(0183)	FCB	\$0	01
00A2:	00	(0184)	FCB	\$0	02
00A3:	00	(0185)	FCB	\$0	03
00A4:	00	(0186)	FCB	\$0	04
00A5:	00	(0187)	FCB	\$0	05
00A6:	00	(0188)	FCB	\$0	06
00A7:	00	(0189)	FCB	\$0	07
00A8:	00	(0190)	FCB	\$0	08
00A9:	00	(0191)	FCB	\$0	09
00AA:	00	(0192)	FCB	\$0	10
00AB:	00	(0193)	FCB	\$0	11
00AC:	00	(0194)	FCB	\$0	12
00AD:	00	(0195)	FCB	\$0	13
00AE:	00	(0196)	FCB	\$0	14
00AF:	00	(0197)	FCB	\$0	15
00B0:	00	(0198)	FCB	\$0	16
00B1:	00	(0199)	FCB	\$0	17
00B2:	00	(0200)	FCB	\$0	18
00B3:	00	(0201)	FCB	\$0	19
00B4:	00	(0202)	FCB	\$0	20
00B5:	00	(0203)	FCB	\$0	21
00B6:	00	(0204)	FCB	\$0	22
00B7:	00	(0205)	FCB	\$0	23
00B8:	00	(0206)	FCB	\$0	24
00B9:	00	(0207)	FCB	\$0	25
00BA:	00	(0208)	FCB	\$0	26
00BB:	00	(0209)	FCB	\$0	27
00BC:	00	(0210)	FCB	\$0	28
00BD:	00	(0211)	FCB	\$0	29
00BE:	00	(0212)	FCB	\$0	30
00BF:	00	(0213)	FCB	\$0	31
		(0214)	*		
		(0215)	*	9.6 KBPS CVSD PROGRAM	
		(0216)	*		
00C0:	00	(0217)	FCB	\$0	00
00C1:	00	(0218)	FCB	\$0	01
00C2:	00	(0219)	FCB	\$0	02
00C3:	00	(0220)	FCB	\$0	03
00C4:	00	(0221)	FCB	\$0	04
00C5:	00	(0222)	FCB	\$0	05
00C6:	00	(0223)	FCB	%0000	06: NOP
00C7:	00	(0224)	FCB	%0000	07: NOP
00C8:	00	(0225)	FCB	%0000	08: NOP

TABLE 3-8. WRITE ADDRESS (Cont)

00C9:	00	(0226)	FCB	%0000	09: NOP
00CA:	08	(0227)	FCB	%1000	10: WRITE: '1'
00CB:	00	(0228)	FCB	%0000	11: ADD
00CC:	00	(0229)	FCB	%0000	12: NOP
00CD:	00	(0230)	FCB	%0000	13: NOP
00CE:	07	(0231)	FCB	%0111	14: WRITE: SPARE
00CF:	08	(0232)	FCB	%1000	15: ADD
00D0:	00	(0233)	FCB	\$0	16
00D1:	00	(0234)	FCB	\$0	17
00D2:	00	(0235)	FCB	\$0	18
00D3:	00	(0236)	FCB	\$0	19
00D4:	00	(0237)	FCB	\$0	20
00D5:	00	(0238)	FCB	\$0	21
00D6:	00	(0239)	FCB	%0000	22: ADD
00D7:	01	(0240)	FCB	%0001	23: WRITE: S'(K-1)
00D8:	07	(0241)	FCB	%0111	24: WRITE: SPARE
00D9:	04	(0242)	FCB	%0100	25: ADD: INITIATE STROBE
00DA:	07	(0243)	FCB	%0111	26: WRITE: SPARE
00DB:	00	(0244)	FCB	%0000	27: ADD
00DC:	02	(0245)	FCB	%0010	28: WRITE: T(K)
00DD:	00	(0246)	FCB	%0000	29: NOP
00DE:	00	(0247)	FCB	%0000	30: NOP
00DF:	00	(0248)	FCB	%0000	31: NOP
		(0249) *			
		(0250) * 9.6 Kbps ADM PROGRAM			
		(0251) *			
00E0:	00	(0252)	FCB	\$0	00
00E1:	00	(0253)	FCB	\$0	01
00E2:	00	(0254)	FCB	\$0	02
00E3:	00	(0255)	FCB	\$0	03
00E4:	00	(0256)	FCB	\$0	04
00E5:	00	(0257)	FCB	\$0	05
00E6:	08	(0258)	FCB	%1000	06: WRITE: '1'
00E7:	0D	(0259)	FCB	%1101	07: ADD
00E8:	09	(0260)	FCB	%1001	08: ADD
00E9:	00	(0261)	FCB	%0000	09: ADD
00EA:	00	(0262)	FCB	%0000	10: NOP
00EB:	00	(0263)	FCB	%0000	11: NOP
00EC:	07	(0264)	FCB	%0111	12: WRITE: SPARE
00ED:	09	(0265)	FCB	%1001	13: ADD
00EE:	01	(0266)	FCB	%0001	14: WRITE: S'(K-1)
00EF:	08	(0267)	FCB	%1000	15: ADD
00F0:	00	(0268)	FCB	\$0	16
00F1:	00	(0269)	FCB	\$0	17
00F2:	00	(0270)	FCB	\$0	18
00F3:	00	(0271)	FCB	\$0	19
00F4:	00	(0272)	FCB	\$0	20
00F5:	00	(0273)	FCB	\$0	21
00F6:	05	(0274)	FCB	%0101	22: WRITE: S'(K-2)
00F7:	00	(0275)	FCB	%0000	23: ADD
00F8:	03	(0276)	FCB	%0011	24: WRITE: S'(K-3)
00F9:	08	(0277)	FCB	%1000	25: ADD
00FA:	04	(0278)	FCB	%0100	26: ADD: INITIATE STROBE
00FB:	04	(0279)	FCB	%0100	27: WRITE: G'(K)
00FC:	07	(0280)	FCB	%0111	28: CONVERT: C=7
00FD:	02	(0281)	FCB	%0010	29: WRITE: T(K)
00FE:	00	(0282)	FCB	%0000	30: NOP
00FF:	00	(0283)	FCB	%0000	31: NOP
		(0284) END			

0100

TABLE 3-9. READ ADDRESS

(0001) * READ-ADDRESS
 (0002) * THIS LISTING SPECIFIES THE LAST 4 BITS (P9-P12) OF THE PROGRAM
 (0003) * WORD DURING BIT TIMES 9 TO 16. BITS P10-P12 ARE PART OF
 (0004) * THE READ ADDRESS OF THE SCRATCHPAD
 (0005) *
 (0006) * 16 KBPS ARC PROGRAM
 (0007) *

0000:	0B	(0008)	FCB	%1011	00: WRITE: LOAD S'(K-3)
0001:	0D	(0009)	FCB	%1101	01: ADD: LOAD S'(K-2)
0002:	09	(0010)	FCB	%1001	02: ADD: LOAD S'(K-1)
0003:	00	(0011)	FCB	%0000	03: ADD
0004:	00	(0012)	FCB	%0000	04: NOP
0005:	00	(0013)	FCB	%0000	05: NOP
0006:	0A	(0014)	FCB	%1010	06: WRITE: LOAD T(K)
0007:	00	(0015)	FCB	%0000	07: ADD
0008:	00	(0016)	FCB	%0000	08: NOP
0009:	00	(0017)	FCB	%0000	09: NOP
000A:	0A	(0018)	FCB	%1010	10: WRITE: LOAD T(K)
000B:	00	(0019)	FCB	%0000	11: ADD
000C:	00	(0020)	FCB	%0000	12: NOP
000D:	00	(0021)	FCB	%0000	13: NOP
000E:	0A	(0022)	FCB	%1010	14: WRITE: LOAD T(K)
000F:	09	(0023)	FCB	%1001	15: ADD: LOAD S'(K-1)
0010:	00	(0024)	FCB	%0000	16: WRITE
0011:	0D	(0025)	FCB	%1101	17: ADD: LOAD S'(K-2)
0012:	00	(0026)	FCB	%0000	18: WRITE
0013:	04	(0027)	FCB	%0100	19: ADD: INITIATE ENCODING /DECODING AND STROBE
0014:	0C	(0028)	FCB	%1100	20: WRITE: LOAD G'(K)
0015:	08	(0029)	FCB	%1000	21: ADD: LOAD '1'
0016:	00	(0030)	FCB	%0000	22: ADD
0017:	00	(0031)	FCB	%0000	23: WRITE
0018:	0E	(0032)	FCB	%1110	24: WRITE: LOAD C'(K)
0019:	08	(0033)	FCB	%1000	25: ADD: LOAD '1'
001A:	00	(0034)	FCB	%0000	26: ADD
001B:	0C	(0035)	FCB	%1100	27: WRITE: LOAD G'(K)
001C:	00	(0036)	FCB	%0000	28: ADD
001D:	08	(0037)	FCB	%1000	29: CONVERT: K=8, LOAD '1' IN R
001E:	00	(0038)	FCB	%0000	30: ADD
001F:	00	(0039)	FCB	%0000	31: WRITE

(0040) *
 (0041) * 14.4 KBPS ARC PROGRAM
 (0042) *

0020:	0B	(0043)	FCB	%1011	00: WRITE: LOAD S'(K-3)
0021:	0D	(0044)	FCB	%1101	01: ADD: LOAD S'(K-2)
0022:	09	(0045)	FCB	%1001	02: ADD: LOAD S'(K-1)
0023:	00	(0046)	FCB	%0000	03: ADD
0024:	00	(0047)	FCB	%0000	04: NOP
0025:	00	(0048)	FCB	%0000	05: NOP
0026:	0A	(0049)	FCB	%1010	06: WRITE: LOAD T(K)
0027:	00	(0050)	FCB	%0000	07: ADD
0028:	00	(0051)	FCB	%0000	08: NOP
0029:	00	(0052)	FCB	%0000	09: NOP
002A:	0A	(0053)	FCB	%1010	10: WRITE: LOAD T(K)
002B:	0C	(0054)	FCB	%0000	11: ADD
002C:	00	(0055)	FCB	%0000	12: NOP
002D:	00	(0056)	FCB	%0000	13: NOP
002E:	0A	(0057)	FCB	%1010	14: WRITE: LOAD T(K)
002F:	09	(0058)	FCB	%1001	15: ADD: LOAD S'(K-1)

TABLE 3-9. READ ADDRESS (Cont)

0030:	00	(0059)	FCB	%0000	16: WRITE
0031:	00	(0060)	FCB	%1101	17: ADD: LOAD S'(K-2)
0032:	00	(0061)	FCB	%0000	18: WRITE
0033:	04	(0062)	FCB	%0100	19: ADD: INITIATE ENCODING /DECODING AND STROBE
0034:	0C	(0063)	FCB	%1100	20: WRITE: LOAD G'(K)
0035:	08	(0064)	FCB	%1000	21: ADD: LOAD '1'
0036:	00	(0065)	FCB	%0000	22: ADD
0037:	00	(0066)	FCB	%0000	23: WRITE
0038:	0E	(0067)	FCB	%1110	24: WRITE: LOAD C'(K)
0039:	08	(0068)	FCB	%1000	25: ADD: LOAD '1'
003A:	00	(0069)	FCB	%0000	26: ADD
003B:	0C	(0070)	FCB	%1100	27: WRITE: LOAD G'(K)
003C:	00	(0071)	FCB	%0000	28: ADD
003D:	08	(0072)	FCB	%1000	29: CONVERT: K=8, LOAD '1' IN R
003E:	00	(0073)	FCB	%0000	30: ADD
003F:	00	(0074)	FCB	%0000	31: WRITE
		(0075) *			
		(0076) *	16 KBPS CVSD PROGRAM		
		(0077) *			
0040:	00	(0078)	FCB	\$0	00
0041:	00	(0079)	FCB	\$0	01
0042:	00	(0080)	FCB	\$0	02
0043:	00	(0081)	FCB	\$0	03
0044:	00	(0082)	FCB	\$0	04
0045:	00	(0083)	FCB	\$0	05
0046:	00	(0084)	FCB	\$0	06
0047:	00	(0085)	FCB	\$0	07
0048:	00	(0086)	FCB	\$0	08
0049:	00	(0087)	FCB	\$0	09
004A:	09	(0088)	FCB	%1001	10: WRITE: LOAD S'(K-1)
004B:	0C	(0089)	FCB	%0000	11: ADD
004C:	00	(0090)	FCB	%0000	12: NOP
004D:	00	(0091)	FCB	%0000	13: NOP
004E:	0A	(0092)	FCB	%1010	14: WRITE: LOAD T(K)
004F:	08	(0093)	FCB	%1000	15: ADD: LOAD '1'
0050:	00	(0094)	FCB	\$0	16
0051:	00	(0095)	FCB	\$0	17
0052:	00	(0096)	FCB	\$0	18
0053:	00	(0097)	FCB	\$0	19
0054:	00	(0098)	FCB	\$0	20
0055:	00	(0099)	FCB	\$0	21
0056:	00	(0100)	FCB	\$0	22
0057:	00	(0101)	FCB	\$0	23
0058:	00	(0102)	FCB	\$0	24
0059:	00	(0103)	FCB	\$0	25
005A:	00	(0104)	FCB	%0000	26: ADD
005B:	0A	(0105)	FCB	%1010	27: WRITE: LOAD T(K)
005C:	04	(0106)	FCB	%0100	28: ADD: INITIATE STROBE
005D:	08	(0107)	FCB	%1000	29: WRITE: LOAD '1'
005E:	00	(0108)	FCB	%0000	30: ADD
005F:	00	(0109)	FCB	%0000	31: WRITE
		(0110) *			

TABLE 3-9. READ ADDRESS (Cont)

		(0111) *	16 KBPS ADM PROGRAM		
		(0112) *			
0060:	00	(0113)	FCB	\$0	00
0061:	00	(0114)	FCB	\$0	01
0062:	00	(0115)	FCB	\$0	02
0063:	00	(0116)	FCB	\$0	03
0064:	00	(0117)	FCB	\$0	04
0065:	00	(0118)	FCB	\$0	05
0066:	00	(0119)	FCB	\$0	06
0067:	00	(0120)	FCB	\$0	07
0068:	00	(0121)	FCB	\$0	08
0069:	00	(0122)	FCB	\$0	09
006A:	09	(0123)	FCB	%1001	10: WRITE: LOAD S'(K-1)
006B:	00	(0124)	FCB	%0000	11: ADD
006C:	00	(0125)	FCB	%0000	12: NOP
006D:	00	(0126)	FCB	%0000	13: NOP
006E:	0A	(0127)	FCB	%1010	14: WRITE: LOAD T(K)
006F:	00	(0128)	FCB	%0000	15: ADD
0070:	00	(0129)	FCB	\$0	16
0071:	00	(0130)	FCB	\$0	17
0072:	00	(0131)	FCB	\$0	18
0073:	00	(0132)	FCB	\$0	19
0074:	00	(0133)	FCB	\$0	20
0075:	00	(0134)	FCB	\$0	21
0076:	00	(0135)	FCB	\$0	22
0077:	00	(0136)	FCB	\$0	23
0078:	00	(0137)	FCB	\$0	24
0079:	00	(0138)	FCB	\$0	25
007A:	0C	(0139)	FCB	%1100	26: WRITE: LOAD G'(K)
007B:	08	(0140)	FCB	%1000	27: ADD: LOAD '1'
007C:	04	(0141)	FCB	%0100	28: ADD: INITIATE STROBE
007D:	00	(0142)	FCB	%0000	29: WRITE
007E:	07	(0143)	FCB	%0111	30: CONVERT: K=7
007F:	00	(0144)	FCB	%0000	31: WRITE
		(0145) *			
		(0146) *	9.6 KBPS ARC PROGRAM		
		(0147) *			
0080:	08	(0148)	FCB	%1011	00: WRITE: LOAD S'(K-3)
0081:	0D	(0149)	FCB	%1101	01: ADD: LOAD S'(K-2)
0082:	08	(0150)	FCB	%1000	02: ADD: LOAD S'(K-1)
0083:	00	(0151)	FCB	%0000	03: ADD
0084:	00	(0152)	FCB	%0000	04: NOP
0085:	00	(0153)	FCB	%0000	05: NOP
0086:	0A	(0154)	FCB	%1010	06: WRITE: LOAD T(K)
0087:	00	(0155)	FCB	%0000	07: ADD
0088:	00	(0156)	FCB	%0000	08: NOP
0089:	00	(0157)	FCB	%0000	09: NOP
008A:	04	(0158)	FCB	%1010	10: WRITE: LOAD T(K)
008B:	00	(0159)	FCB	%0000	11: ADD
008C:	00	(0160)	FCB	%0000	12: NOP
008D:	00	(0161)	FCB	%0000	13: NOP
008E:	0A	(0162)	FCB	%1010	14: WRITE: LOAD T(K)
008F:	08	(0163)	FCB	%1000	15: ADD: LOAD S'(K-1)
0090:	00	(0164)	FCB	%0000	16: WRITE
0091:	0D	(0165)	FCB	%1101	17: ADD: LOAD S'(K-2)
0092:	00	(0166)	FCB	%0000	18: WRITE
0093:	04	(0167)	FCB	%0100	19: ADD: INITIATE ENCODING /DECODING AND STROBE

TABLE 3-9. READ ADDRESS (Cont)

0094:	0C	(0168)	FCB	%1100	20: WRITE: LOAD G'(K)
0095:	09	(0169)	FCB	%1001	21: ADD: LOAD '1'
0096:	00	(0170)	FCB	%0000	22: ADD
0097:	00	(0171)	FCB	%0000	23: WRITE
0098:	0E	(0172)	FCB	%1110	24: WRITE: LOAD C'(K)
0099:	09	(0173)	FCB	%1001	25: ADD: LOAD '1'
009A:	00	(0174)	FCB	%0000	26: ADD
009B:	0C	(0175)	FCB	%1100	27: WRITE: LOAD G'(K)
009C:	00	(0176)	FCB	%0000	28: ADD
009D:	09	(0177)	FCB	%1001	29: CONVERT: K=9, LOAD '1' IN R
009E:	00	(0178)	FCB	%0000	30: ADD
009F:	00	(0179)	FCB	%0000	31: WRITE
		(0180)	*		
		(0181)	*	UNSPECIFIED	
		(0182)	*		
00A0:	00	(0183)	FCB	\$0	00
00A1:	00	(0184)	FCB	\$0	01
00A2:	00	(0185)	FCB	\$0	02
00A3:	00	(0186)	FCB	\$0	03
00A4:	00	(0187)	FCB	\$0	04
00A5:	00	(0188)	FCB	\$0	05
00A6:	00	(0189)	FCB	\$0	06
00A7:	00	(0190)	FCB	\$0	07
00A8:	00	(0191)	FCB	\$0	08
00A9:	00	(0192)	FCB	\$0	09
00AA:	00	(0193)	FCB	\$0	10
00AB:	00	(0194)	FCB	\$0	11
00AC:	00	(0195)	FCB	\$0	12
00AD:	00	(0196)	FCB	\$0	13
00AE:	00	(0197)	FCB	\$0	14
00AF:	00	(0198)	FCB	\$0	15
00B0:	00	(0199)	FCB	\$0	16
00B1:	00	(0200)	FCB	\$0	17
00B2:	00	(0201)	FCB	\$0	18
00B3:	00	(0202)	FCB	\$0	19
00B4:	00	(0203)	FCB	\$0	20
00B5:	00	(0204)	FCB	\$0	21
00B6:	00	(0205)	FCB	\$0	22
00B7:	00	(0206)	FCB	\$0	23
00B8:	00	(0207)	FCB	\$0	24
00B9:	00	(0208)	FCB	\$0	25
00BA:	00	(0209)	FCB	\$0	26
00BB:	00	(0210)	FCB	\$0	27
00BC:	00	(0211)	FCB	\$0	28
00BD:	00	(0212)	FCB	\$0	29
00BE:	00	(0213)	FCB	\$0	30
00BF:	00	(0214)	FCB	\$0	31
		(0215)	*		
		(0216)	*	9.6 KBPS CVSD PROGRAM	
		(0217)	*		
00C0:	00	(0218)	FCB	\$0	00
00C1:	00	(0219)	FCB	\$0	01
00C2:	00	(0220)	FCB	\$0	02
00C3:	00	(0221)	FCB	\$0	03
00C4:	00	(0222)	FCB	\$0	04
00C5:	00	(0223)	FCB	\$0	05
00C6:	00	(0224)	FCB	%0000	06: NOP
00C7:	00	(0225)	FCB	%0000	07: NOP

TABLE 3-9. READ ADDRESS (Cont)

00C8:	00	(0226)	FCB	%0000	08: NOP
00C9:	00	(0227)	FCB	%0000	09: NOP
00CA:	09	(0228)	FCB	%1001	10: WRITE: LOAD S*(K-1)
00CB:	00	(0229)	FCB	%0000	11: ADD
00CC:	00	(0230)	FCB	%0000	12: NOP
00CD:	00	(0231)	FCB	%0000	13: NOP
00CE:	0A	(0232)	FCB	%1010	14: WRITE: LOAD T(K)
00CF:	08	(0233)	FCB	%1000	15: ADD: LOAD '1'
00D0:	00	(0234)	FCB	\$0	16
00D1:	00	(0235)	FCB	\$0	17
00D2:	00	(0236)	FCB	\$0	18
00D3:	00	(0237)	FCB	\$0	19
00D4:	00	(0238)	FCB	\$0	20
00D5:	00	(0239)	FCB	\$0	21
00D6:	03	(0240)	FCB	%0000	22: ADD
00D7:	00	(0241)	FCB	%0000	23: WRITE
00D8:	0A	(0242)	FCB	%1010	24: WRITE: LOAD T(K)
00D9:	04	(0243)	FCB	%0100	25: ADD: INITIATE STROBE
00DA:	08	(0244)	FCB	%1000	26: WRITE: LOAD '1'
00DB:	00	(0245)	FCB	%0000	27: ADD
00DC:	00	(0246)	FCB	%0000	28: WRITE
00DD:	00	(0247)	FCB	%0000	29: NOP
00DE:	00	(0248)	FCB	%0000	30: NOP
00DF:	00	(0249)	FCB	%0000	31: NOP
		(0250)	*		
		(0251)	* 9.6 KBPS ADM PROGRAM		
		(0252)	*		
00E0:	00	(0253)	FCB	\$0	00
00E1:	00	(0254)	FCB	\$0	01
00E2:	00	(0255)	FCB	\$0	02
00E3:	00	(0256)	FCB	\$0	03
00E4:	00	(0257)	FCB	\$0	04
00E5:	00	(0258)	FCB	\$0	05
00E6:	08	(0259)	FCB	%1011	06: WRITE: LOAD S*(K-3)
00E7:	00	(0260)	FCB	%1101	07: ADD: LOAD S*(K-2)
00E8:	03	(0261)	FCB	%1001	08: ADD: LOAD S*(K-1)
00E9:	00	(0262)	FCB	%0000	09: ADD
00EA:	00	(0263)	FCB	%0000	10: NOP
00EB:	00	(0264)	FCB	%0000	11: NOP
00EC:	0A	(0265)	FCB	%1010	12: WRITE: LOAD T(K)
00ED:	09	(0266)	FCB	%1001	13: ADD: LOAD S*(K-1)
00EE:	00	(0267)	FCB	%0000	14: WRITE
00EF:	0D	(0268)	FCB	%1101	15: ADD: LOAD S*(K-2)
00F0:	00	(0269)	FCB	\$0	16
00F1:	00	(0270)	FCB	\$0	17
00F2:	00	(0271)	FCB	\$0	18
00F3:	00	(0272)	FCB	\$0	19
00F4:	00	(0273)	FCB	\$0	20
00F5:	00	(0274)	FCB	\$0	21
00F6:	00	(0275)	FCB	%0000	22: WRITE
00F7:	00	(0276)	FCB	%0000	23: ADD
00F8:	0C	(0277)	FCB	%1100	24: WRITE: LOAD G*(K)
00F9:	08	(0278)	FCB	%1000	25: ADD: LOAD '1'
00FA:	04	(0279)	FCB	%0100	26: ADD: INITIATE STROBE
00FB:	00	(0280)	FCB	%0000	27: WRITE
00FC:	07	(0281)	FCB	%0111	28: CONVERT: K=7
00FD:	00	(0282)	FCB	%0000	29: WRITE
00FE:	00	(0283)	FCB	%0000	30: NOP
00FF:	00	(0284)	FCB	%0000	31: NOP
	0100	(0285)	END		

3.4.2 PROGRAM AND PARAMETER ROM's

The schematic for the program and parameter ROM's is given in Figure 3-10. As mentioned above, the program ROM (74S471, J1) and the write- (74S287, A13) and read-address (74S287, A12) ROM's are all addressed by the mode-control signals M0, M1, M2 and the program count W5,...W1. The outputs of the two 256x4 address ROM's are tied together to provide the scratchpad address bits P10, P11, P12 and a control bit P9. The write-address ROM is enabled during the first half of each word time and the read-address ROM is enabled for the last half of the word time. The mode control signals M0, M1, M2 and the control signals P3, P7 and P8 form the first 6 address bits of the parameter ROM. The last two address bits (G, H) are given as follows:

	M1	P3	G	H
ARC	$\left\{ \begin{array}{l} 0 \\ 0 \end{array} \right.$	0	Q ₁ ,	Q ₂
		1	P4,	0
CVSD/ADM	$\left\{ \begin{array}{l} 1 \\ 1 \end{array} \right.$	0	BK1,	BK2
		1	P4,	0

Clearly, the parameters are independent of the quantizer level (present and two previous quantizer levels in the case of CVSD/ADM) when P3=1. Table 3-6 shows the address assignments for different parameters.

The 8-bit output of the parameter ROM gives the 'multiplier' $\pm 2^{-n1} \pm 2^{-n2}$. This 'multiplier' is used during the ADD instruction to perform the operation

$$A \pm (\pm 2^{-n1} \pm 2^{-n2}) R.$$

The sign of the 'multiplier' is dependent on the sign of the quantizer level (QS/BS) when the program control signal P5=1 in the ADD instruction.

The rules for coding the parameters are specified in Table 3-10, and a listing of the set of parameters for each mode appears in Table 3-11.

TABLE 3-10

CODING OF PARAMETER VALUES

1. 8 bits represent $\pm 2^{-n_1} \pm 2^{-n_2}$
2. First 4 bits give $\pm 2^{-n_1}$ and the last 4 give $\pm 2^{-n_2}$
3. The sign bit is determined by the first bit of each hexadecimal digit: 0 is +, 1 is - .
4. n_1 and n_2 are coded as follows:

	n_1	n_2
000	0	∞
001	1	8
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	∞	7

TABLE 3-11. PARAMETERS

(0001) * PARAMETERS		
(0002) * THIS LISTING SPECIFIES THE CONTENTS OF THE PARAMETER ROM		
(0003) *		
(0004) * 16 KBPS ARC PARAMETERS		
(0005) *		
0000: 00	(0006)	FCB \$00 00: UNUSED
0001: 15	(0007)	FCB %00010101 01: $A(2) = 1/2 + 1/32 = 17/32$
0002: 35	(0008)	FCB %00110101 02: $A(1) = 1/8 + 1/32 = 5/32$
0003: 36	(0009)	FCB %00000110 03: $A(3) = 1 + 1/64 = 65/64$
0004: 70	(0010)	FCB %01110000 04: $L(0) = 0 + 0 = 0$
0005: 24	(0011)	FCB %00100100 05: $L(1) = 1/4 + 1/16 = 5/16$
0006: 12	(0012)	FCB %00010010 06: $L(2) = 1/2 + 1/4 = 3/4$
0007: 02	(0013)	FCB %00000010 07: $L(3) = 1 + 1/4 = 5/4$
0008: 07	(0014)	FCB %11000111 08: $F1(0) = -1/16 + 1/128 = -7/128$
0009: 77	(0015)	FCB %01110111 09: $F1(1) = 0 + 1/128 = 1/128$
000A: 46	(0016)	FCB %01000110 10: $F1(2) = 1/16 + 1/64 = 5/64$
000B: 20	(0017)	FCB %00100000 11: $F1(3) = 1/4 + 0 = 1/4$
000C: 70	(0018)	FCB %01110000 12: $F2(0) = 0 + 0 = 0$
000D: 70	(0019)	FCB %01110000 13: $F2(1) = 0 + 0 = 0$
000E: 70	(0020)	FCB %01110000 14: $F2(2) = 0 + 0 = 0$
000F: 09	(0021)	FCB %00001011 15: $F2(3) = 1 - 1/8 = 7/8$
0010: 50	(0022)	FCB %10011100 16: $B2 = -1/2 - 1/16 = -9/16$
0011: 00	(0023)	FCB \$00 17: UNUSED
0012: 12	(0024)	FCB %00010010 18: $(1 - 2^{**}(-M2)) =$ $1/2 + 1/4 = 3/4$
0013: 00	(0025)	FCB \$00 19: UNUSED
0014: 03	(0026)	FCB %00000011 20: $B1 = 1 + 1/8 = 9/8$
0015: 00	(0027)	FCB \$00 21: UNUSED
0016: 00	(0028)	FCB %00000000 22: $ONE = 1 + 0 = 1$
0017: 00	(0029)	FCB \$00 23: UNUSED
0018: 34	(0030)	FCB %00110100 24: $B3 = 1/8 + 1/16 = 3/16$
0019: 00	(0031)	FCB \$00 25: UNUSED
001A: 00	(0032)	FCB \$00 26: UNUSED
001B: 00	(0033)	FCB \$00 27: UNUSED
001C: 50	(0034)	FCB %01010000 28: $TMIN = 1/32 + 0 = 1/32$
001D: 00	(0035)	FCB \$00 29: UNUSED
001E: 0F	(0036)	FCB %00001111 30: $(1 - 2^{**}(-M1)) = 1 - 1/128$ $= 127/128$
001F: 00	(0037)	FCB \$00 31: UNUSED
(0038) *		
(0039) * 14.4 KBPS ARC PARAMETERS		
(0040) *		
0020: 00	(0041)	FCB \$00 00: UNUSED
0021: 15	(0042)	FCB %00010101 01: $A(2) = 1/2 + 1/32 = 17/32$
0022: 35	(0043)	FCB %00110101 02: $A(1) = 1/8 + 1/32 = 5/32$
0023: 36	(0044)	FCB %00000110 03: $A(3) = 1 + 1/64 = 65/64$
0024: 70	(0045)	FCB %01110000 04: $L(0) = 0 + 0 = 0$
0025: 24	(0046)	FCB %00100100 05: $L(1) = 1/4 + 1/16 = 5/16$
0026: 12	(0047)	FCB %00010010 06: $L(2) = 1/2 + 1/4 = 3/4$
0027: 02	(0048)	FCB %00000010 07: $L(3) = 1 + 1/4 = 5/4$
0028: 07	(0049)	FCB %11000111 08: $F1(0) = -1/16 + 1/128 = -7/128$
0029: 76	(0050)	FCB %01110110 09: $F1(1) = 0 + 1/64 = 1/64$
002A: 3E	(0051)	FCB %00111110 10: $F1(2) = 1/8 - 1/64 = 7/64$
002B: 24	(0052)	FCB %00100100 11: $F1(3) = 1/4 + 1/16 = 5/16$
002C: 70	(0053)	FCB %01110000 12: $F2(0) = 0 + 0 = 0$
002D: 70	(0054)	FCB %01110000 13: $F2(1) = 0 + 0 = 0$

TABLE 3-11. PARAMETERS (Cont)

002E:	70	(0055)	FCB	%01110000	14: $F2(2) = 0 + 0 = 0$
002F:	0C	(0056)	FCB	%00001100	15: $F2(3) = 1 - 1/16 = 15/16$
0030:	9C	(0057)	FCB	%10011100	16: $B2 = -1/2 - 1/16 = -9/16$
0031:	00	(0058)	FCB	\$00	17: UNUSED
0032:	12	(0059)	FCB	%00010010	18: $(1 - 2^{**}(-12)) =$ $1/2 + 1/4 = 3/4$
0033:	00	(0060)	FCB	\$00	19: UNUSED
0034:	03	(0061)	FCB	%00000011	20: $B1 = 1 + 1/8 = 9/8$
0035:	00	(0062)	FCB	\$00	21: UNUSED
0036:	00	(0063)	FCB	%00000000	22: $ONE = 1 + 0 = 1$
0037:	00	(0064)	FCB	\$00	23: UNUSED
0038:	34	(0065)	FCB	%00110100	24: $B3 = 1/8 + 1/16 = 3/16$
0039:	00	(0066)	FCB	\$00	25: UNUSED
003A:	00	(0067)	FCB	\$00	26: UNUSED
003B:	00	(0068)	FCB	\$00	27: UNUSED
003C:	50	(0069)	FCB	%01010000	28: $THIN' = 1/32 + 0 = 1/32$
003D:	00	(0070)	FCB	\$00	29: UNUSED
003E:	0F	(0071)	FCB	%00001111	30: $(1 - 2^{**}(-11)) =$ $1 - 1/128 = 127/128$
003F:	00	(0072)	FCB	\$00	31: UNUSED
		(0073) *			
		(0074) *	16 KBPS CVSD PARAMETERS		
		(0075) *			
0040:	00	(0076)	FCB	\$00	00: UNUSED
0041:	00	(0077)	FCB	\$00	01: UNUSED
0042:	00	(0078)	FCB	\$00	02: UNUSED
0043:	00	(0079)	FCB	\$00	03: UNUSED
0044:	23	(0080)	FCB	%00100011	04: $L = 1/4 + 1/8 = 3/8$
0045:	23	(0081)	FCB	%00100011	05: $L = 1/4 + 1/8 = 3/8$
0046:	23	(0082)	FCB	%00100011	06: $L = 1/4 + 1/8 = 3/8$
0047:	23	(0083)	FCB	%00100011	07: $L = 1/4 + 1/8 = 3/8$
0048:	2D	(0084)	FCB	%00101101	08: $F(0) = 1/4 - 1/32 = 7/32$
0049:	70	(0085)	FCB	%01110000	09: $F(1) = 0 + 0 = 0$
004A:	70	(0086)	FCB	%01110000	10: $F(2) = 0 + 0 = 0$
004B:	70	(0087)	FCB	%01110000	11: $F(3) = 0 + 0 = 0$
004C:	00	(0088)	FCB	\$00	12: UNUSED
004D:	00	(0089)	FCB	\$00	13: UNUSED
004E:	00	(0090)	FCB	\$00	14: UNUSED
004F:	00	(0091)	FCB	\$00	15: UNUSED
0050:	00	(0092)	FCB	\$00	16: UNUSED
0051:	00	(0093)	FCB	\$00	17: UNUSED
0052:	00	(0094)	FCB	%00001101	18: $(1 - 2^{**}(-12)) =$ $1 - 1/32 = 31/32$
0053:	00	(0095)	FCB	\$00	19: UNUSED
0054:	00	(0096)	FCB	\$00	20: UNUSED
0055:	00	(0097)	FCB	\$00	21: UNUSED
0056:	07	(0098)	FCB	%00000000	22: $ONE = 1 + 0 = 1$
0057:	00	(0099)	FCB	\$00	23: UNUSED
0058:	00	(0100)	FCB	\$00	24: UNUSED
0059:	00	(0101)	FCB	\$00	25: UNUSED
005A:	0F	(0102)	FCB	%00001111	26: $B1 = 1 - 1/128 = 127/128$
005B:	00	(0103)	FCB	\$00	27: UNUSED
005C:	00	(0104)	FCB	\$00	28: UNUSED
005D:	00	(0105)	FCB	\$00	29: UNUSED
005E:	00	(0106)	FCB	%11010000	30: $THIN = -1/32 + 0 = -1/32$
005F:	00	(0107)	FCB	\$00	31: UNUSED
		(0108) *			

TABLE 3-11. PARAMETERS (Cont)

		(0109) * 16 KBPS ADM PARAMETERS			
		(0110) *			
0060:	00	(0111)	FCB	\$00	00: UNUSED
0061:	00	(0112)	FCB	\$00	01: UNUSED
0062:	00	(0113)	FCB	\$00	02: UNUSED
0063:	00	(0114)	FCB	\$00	03: UNUSED
0064:	12	(0115)	FCB	200010010	04: $L = 1/2 + 1/4 = 3/4$
0065:	12	(0116)	FCB	200010010	05: $L = 1/2 + 1/4 = 3/4$
0066:	12	(0117)	FCB	200010010	06: $L = 1/2 + 1/4 = 3/4$
0067:	12	(0118)	FCB	200010010	07: $L = 1/2 + 1/4 = 3/4$
0068:	34	(0119)	FCB	200110100	08: $F(0) = 1/8 + 1/16 = 3/16$
0069:	46	(0120)	FCB	201000110	09: $F(1) = 1/16 + 1/64 = 5/64$
006A:	DF	(0121)	FCB	211011111	10: $F(2) = -1/32 - 1/128 = -5/128$
006B:	70	(0122)	FCB	201110000	11: $F(3) = 0 + 0 = 0$
006C:	00	(0123)	FCB	\$00	12: UNUSED
006D:	00	(0124)	FCB	\$00	13: UNUSED
006E:	00	(0125)	FCB	\$00	14: UNUSED
006F:	00	(0126)	FCB	\$00	15: UNUSED
0070:	00	(0127)	FCB	\$00	16: UNUSED
0071:	00	(0128)	FCB	\$00	17: UNUSED
0072:	00	(0129)	FCB	\$00	18: UNUSED
0073:	00	(0130)	FCB	\$00	19: UNUSED
0074:	8F	(0131)	FCB	200001111	20: $B1 = 1 - 1/128 = 127/128$
0075:	00	(0132)	FCB	\$00	21: UNUSED
0076:	00	(0133)	FCB	200000000	22: $ONE = 1 + 0 = 1$
0077:	00	(0134)	FCB	\$00	23: UNUSED
0078:	00	(0135)	FCB	\$00	24: UNUSED
0079:	00	(0136)	FCB	\$00	25: UNUSED
007A:	00	(0137)	FCB	\$00	26: UNUSED
007B:	00	(0138)	FCB	\$00	27: UNUSED
007C:	00	(0139)	FCB	\$00	28: UNUSED
007D:	00	(0140)	FCB	\$00	29: UNUSED
007E:	0F	(0141)	FCB	200001111	30: $(1 - 2^{**}(-M1)) =$ $1 - 1/128 = 127/128$
007F:	00	(0142)	FCB	\$00	31: UNUSED
		(0143) *			
		(0144) * 9.6 KBPS ARC PARAMETERS			
		(0145) *			
0080:	00	(0146)	FCB	\$00	00: UNUSED
0081:	03	(0147)	FCB	200000011	01: $A(2) = -1 + 1/8 = 9/8$
0082:	20	(0148)	FCB	200100000	02: $A(1) = 1/4 + 0 = 1/4$
0083:	03	(0149)	FCB	200000011	03: $A(3) = A(2) = 9/8$
0084:	70	(0150)	FCB	201110000	04: $L(0) = 0 + 0 = 0$
0085:	10	(0151)	FCB	200010000	05: $L(1) = 1/2 + 0 = 1/2$
0086:	02	(0152)	FCB	200000010	06: $L(2) = 1 + 1/4 = 5/4$
0087:	02	(0153)	FCB	200000010	07: $L(3) = L(2) = 5/4$
0088:	07	(0154)	FCB	211000111	08: $F1(0) = -1/16 + 1/128 = -7/128$
0089:	3E	(0155)	FCB	200111110	09: $F1(1) = 1/8 - 1/64 = 7/64$
0090:	2E	(0156)	FCB	200101110	10: $F1(2) = 1/4 - 1/64 = 15/64$
0091:	2E	(0157)	FCB	200101110	11: $F1(3) = F1(2) = 15/64$
0092:	70	(0158)	FCB	201110000	12: $F2(0) = 0 + 0 = 0$
0093:	70	(0159)	FCB	201110000	13: $F2(1) = 0 + 0 = 0$
0094:	12	(0160)	FCB	200010010	14: $F2(2) = 1/2 + 1/4 = 3/4$
0095:	12	(0161)	FCB	200010010	15: $F2(3) = F2(2) = 3/4$
0096:	94	(0162)	FCB	210010100	16: $B2 = -1/2 + 1/16 = -7/16$
0097:	00	(0163)	FCB	\$00	17: UNUSED
0098:	12	(0164)	FCB	200010010	18: $(1 - 2^{**}(-M2)) = 1/2 + 1/4 =$ $3/4$
0099:	00	(0165)	FCB	\$00	19: UNUSED

TABLE 3-11. PARAMETERS (Cont)

0084:	00	(0166)	FCB	%00000000	20: B1 = 1 + 0 = 1
0085:	00	(0167)	FCB	\$00	21: UNUSED
0086:	00	(0168)	FCB	%00000000	22: ONE = 1 + 0 = 1
0087:	00	(0169)	FCB	\$00	23: UNUSED
0088:	45	(0170)	FCB	%01000101	24: B3 = 1/16 + 1/32 = 3/32
0089:	00	(0171)	FCB	\$00	25: UNUSED
008A:	00	(0172)	FCB	\$00	26: UNUSED
008B:	00	(0173)	FCB	\$00	27: UNUSED
008C:	50	(0174)	FCB	%01000000	28: TMIN' = 1/64 + 0 = 1/64
008D:	00	(0175)	FCB	\$00	29: UNUSED
008E:	0F	(0176)	FCB	%00001111	30: (1 - 2**(-M1)) = 1 - 1/128 = 127/128
008F:	00	(0177)	FCB	\$00	31: UNUSED
		(0178) *			
		(0179) *	UNSPECIFIED		
		(0180) *			
00A0:	00	(0181)	FCB	\$00	00
00A1:	00	(0182)	FCB	\$00	01
00A2:	00	(0183)	FCB	\$00	02
00A3:	00	(0184)	FCB	\$00	03
00A4:	00	(0185)	FCB	\$00	04
00A5:	00	(0186)	FCB	\$00	05
00A6:	00	(0187)	FCB	\$00	06
00A7:	00	(0188)	FCB	\$00	07
00A8:	00	(0189)	FCB	\$00	08
00A9:	00	(0190)	FCB	\$00	09
00AA:	00	(0191)	FCB	\$00	10
00AB:	00	(0192)	FCB	\$00	11
00AC:	00	(0193)	FCB	\$00	12
00AD:	00	(0194)	FCB	\$00	13
00AE:	00	(0195)	FCB	\$00	14
00AF:	00	(0196)	FCB	\$00	15
00B0:	00	(0197)	FCB	\$00	16
00B1:	00	(0198)	FCB	\$00	17
00B2:	00	(0199)	FCB	\$00	18
00B3:	00	(0200)	FCB	\$00	19
00B4:	00	(0201)	FCB	\$00	20
00B5:	00	(0202)	FCB	\$00	21
00B6:	00	(0203)	FCB	\$00	22
00B7:	00	(0204)	FCB	\$00	23
00B8:	00	(0205)	FCB	\$00	24
00B9:	00	(0206)	FCB	\$00	25
00BA:	00	(0207)	FCB	\$00	26
00BB:	00	(0208)	FCB	\$00	27
00BC:	00	(0209)	FCB	\$00	28
00BD:	00	(0210)	FCB	\$00	29
00BE:	00	(0211)	FCB	\$00	30
00BF:	00	(0212)	FCB	\$00	31
		(0213) *			
		(0214) *	9.6 Kbps CVSD PARAMETERS		
		(0215) *			
00C0:	00	(0216)	FCB	\$00	00: UNUSED
00C1:	00	(0217)	FCB	\$00	01: UNUSED
00C2:	00	(0218)	FCB	\$00	02: UNUSED
00C3:	00	(0219)	FCB	\$00	03: UNUSED
00C4:	23	(0220)	FCB	%00100011	04: L = 1/4 + 1/8 = 3/8
00C5:	23	(0221)	FCB	%00100011	05: L = 1/4 + 1/8 = 3/8
00C6:	23	(0222)	FCB	%00100011	06: L = 1/4 + 1/8 = 3/8
00C7:	23	(0223)	FCB	%00100011	07: L = 1/4 + 1/8 = 3/8
00C8:	20	(0224)	FCB	%00101101	08: F(0) = 1/4 - 1/32 = 7/32

TABLE 3-11. PARAMETERS (Cont)

0009:	70	(0235)	FCB	X01110000	09: F(1) = 0 + 0 = 0
000A:	70	(0226)	FCB	X01110000	10: F(2) = 0 + 0 = 0
000B:	70	(0227)	FCB	X01110000	11: F(3) = 0 + 0 = 0
000C:	00	(0228)	FCB	\$00	12: UNUSED
000D:	00	(0229)	FCB	\$00	13: UNUSED
000E:	00	(0230)	FCB	\$00	14: UNUSED
000F:	00	(0231)	FCB	\$00	15: UNUSED
0010:	00	(0232)	FCB	\$00	16: UNUSED
0011:	00	(0233)	FCB	\$00	17: UNUSED
0012:	00	(0234)	FCB	X00001101	18: $(1 - 2^{*(-12)}) =$ $1 - 1/32 = 31/32$
0013:	00	(0235)	FCB	\$00	19: UNUSED
0014:	00	(0236)	FCB	\$00	20: UNUSED
0015:	00	(0237)	FCB	\$00	21: UNUSED
0016:	00	(0238)	FCB	X00000000	22: ONE = 1 + 0 = 1
0017:	00	(0239)	FCB	\$00	23: UNUSED
0018:	00	(0240)	FCB	\$00	24: UNUSED
0019:	00	(0241)	FCB	\$00	25: UNUSED
001A:	0F	(0242)	FCB	X00001111	26: B1 = 1 - 1/128 = 127/128
001B:	00	(0243)	FCB	\$00	27: UNUSED
001C:	00	(0244)	FCB	\$00	28: UNUSED
001D:	00	(0245)	FCB	\$00	29: UNUSED
001E:	10	(0246)	FCB	X11010000	30: TMIN = -1/32 + 0 = -1/32
001F:	00	(0247)	FCB	\$00	31: UNUSED
		(0248) *			
		(0249) *	9.6 KBPS ADM PARAMETERS		
		(0250) *			
00E0:	00	(0251)	FCB	\$00	00: UNUSED
00E1:	00	(0252)	FCB	\$00	01: UNUSED
00E2:	00	(0253)	FCB	\$00	02: UNUSED
00E3:	00	(0254)	FCB	\$00	03: UNUSED
00E4:	12	(0255)	FCB	X00010010	04: L = 1/2 + 1/4 = 3/4
00E5:	12	(0256)	FCB	X00010010	05: L = 1/2 + 1/4 = 3/4
00E6:	12	(0257)	FCB	X00010010	06: L = 1/2 + 1/4 = 3/4
00E7:	12	(0258)	FCB	X00010010	07: L = 1/2 + 1/4 = 3/4
00E8:	24	(0259)	FCB	X00100100	08: F(8) = 1/4 + 1/16 = 5/16
00E9:	30	(0260)	FCB	X00110000	09: F(1) = 1/8 + 0 = 1/8
00EA:	00	(0261)	FCB	X11000000	10: F(2) = -1/16 + 0 = -1/16
00EB:	70	(0262)	FCB	X01110000	11: F(3) = 0 + 0 = 0
00EC:	00	(0263)	FCB	\$00	12: UNUSED
00ED:	00	(0264)	FCB	\$00	13: UNUSED
00EE:	00	(0265)	FCB	\$00	14: UNUSED
00EF:	00	(0266)	FCB	\$00	15: UNUSED
00F0:	94	(0267)	FCB	X10010100	16: B2 = -1/2 + 1/16 = -7/16
00F1:	00	(0268)	FCB	\$00	17: UNUSED
00F2:	00	(0269)	FCB	\$00	18: UNUSED
00F3:	00	(0270)	FCB	\$00	19: UNUSED
00F4:	04	(0271)	FCB	X00000100	20: B1 = 1 + 1/16 = 17/16
00F5:	00	(0272)	FCB	\$00	21: UNUSED
00F6:	00	(0273)	FCB	X00000000	22: ONE = 1 + 0 = 1
00F7:	00	(0274)	FCB	\$00	23: UNUSED
00F8:	45	(0275)	FCB	X01000101	24: B3 = 1/16 + 1/32 = 3/32
00F9:	00	(0276)	FCB	\$00	25: UNUSED
00FA:	00	(0277)	FCB	\$00	26: UNUSED
00FB:	00	(0278)	FCB	\$00	27: UNUSED
00FC:	00	(0279)	FCB	\$00	28: UNUSED
00FD:	00	(0280)	FCB	\$00	29: UNUSED
00FE:	0F	(0281)	FCB	X00001111	30: $(1 - 2^{*(-11)}) =$ $1 - 1/128 = 127/128$
00FF:	00	(0282)	FCB	\$00	31: UNUSED
	0100	(0283)	END		

3.4.3 'A' REGISTER AND CONVERT CIRCUITRY

The 'A' register consists of three 4-bit universal shift registers (3x74LS194 A17, A16 and A21) which are clocked by \overline{HCL} during the last 12 bit times of a word (see Figure 3-11). During the ADD instruction CVTLD=0 (see Figure 3-13), P1=0 and CARRY=1 giving S1=0 and S0=1 which causes the A register to shift right serially with AIN=AOUT if P2=0 and AIN= Σ OUT if P2=1 (see Figure 3-13). During the WRITE instruction CVTLD=0, P1=1 and CARRY=1 giving S1=0 and S0=0 which means that the A register is normally stable. However, the A register may be cleared during bit times 3 and 4 if P5=T1=1 or during the last bit time if P3=1. Thus,

$$\overline{ACLEAR} = \overline{P1.P5.T1.BITS\ 3-4. P1.P3.WDCLK}$$

The parallel output of the A register A1-A12 is fed to the RAM scratchpad (Figure 3-12) with the 4th MSB inverted if P9=1 during the write time (first half, see timing diagram Figure 3-2).

The four MSB's of A are also fed to the CONVERT circuit where they are parallel added to P9-P12 during the CONVERT instruction (P1=1, P2=1). The output of the adder (74LS83, A22) is loaded into a 4-bit counter (74LS163, A23) on the 5th bit time of the CONVERT instruction (see timing diagram Figure 3-2). The carry (C_4) out of the adder is stored in a flip-flop (1/2 74LS74, A19) to indicate overflow during the addition of A1-A4 (4-bit integer part of $G'_k + C'_k$) and P9-P12 (the constant C which determines the minimum quantizer step size). As the adder output is parallel loaded into the counter (at bit time 5) the A register is parallel loaded (S0=1, S1=1) with 01, A5, ..., A12, 00, where A5, ..., A12 represents the fractional part of $G'_k + C'_k$. Thus, at bit time 5 of the CONVERT instruction the contents of the A register represent the integer

$$2^{10} (1 + G'_k + C'_k - \lfloor G'_k + C'_k \rfloor).$$

The A register is then shifted right n times where

$$n = 15 - ([G'_k + C'_k] + C)$$

resulting in

$$T'_k = 2^{[G'_k + C'_k] + C - 5} (1 + G'_k + C'_k - [G'_k + C'_k])$$

which is a piece-wise linear approximation of

$$2^{G'_k + C'_k + C - 5}$$

Note that if $[G'_k + C'_k] + C$ is greater than 15, the OVRFLO flipflop is set which prevents the A register from shifting right.

When $G'_k + C'_k = 0$, $T'_k = 2^{C-5}$. To permit more freedom in selecting, the minimum step size we further add T'_{min} to T'_k to get the quantizer step size T_k (see instructions 29, 30 and 31 of the ARC microprogram given in Table 3-7).

3.4.4 SCRATCHPAD, R REGISTER and MUXES

The 16x12 bit scratchpad RAM consists of 3 CMOS 16x4 random-access memories (3X74C89, locations A28, A29 and A30) with tristate output (see Figure 3-6). The RAM's are addressed by W6, P10, P11 and P12 (see Table 3-5). During the WRITE instruction (P1=1, P2=0) the scratchpad write enable signal (\overline{SPWE}) goes low during bit times 1 through 8 causing A1-A12 to be written into the desired location. If P9=1 bit A4 is inverted before it is written in the scratchpad.

The R register (locations B28, B29 and B30) is clocked by \overline{RCL} during the last 13 bit times of each word. In the ADD instruction P1=0, therefore S0=1 and S1=0 except at the last bit time (WDTIME) when S1=1 if P9=1. Thus, the contents of the R register are shifted right arithmetically (i.e., with sign extension) except at the last bit time when it may be parallel loaded from the scratchpad (if P9=1). During the WRITE and CONVERT instructions the R register is stable except if P9=1 which causes it to be parallel loaded from the scratchpad at the last bit time.

The shifted and/or negated outputs of the R register RN1 and RN2 are obtained through two 8 to 1 multiplexers (74LS151, locations B24 and B25) and two exclusive-or gates (1/2 74LS86, A18 and B23). The multiplexers have complemented outputs which compensates for the complemented outputs of the scratchpad. The coding rules for the parameters 1NS, 1N1, 1N2, 1N3 and 2NS, 2N1, 2N2, 2N3 are given in Table 3-10.

Note that during the ADD instruction the R register is shifted at the last 13 bit times instead of 12 to permit rounding during the addition of A, RN1 and RN2. The extra (13th bit) is stored in one of the bits of a 6-bit latch (74LS174, B19) shown in Figure 3-13.

To prevent overflow during formation of T_k (instruction 30 of the ARC in Table 3-7) the R register is cleared at bit times 3 and 4 if TSAT is high, i.e., if $\lfloor G'_k + C'_k \rfloor + C > 15$. Moreover, circuitry is provided to clear the R register at bit times 3 and 4 during the formation of the prediction p_k in the ARC mode (instructions 1, 2 and 3), subject to the value of the quantizer step size. Thus, if $\lfloor G'_k + C'_k \rfloor \leq 1$, TSMALL = $\overline{A1} \cdot \overline{A2} \cdot \overline{A3} \cdot \overline{M1}$ is high in the ARC mode causing TSMALLD to be high during the following transmit/receive cycle (see Figures 3-11 and 3-13). This causes RCLEAR to go low at the instruction times noted above resulting in zero prediction for low-level unvoiced sounds or background noise.

3.4.5 ADDER and T REGISTER

The serial addition $A + RN1 + RN2$ is performed in a dual one-bit full adder (74H183, B15). To perform rounding we force the bit before the LSB of A to a 1 (bit time 4). The initial carries for the two adders and the parameter sign bits 1NS and 2NS are stored in part of a 6-bit latch (74LS174, B19). Subsequent C_{in} 's are simply the previous C_{out} 's of the two adders. Further, the first R13 (at bit time 4) is 1 and subsequent R13's are the previous R12's (LSB of the R register).

The adder output is serially loaded into the 12-bit T register (74LS164 + 74LS195, B3 and B2) at all times by the clock \overline{HCL} . At the last bit time of an ADD

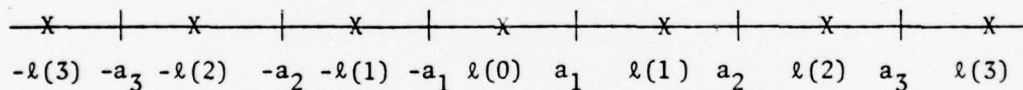
instruction Σ OUT and T1-T11 may be parallel loaded into the D/A buffer register (on the analog card) if P6 = 1.

3.4.6 Q REGISTER

The quantizer (Q) register and the associated logic shown in Figure 3-14 is used to generate a 3-bit number which represents the intermediate quantizer thresholds and the final quantizer level for the ARC algorithm. During the transmit cycle the source is the comparator output COMP which takes the values b_1 , b_2 and b_3 during the three successive tests. Then the Q register outputs Q_s , Q_1 , Q_2 take the following values at the last bit time of the following word times

Word time	Q_s	Q_1	Q_2
4	b_1	0	1
8	b_1	1	$\overline{b_2 \oplus b_1}$
12	b_1	$\overline{b_2 \oplus b_1}$	$\overline{b_3 \oplus b_1}$

If the quantizer has the following thresholds and levels



then they correspond to the following values of Q_s , Q_1 , Q_2

	Q_s	Q_1	Q_2
a_1	0	1	0
$-a_1$	1	1	0
$-a_2$	0	0	1
$-a_2$	1	0	1
a_3	0	1	1
$-a_3$	1	1	1
$\ell(0)$	0	0	0
$-\ell(0)$	1	0	0
$\ell(1)$	0	0	1
$-\ell(1)$	1	0	1
$\ell(2)$	0	1	0
$-\ell(2)$	1	1	0
$\ell(3)$	0	1	1
$-\ell(3)$	1	1	1

The quantizer clear circuitry is so arranged that upon transmit FIFO overflow (XFLAG=1) the Q register is cleared (forced to the zero level $\ell(0)$) only at 9.6kb/s. At 16 kb/s the input to the Q register (normally $b_2 \oplus b_1$) is inhibited (forced to zero) during word time 8 by using the program control bit P6=1. This restricts the quantizer to the inner three levels permitting the transmit FIFO to empty out. Seven level quantization resumes as soon as XFLAG goes low.

During the receive cycle the Q register is loaded directly with D1, D2, D3 from the decoder ROM. However, at 9.6 kb/s if a two-level codeword (11 110 corresponding to two $-\ell(1)$'s in a row) is decoded unloading of the receive FIFO is inhibited at the next sample interval and Q is loaded with D4, D5, D6. The 'second-word' indication is provided by the signal SECONDWD.

If the TEST input is grounded, the Q register will retain the transmitted quantizer level during the receive cycle. This permits the ARC as well as the CVSD/ADM modes to be tested in an internal loopback mode bypassing the encoder, TX FIFO, RX FIFO and the decoder.

3.5 ENCODER, DECODER, B REGISTER AND FIFO's

The encoder and decoder, used only in the ARC modes, implement the two codes given in Table 2-1 which are selected by rate control MO. In this section we describe the encoder, decoder, B register and transmit and receive FIFO circuits.

3.5.1 ENCODER

Note that every code word consists of a string of ones, which may or may not be preceded by a zero or trailed by a zero. The encoder is therefore built around a 32x8 bit ROM (82S123, B7) (see Figure 3-15). Regular code words are generated and loaded into the transmit FIFO during the transmit cycle. If the transmit FIFO empties out it is loaded with the filler code word during the receive cycle.

In the transmit cycle the ROM is addressed by MO, MINUS, Q_S, Q₁ and Q₂. The signal MINUS is always zero at 16 kb/s. At 9.6 kb/s MINUS = 1 if the previous quantizer level was '- $\ell(1)$ '; MINUS = 0 otherwise. A list of encoder ROM addresses appears in Table 3-12.

The first four bits E1-E4 of the encoder ROM output specify the codeword length as follows:

$$E1-E4 = 16-\ell_c$$

where ℓ_c is the length of the code word in bits. Two other ROM output bits E5 and E6 indicate whether the first bit of the code word is zero and whether the last bit is zero, respectively. In addition, the occurrence of level '- $\ell(1)$ ' at 9.6 kb/s causes a seventh ROM output, E7, to be 1, which sets a flipflop (1/2 74LS107, B4)

whose output is the MINUS signal.

Encoding is initiated in the transmit cycle by the microprogram when P9=0 and P10=1 during an ADD instruction (see also Figures 3-13 and 3-14). An extra encoding cycle to generate the filler codeword is initiated during the receive cycle if the transmit FIFO EMPTY signal is high. The stable state for the codeword length counter (74LS163,B12) is ECARRY=1. Thus when INITENCODE=1 during QTR3 the ENCODE flip-flop is set (see the encoder timing diagram Figure 3-10) on the negative going edge of QTR3. At the same instant the 4-bit ROM output E1-E4 is loaded into the codeword length counter (74LS163) resulting in ECARRY=0 unless E1-E4 = 15 (codeword length=1). The ENCODE flipflop is reset when INITENCODE=0 and ECARRY=1. The FIRSTBIT flipflop is set by INITENCODE and remains set for exactly one word time as shown in Figure 3-3. The last bit of the encoding cycle is indicated by CARRY=1. Thus,

$$\text{ENCODEDATA} = \overline{E_5} \text{FIRSTBIT} + E_6 \text{ CARRY.}$$

and

$$\text{ENCODECLK} = \text{ENCODE QTR2.}$$

Table 3-13 is a listing of the encoder ROM contents. The first column shows the ROM address and the second column the corresponding contents in hexadecimal notation.

TABLE 3-12
ENCODER ROM ADDRESS TABLE

A	B	C	D	E	
0	0	Q _s	Q ₁	Q ₂	8 codewords for 16 kb/s code
1	0	Q _s	Q ₁	Q ₂	9.6 kb/s code 1 (previous code word not -l(1))
1	1	Q _s	Q ₁	Q ₂	9.6 kb/s code 2 (previous code word was -l(1))
0	1	0	X	X	16 kb/s filler code word
0	1	1	X	X	9.6 kb/s filler code word

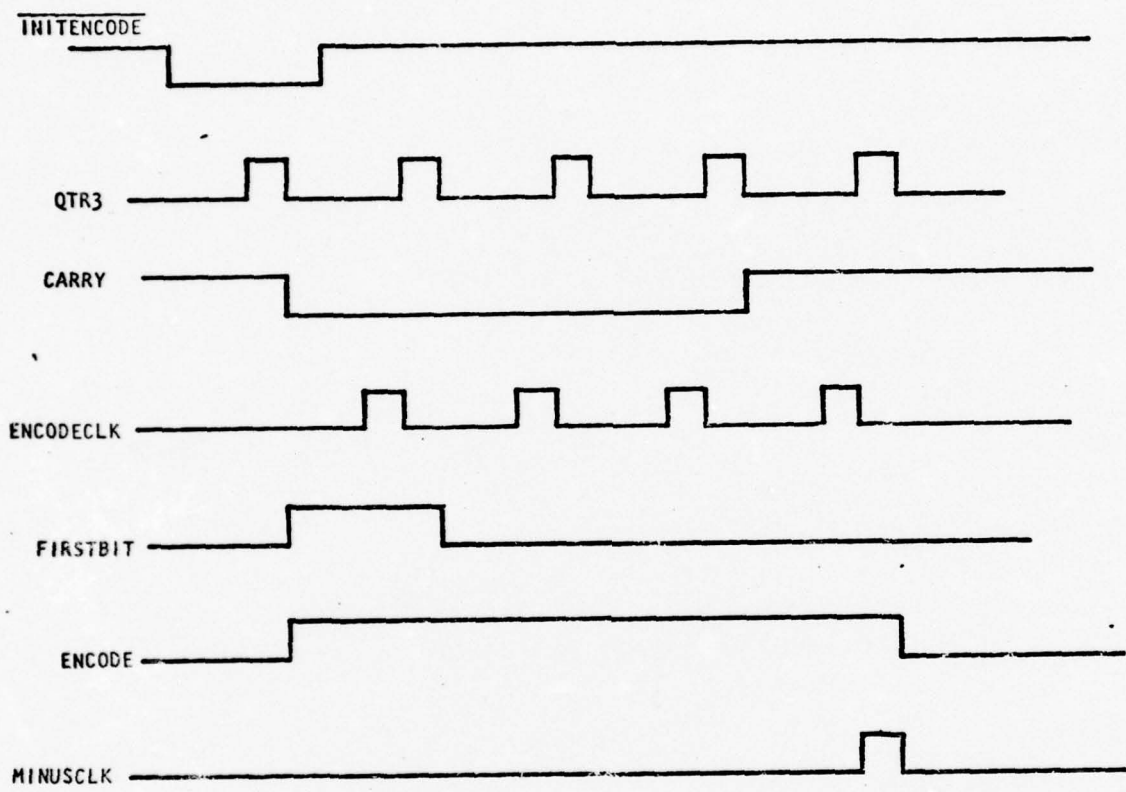


Figure 3-3. Encoder Timing

TABLE 3-13. ENCODER

```

(0001) * ENCODER
(0002) * THIS LISTING SPECIFIES THE CONTENTS OF THE ENCODER
(0003) * ROM
(0004) * -
(0005) * 16 KBPS CODE
(0006) *
0000: E4 (0007) FCB %11100100 00: CODEWORD FOR L(0) = 10
0001: EC (0008) FCB %11101100 01: CODEWORD FOR L(1) = 00
0002: CC (0009) FCB %11001100 02: CODEWORD FOR L(2) = 0110
0003: BC (0010) FCB %10111100 03: CODEWORD FOR L(3) = 01110
0004: E4 (0011) FCB %11100100 04: CODEWORD FOR -L(0) = 10
0005: E0 (0012) FCB %11100000 05: CODEWORD FOR -L(1) = 11
0006: DC (0013) FCB %11011100 06: CODEWORD FOR -L(2) = 010
0007: AC (0014) FCB %10101100 07: CODEWORD FOR -L(3) = 011110
(0015) *
(0016) * 16 KBPS FILLER CODE WORD
(0017) *
0008: A8 (0018) FCB %10101000 08: FILLER = 011111
0009: A8 (0019) FCB %10101000 09: FILLER = 011111
000A: A8 (0020) FCB %10101000 10: FILLER = 011111
000B: A8 (0021) FCB %10101000 11: FILLER = 011111
(0022) *
(0023) * 9.6 KBPS FILLER CODE WORD
(0024) *
000C: 70 (0025) FCB %01110000 12: FILLER = 11111111
000D: 70 (0026) FCB %01110000 13: FILLER = 11111111
000E: 70 (0027) FCB %01110000 14: FILLER = 11111111
000F: 70 (0028) FCB %01110000 15: FILLER = 11111111
(0029) *
(0030) * 9.6 KBPS CODE 1 (PREVIOUS LEVEL NOT -L(1))
(0031) *
0010: FC (0032) FCB %11111100 16: CODEWORD FOR L(0) = 0
0011: E4 (0033) FCB %11100100 17: CODEWORD FOR L(1) = 10
0012: A4 (0034) FCB %10100100 18: CODEWORD FOR L(2) = 111110
0013: A4 (0035) FCB %10100100 19: CODEWORD FOR L(3) = 111110
0014: FC (0036) FCB %11111100 20: CODEWORD FOR -L(0) = 0
0015: E2 (0037) FCB %11100010 21: CODEWORD FOR -L(1) = 11
0016: 94 (0038) FCB %10010100 22: CODEWORD FOR -L(2) = 111110
0017: 94 (0039) FCB %10010100 23: CODEWORD FOR -L(3) = 111110
(0040) *
(0041) * 9.6 KBPS CODE 2 (PREVIOUS LEVEL WAS -L(1))
(0042) *
0018: FC (0043) FCB %11111100 24: CODEWORD FOR L(0) = 0
0019: E4 (0044) FCB %11100100 25: CODEWORD FOR L(1) = 10
001A: 94 (0045) FCB %10100100 26: CODEWORD FOR L(2) = 111110
001B: A4 (0046) FCB %10100100 27: CODEWORD FOR L(3) = 111110
001C: FC (0047) FCB %11111100 28: CODEWORD FOR -L(0) = 0
001D: 04 (0048) FCB %11010100 29: CODEWORD FOR -L(1) = 110
001E: 94 (0049) FCB %10010100 30: CODEWORD FOR -L(2) = 111110
001F: 94 (0050) FCB %10010100 31: CODEWORD FOR -L(3) = 111110
0020 (0051) END

```

3.5.2 B REGISTER AND TRANSMIT FIFO

The input to the B register (74LS164,C15) is COMP during the transmit cycle and RFIFOUT during the receive cycle (see Figures 3-14 and 3-16, i.e.

$$\text{BIN} = \text{COMP} \cdot \overline{\text{W6}} + \text{RFIFOUT} \cdot \text{W6}$$

The B register may be serially loaded during the WRITE or CONVERT instructions (P1=1) if P4=1 at the last bit time of that word. When the speech coder is in the CVSD or ADM mode B is serially loaded once during the transmit cycle and once during the receive cycle. Thus, the previous transmit and receive bits are stored in B in interleaved form and we can get

$$\text{BK1} = b_k \oplus b_{k-1}$$

and
$$\text{BK2} = b_k \oplus b_{k-2}$$

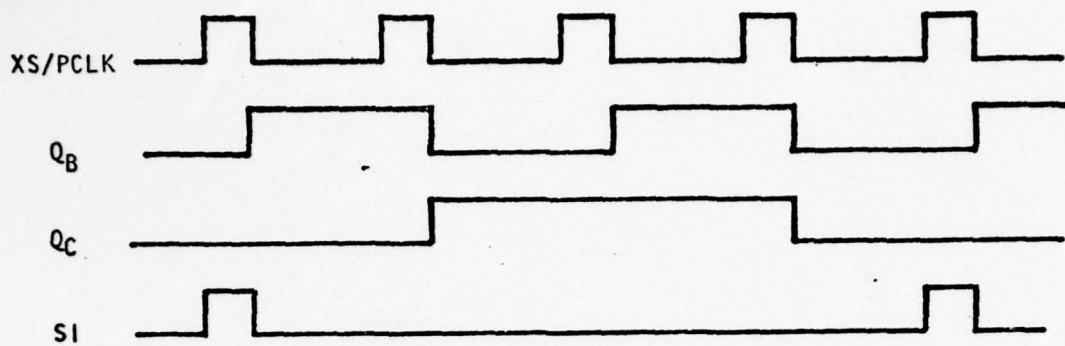
with b_k, b_{k-1}, b_{k-2} being all either transmit or receive bits at the proper time. BK1 and BK2 are used to address the parameter ROM in the CVSD/ADM mode.

The transmit FIFO consists of two 64x4 bit FIFO buffers (2XAM2841, locations C24 and C23) and 4-bit serial-to-parallel (74LS95,C19) and parallel-to-serial (74LS195,C18) converters. The serial-to-parallel clock (XS/PCLK) is ENCODECLK in the ARC mode (M1=0) and FIRSTBIT in the CVSD/ADM mode (M1=1). Similarly, the data is

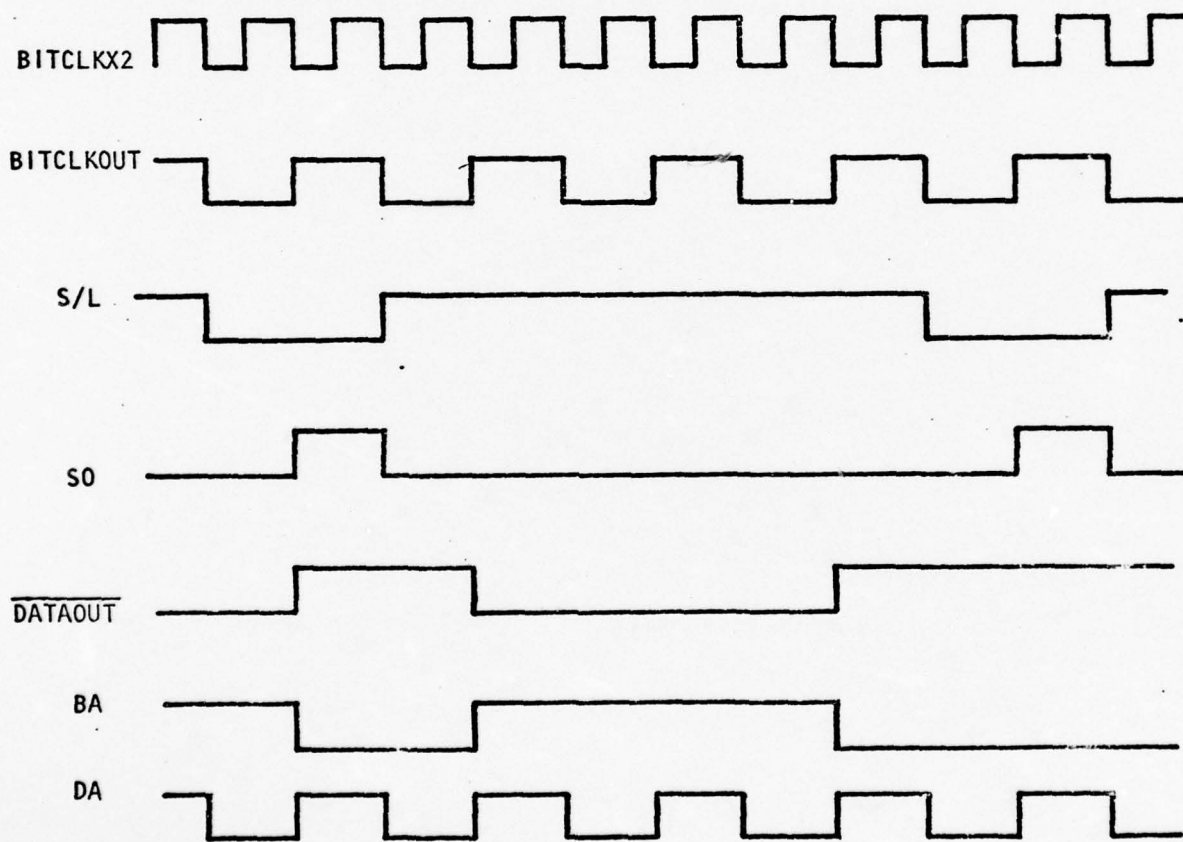
$$\text{XS/PDATA} = \text{ENCODEDATA} \cdot \overline{\text{M1}} \\ + \text{BS} \cdot \text{M1}$$

where BS is the most recent bit in the B register.

This data is serially loaded into a 4-bit register (74LS95,C19) at the negative transitions of XS/PCLK. Figure 3-4 (a) shows the XS/PCLK and the FIFO input clock SI. The FIFO accepts the 4 data bits at the positive transitions of SI. When the FIFO is full the input ready signal does not go high after the SI clock goes low. Thus, XFLAG=1 when the FIFO is full.



(A) INPUT (SERIAL/PARALLEL) TIMING



(B) OUTPUT (PARALLEL/SERIAL) TIMING

Figure 3=4. Transmit FIFO timing

The FIFO output is unloaded by the SO signal shown in Figure 3-4 (b). When the FIFO is empty the output ready signal OR does not go high after the SO signal goes low. This indication is sampled by BITCLKOUT and stored in a flipflop (1/2 74LS74,A19) shown in Figure 3-16.

The transmit FIFO has a maximum buffering capacity of 512 bits.

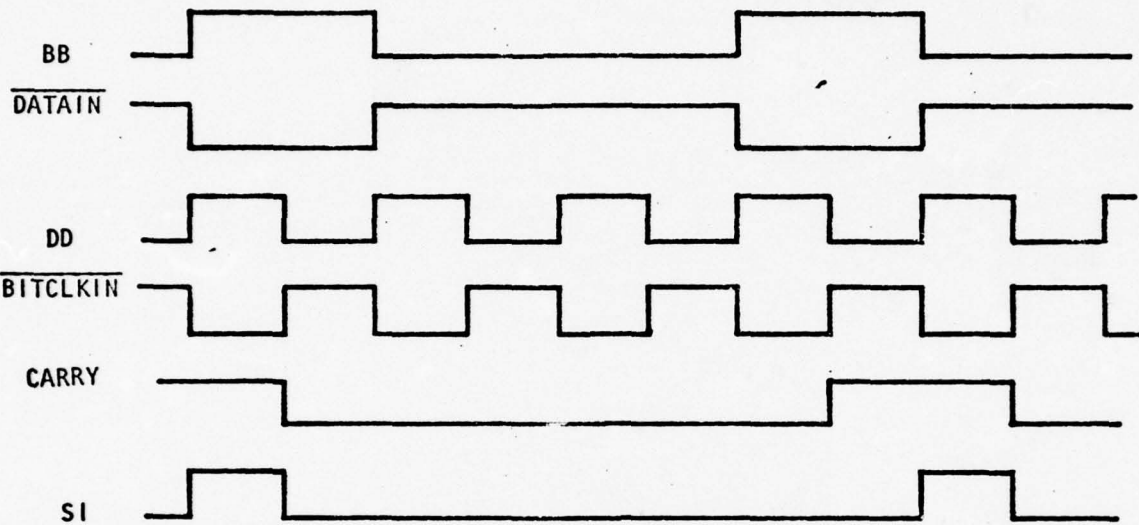
3.5.3 RECEIVE FIFO

The receive FIFO consists of three 64x4 bit FIFO's (3XAM2841, locations C17, C22 and C27) connected in tandem and 4-bit serial-to-parallel (74LS195,C13) and parallel-to-serial (74LS195,C26) converters (see Figure 3-17).

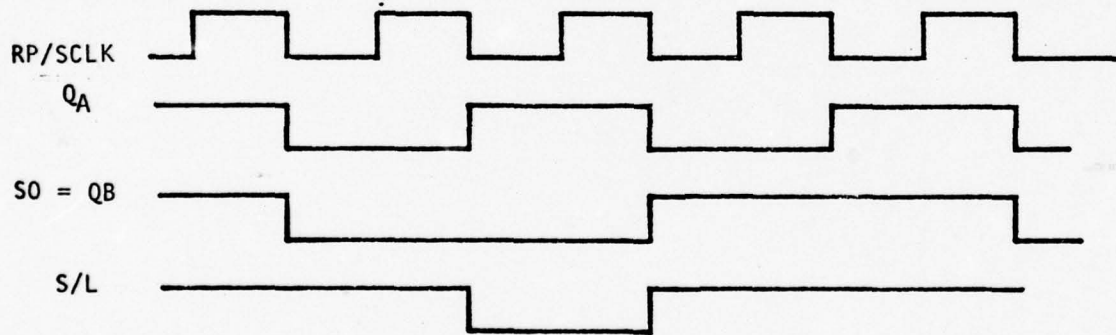
As explained in Section 2, initial synchronization of buffer depths is achieved by ensuring that the receive FIFO is 'full' when the filler word is received. If the buffer is not 'full' and a filler is received, the regular decoding cycle (buffer unloading) is suppressed and the zero quantizer level is substituted. In the above context 'full' implies buffering of as many bits as the maximum capacity of the transmit FIFO, i.e., 512 bits. Clearly, some additional buffering must be provided at the receiver to allow time to react to the receiver FIFO 'full' signal, RFLAG. The additional AM2841 serves this purpose.

The input data $\overline{\text{DATAIN}}$ is normally $\overline{\text{BBTTL}}$ except in the LOOPBACK configuration when it is $\overline{\text{DATAOUT}}$. The accompanying clock $\overline{\text{BITCLKIN}}$ is similarly either $\overline{\text{DDTTL}}$ or $\overline{\text{BITCLKOUT}}$. As shown in the timing diagram of Figure 3-5 (a) $\overline{\text{DATAIN}}$ is sampled and serially loaded into the 74LS195 when it is stable. When 4 bits have been collected they are parallel loaded into the AM2841 by SI. The data trickles down to the last empty slot in one of the remaining two FIFO's. When both the second and third FIFO's are full the input ready (IR) signal on the second FIFO does not go high after the SI clock goes low. Thus, RFLAG is simply $\overline{\text{IR}}$ (see Figure 3-17).

The timing diagram for the output (parallel-to-serial) side of the receive FIFO is given in Figure 3-5 (b). RP/SCLK is the DECODECLK in the ARC mode (M1=0) and FIRSTBIT in the CVSD or ADM modes (M1=1) as shown in Figure 3-16.



(A) INPUT (S/P) TIMING



(B) OUTPUT (P/S) TIMING

Figure 3-5. Receive FIFO Timing

3.5.4 DECODER

Two different decoding strategies are used for the 9.6 kb/s code and the 16 kb/s code.

At 9.6 kb/s, the bits are counted out until a zero is obtained; the number of ones counted plus 1 addresses a 32x8 bit decoder ROM (82S123,B21) whose output gives the quantizer level (see Figure 3-18).

The decoding process is normally initiated during a receive cycle when P9=0 and P10=1 in an ADD instruction. However, if the previous code word was a 'two-level' codeword, e.g., two '-1(1)' levels in a row, then the SECONDWD signal is high which inhibits the decoding cycle. Further, an extra decoding cycle is initiated when the code word just decoded is a filler (RFILLER=1) and the receive FIFO is full (RFLAG=1). Thus,

$$\text{INITDECODE} = \text{ENDECODE} \cdot \overline{\text{SECONDWD}} + \text{RFILLER} \cdot \text{RFLAG},$$

where $\text{ENDECODE} = \overline{W6} \cdot \overline{P9} \cdot P10 \cdot \overline{P1}$. The INITDECODE pulse clears the decode counter (74LS163,B22) and sets the ZEROBIT flipflop (ZEROBIT=0) (1/2 74LS107,B26). The decode ROM is addressed by M0 and the four counter outputs Q_D through Q_A. When the counter is cleared the eighth ROM output, which is RFILLER, is zero. As shown in the decoder timing diagram (Figure 3-6) CONTINUE goes up enabling the DECODECLK, and permitting the counter to count up since at 9.6 kb/s (M0=1) COUNT=CONTINUE. As soon as a zero bit is unloaded (RFIFOUT=1) the ZEROBIT flipflop is reset (ZEROBIT=1) stopping the DECODECLK and inhibiting the counter. The counter output is then one greater than the length of the code word and the decoder ROM output D1-D3 corresponds to the quantizer level. If the 'two-level' code word has been received ROM outputs D4-D6 correspond to the second quantizer level and D7=1. The latter signal on the negative going edge of $\overline{\text{STROBE}}$ and stored in the SECONDWD flipflop (1/2 74LS 107,B26) for exactly one sample interval.

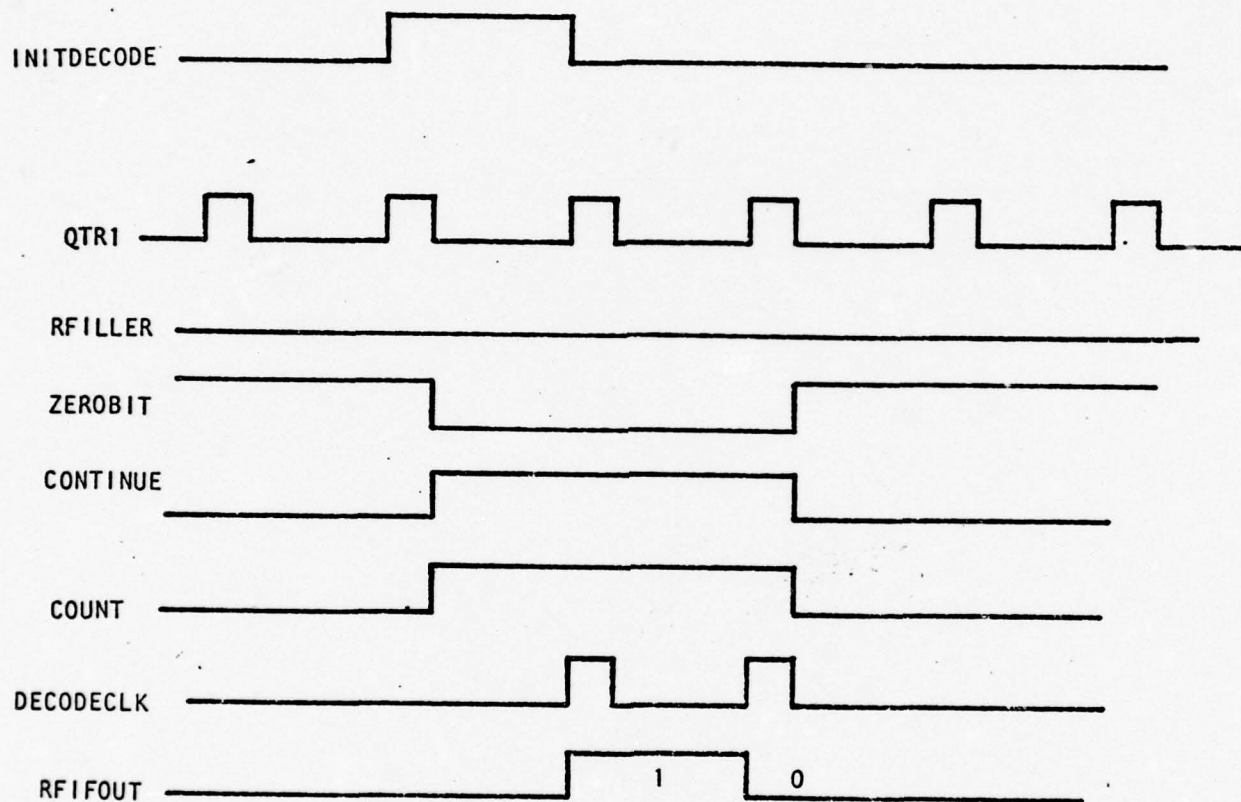


Figure 3-6. Decoder Timing for 9.6 Kb/s Code

At 16 kb/s, we use tree decoding (see Figure 3-7 for the code tree). ROM output bits D1-D3 and RFIFOUT are loaded into the counter which is used as a register (COUNT=0). The counter output forms the next base address for interior nodes (indicated at the ROM output by D7=1). Further, CONTINUE=D7. Thus, the decoder progresses through the tree as each bit is unloaded from the FIFO. The decode cycle stops at a terminal node (D7=0) at which time the ROM output bits D1-D3 correspond to the quantizer level, which can then be loaded into the Q register. SECONDWD is always zero for 16 kb/s operation (M0=0).

The end of a filler word is considered to be an interior node (D7=0, RFILLER=1) and the decoding process is reinitiated, provided that the receive FIFO 'full' signal RFLAG is high.

The contents of the decoder ROM are given in Table 3-14. The first column specifies the contents in hexadecimal notation.

3.6 ANALOG CIRCUITS

The analog circuitry consists of a signal limiter, transmit and receive low-pass filters, sample-and-hold circuit, D/A converter and a comparator.

3.6.1 Low-Pass Filters

Each low-pass filter is a 7th-order elliptic filter consisting of three biquad sections and one single-pole section. The transfer function is of the form

$$\frac{k}{s + \sigma_0} \prod_{i=1}^3 \frac{s^2 + \Omega_{zi}^2}{s^2 + 2\sigma_i s + (\sigma_i^2 + \Omega_{pi}^2)}$$

with the following normalized values of the poles and zeros for a filter with a 0 dB bandwidth of unity:

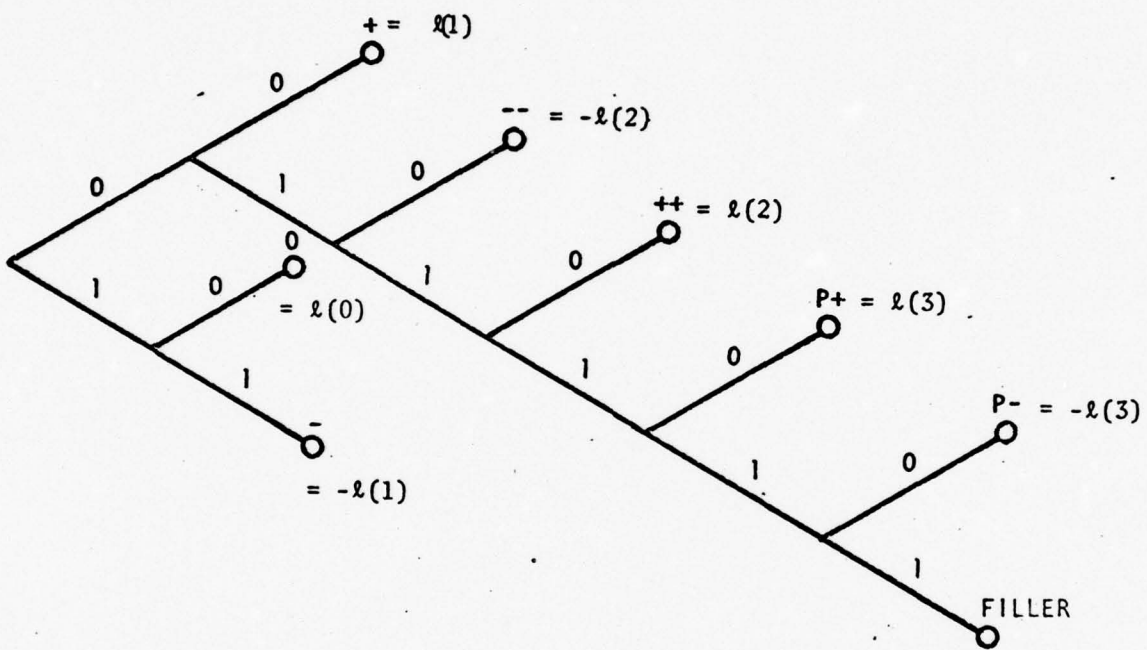


Figure 3-7. 16 Kb/s Code Tree

TABLE 3-14. Decoder

(0001) * DECODER			
(0002) * THIS LISTING SPECIFIES THE CONTENTS OF THE DECODER			
(0003) * ROM			
(0004) *			
(0005) * 16 KBPS DECODER			
(0006) *			
(0007)	FCB	%00100010	00: STARTING NODE
(0008)	FCB	\$00	01: UNUSED
(0009)	FCB	%01000010	02: BIT1 = 0
(0010)	FCB	%01100010	03: BIT1 = 1
(0011)	FCB	%00100000	04: CODE WORD 00, LEVEL = L(1)
(0012)	FCB	%10000010	05: BIT1 = 0, BIT2 = 1
(0013)	FCB	%00000000	06: CODE WORD 10, LEVEL = L(0)
(0014)	FCB	%10100000	07: CODE WORD 11, LEVEL = -L(1)
(0015)	FCB	%11000000	08: CODE WORD 010, LEVEL = -L(2)
(0016)	FCB	%10100010	09: BIT1 = 0, BIT2 = 1, BIT3 = 1
(0017)	FCB	%01000000	10: CODE WORD 0110, LEVEL = L(2)
(0018)	FCB	%11000010	11: BIT1 = 0, BIT2 = 1, BIT3 = 1, BIT4 = 1
(0019)	FCB	%01100000	12: CODE WORD 01110, LEVEL = L(3)
(0020)	FCB	%11100010	13: BIT1 = 0, BIT2 = 1, BIT3 = 1, BIT4 = 1, BITS = 1
(0021)	FCB	%11100000	14: CODE WORD 01110, LEVEL = -L(3)
(0022)	FCB	%00000001	15: CODE WORD 011111, FILLER
(0023) *			
(0024) * 9.6 KBPS DECODER			
(0025) *			
(0026)	FCB	%00000000	16: NO. OF ONES = 0, LEVEL = L(0)
(0027)	FCB	%00100000	17: NO. OF ONES = 1, LEVEL = L(1)
(0028)	FCB	%10100010	18: NO. OF ONES = 2, LEVELS = -L(1), L(0)
(0029)	FCB	%10100110	19: NO. OF ONES = 3, LEVELS = -L(1), L(1)
(0030)	FCB	%10110110	20: NO. OF ONES = 4, LEVELS = -L(1), -L(1)
(0031)	FCB	%01000000	21: NO. OF ONES = 5, LEVEL = L(2)
(0032)	FCB	%11000000	22: NO. OF ONES = 6, LEVEL = -L(2)
(0033)	FCB	%10101010	23: NO. OF ONES = 7, LEVELS = -L(1), L(2)
(0034)	FCB	%10110110	24: NO. OF ONES = 8, LEVELS = -L(1), -L(2)
(0035)	FCB	%00000001	25: NO. OF ONES = 9, FILLER
(0036)	FCB	\$00	26: UNUSED
(0037)	FCB	\$00	27: UNUSED
(0038)	FCB	\$00	28: UNUSED
(0039)	FCB	\$00	29: UNUSED
(0040)	FCB	\$00	30: UNUSED
(0041)	FCB	\$00	31: UNUSED
(0042)			END
0000:	22		
0001:	00		
0002:	42		
0003:	62		
0004:	20		
0005:	82		
0006:	00		
0007:	A0		
0008:	C0		
0009:	A2		
000A:	40		
000B:	C2		
000C:	60		
000D:	E0		
000E:	E2		
000F:	01		
0010:	00		
0011:	20		
0012:	A2		
0013:	A6		
0014:	B6		
0015:	40		
0016:	C0		
0017:	AA		
0018:	BA		
0019:	01		
001A:	00		
001B:	00		
001C:	00		
001D:	00		
001E:	00		
001F:	00		
	0020		

$$\begin{aligned}
\sigma_0 &= 0.52508, \\
\Omega_{z1} &= 2.2039, \\
\sigma_1 &= 0.04038, \quad \Omega_{p1} = 1.0243, \\
\Omega_{z2} &= 1.3817, \\
\sigma_2 &= 0.15999, \quad \Omega_{p2} = 0.93061, \\
\Omega_{z3} &= 1.2085, \\
\sigma_3 &= 0.37238, \quad \Omega_{p3} = 0.63184.
\end{aligned}$$

The minimum out-of-band attenuation of this filter is about 50 dB and it has a transition width of about 20%. The active-filter circuit configuration used for each biquad section is apparent from the schematics of the transmit and receive low-pass filters shown in Figures 3-19 and 3-20, respectively. Note that for dynamic range considerations the single pole section is the first filter section and the biquad sections are connected in increasing order of Q.

Figure 3-8 shows measured amplitude and delay characteristics of the 7th-order elliptic filter.

The transmit elliptic filter is preceded by a signal limiter (see Figure 3-19) which blocks out dc and symmetrically clips large level signals which would cause saturation in the analog circuitry or overflow in the digital arithmetic circuits.

A switch is provided on the panel which permits selection of the voice input from the attached handset or an external source through an RCA type phono jack. The input impedance at the phono jack is approximately 600 Ω .

The receive elliptic filter is followed by a two-pole Chebyshev filter (Figure 3-20) with 0.5 dB passband ripple and a 3 dB cutoff frequency of 3700 Hz. This filter section which has an output impedance of 600 Ω , not only serves as a buffer but also provides additional rejection of out-of-band signal components, e.g., the 6000 Hz sampling frequency.

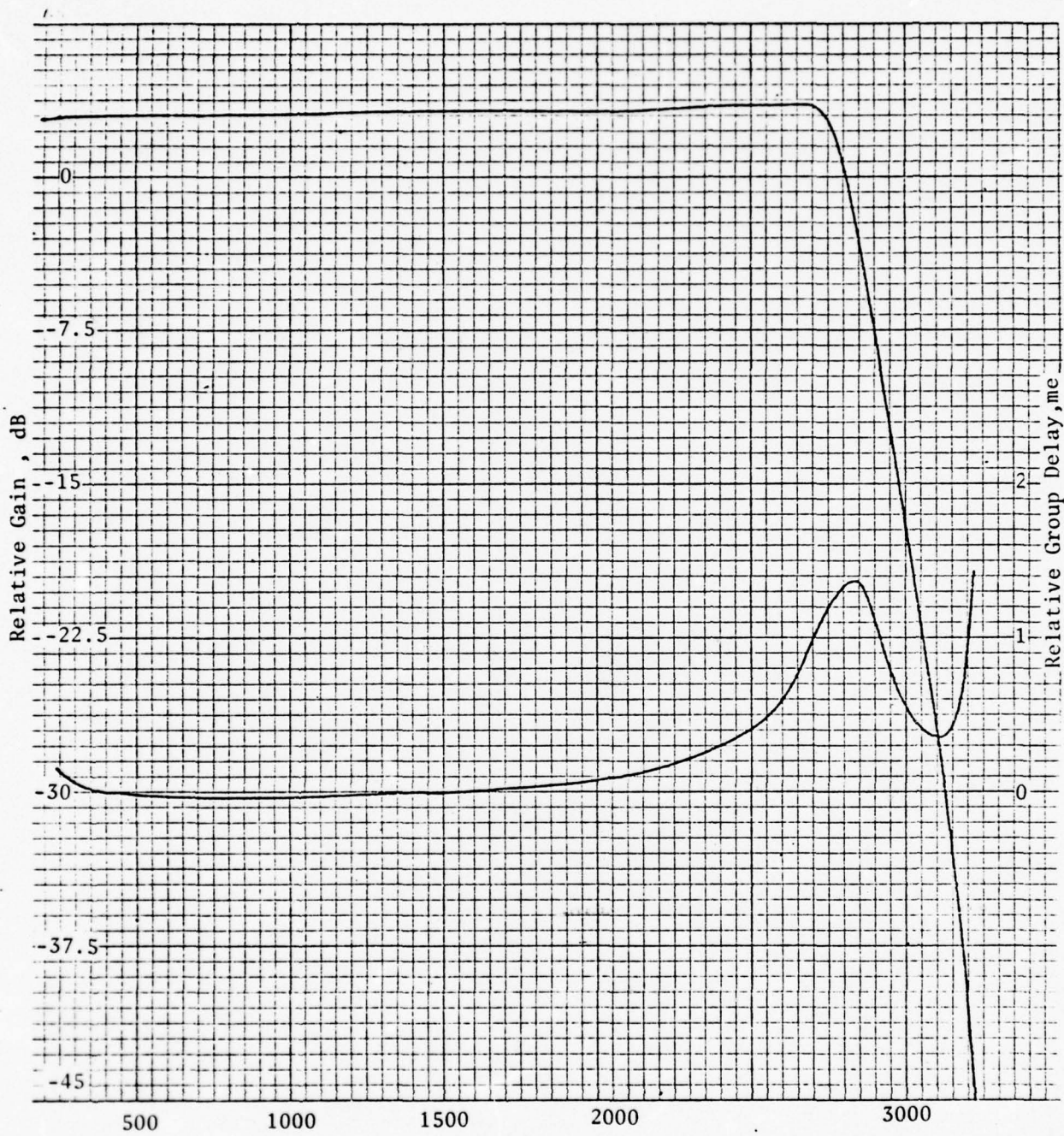


Figure 3-8 Frequency , Hz

3.6.2 Sample-and-Hold, D/A and Comparator

The output of the transmit low-pass filter is connected to the sample-and-hold circuit as shown in Figure 3-21. The low on-resistance analog switch (DG200, U10) is turned on by the $\overline{\text{STROBE}}$ signal from the digital card during the last half of each receive cycle, causing capacitor C21 to charge up to the new signal voltage. The analog switch is turned off at the start of the transmit cycle and the signal sample is held on the capacitor by buffering it from the comparator through a high input-impedance voltage follower (LM310, U9).

The output of the adder and eleven T register outputs (Figure 3-7) are fed to the analog card through connector J1 where they are clocked into the D/A buffer register (2X74LS174, U6 and U5) at the positive transitions of D/ABFRCLK. During the first half of the transmit cycle the D/A buffer register is loaded with successive test values (predictions $p_k \pm$ quantizer thresholds). These 12-bit numbers are converted to analog voltages through a 12-bit D/A converter (DACHY12BC, U1) and compared to the input analog sample. The comparator (LM311, U3) output COMP which is a TTL compatible signal is fed back to the digital card to be stored in the Q or B register.

During the last half of the receive cycle the reconstructed speech sample is loaded into the D/A buffer register and converted to an analog voltage. After the D/A output has settled (about 5 μ s) this voltage is pulsed into the receive low-pass filter through an analog gate (DG200, U4) which is turned on by $\overline{\text{STROBE}}$. When $\overline{\text{STROBE}}$ is high the complementary signal STROBE grounds RLPPIN1 through a second analog switch.

When the speech digitizer is in the normal configuration a controlled amount of coupling is provided between the transmitter and the receiver through an analog switch (U10) controlled by the signal LOOPBACK from the digital card. This coupling, commonly called 'sidetone', is deemed essential by the telephone company. Too much sidetone causes the talker to lower his voice, thereby reducing the volume which the

listener receives; too little sidetone makes telephone conversation seem unnatural and tends to cause people to talk too loudly. The sidetone is totally suppressed in the loopback configuration to permit the speech digitizer to be tested and used for processing recorded speech.

3.7 INTERFACE CIRCUITS

The TTL output signals for data and clock are passed to the analog board where they are converted to nominal MIL-188 levels by high speed operational amplifier drivers. Rise and fall times are controlled by integrating capacitors which have been chosen to be a compromise for the data rates anticipated. Continuous signals CA and CD are provided by integrated MIL-188 drivers.

Receivers provided for clock and data are integrated MIL-188 interfaces which convert to TTL signal levels which are then fed back to the digital card. Proper operation of all these devices is guaranteed over a range of speeds and cable lengths. See Figure 3-22.

SECTION 4

OPERATION

4.1 GENERAL

This section contains a functional description of the input/output signals, and controls instructions for operating the Codex speech digitizer.

4.2 I/O SIGNALS, CONTROLS AND INDICATORS

Table 4-1 lists the input-output signals carried via connectors located on the panel. The relevant pin numbers and a brief description of the signals is also given in Table 4-1. All signals on the 25-pin connector are of the EIA type for direct connection to the proper modem EIA connector. The EIA outputs are also suitable for directly driving the corresponding inputs on another ARC via a crossover cable as shown in Figure 4-1.

The input and output impedances of the audio circuits are approximately 600Ω . When using an audio source other than the supplied telephone handset, tape recorder IN switch should be turned ON and the input signal level should be adjusted to be less than 2.2 volts peak-to-peak. The audio output may be switched independently between the handset and a tape recorder.

Table 4-2 lists all the control switches available on the Codex speech digitizer with a brief description of the function of each switch. As shown in Figure 4-2 the LOOPBACK switch connects the local ARC transmitter and receiver back-to-back thus permitting the unit to be functionally tested in an isolated environment. If both tape recorder switches are OFF (handset connected) then in the LOOPBACK configuration one can listen to one's own voice after it has been digitized and reconstructed.

TABLE 4-1 INPUT-OUTPUT SIGNALS

SIGNAL NAME	TYPE	LOCATION	DESCRIPTION
Protective Ground (AA)	MIL 188	J7-1	Chassis ground. Isolated from Signal Ground (AB) in the ARC.
Signal Ground (AB)	MIL 188	J7-7	Common signal and dc power supply ground. Isolated from protective ground (AA) in the ARC.
Transmit Output Data (BA)	MIL 188	J7-2	Serial binary data with transitions on the positive-going transitions of the internal transmit clock (DA) or a modem supplied clock (DB) (selectable by toggle switch on A4 card).
Receive Input Data (BB)	MIL 188	J7-3	Serial binary data from a modem or directly from an ARC transmitter. Data transitions must occur on the positive-going transitions of the accompanying clock (DD).
Request to Send (CA) (output)	MIL 188	J7-4	Constantly held at a positive level to indicate that data transmission is desired.
Data Terminal Ready (CD) (output)	MIL 188	J7-20	Constantly held at a positive level.
Transmit Signal Element Timing (DA) (output)	MIL 188	J7-24	A serial data rate clock with positive transitions corresponding to data (BA) transitions.
External Transmit Serial Clock (DB) (input)	MIL 188	J7-15	A data rate clock from a modem. When selected by the toggle switch on A4 card the data (BA) transitions will occur on positive-going transitions of this clock (DB).
Receive Signal Element Timing (DD) (input)	MIL 188	J7-17	Data rate clock accompanying the receive data (BB).

TABLE 4-1 (continued)

Voice in	Audio	RCA phono Jack (IN)	Audio input (to be digitized) from tape recorder
Voice Out	Audio	RCA phono Jack (OUT)	Audio output reconstructed by the ARC

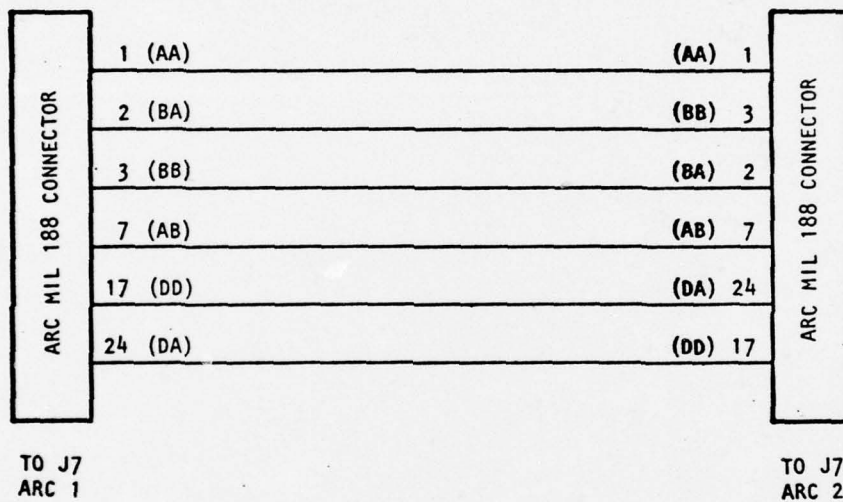


FIGURE 4-1

CROSSOVER CABLE FOR DIRECTLY CONNECTING TWO ARC'S.

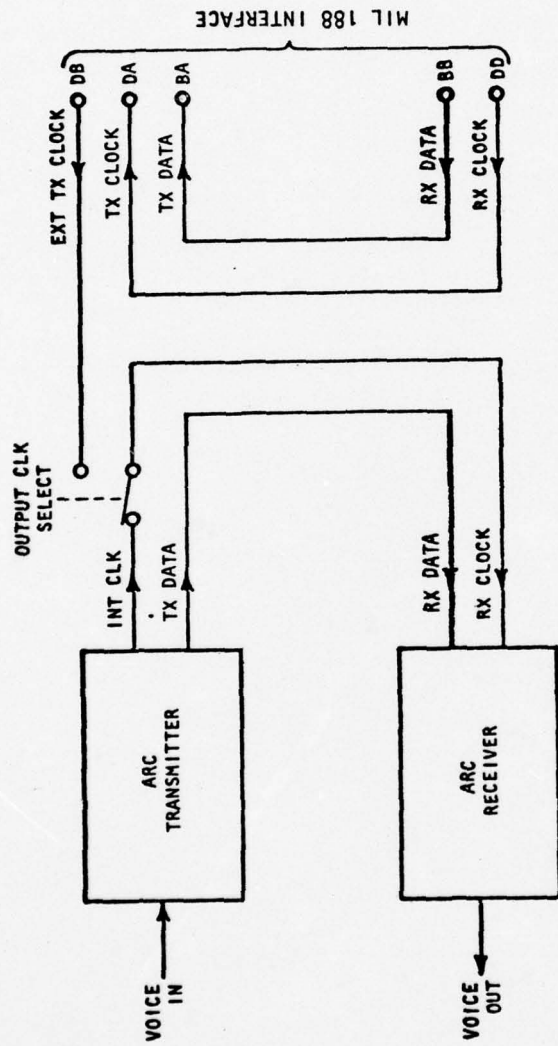


FIGURE 4-2
LOOPBACK CONFIGURATION

TABLE 4-2 CONTROL SWITCHES

CONTROL	LOCATION	FUNCTION																				
POWER ON/OFF switch	Panel	Controls the application of 115/230 volts ac power to the dc power supply and cooling fan motors.																				
MODE Control switches	Location A6 switches 1-3 on digital card	<p>Switch 1. Controls bit rate: ON - 16K bps, OFF - 9.6K bps.</p> <table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="2">Switches</th> </tr> <tr> <th colspan="2"></th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr> <td></td> <td>ARC</td> <td>ON</td> <td>ON</td> </tr> <tr> <td>MODE</td> <td>ADM</td> <td>OFF</td> <td>ON</td> </tr> <tr> <td></td> <td>CVSD</td> <td>OFF</td> <td>OFF</td> </tr> </tbody> </table>			Switches				2	3		ARC	ON	ON	MODE	ADM	OFF	ON		CVSD	OFF	OFF
		Switches																				
		2	3																			
	ARC	ON	ON																			
MODE	ADM	OFF	ON																			
	CVSD	OFF	OFF																			
LOOPBACK switch	Location A6 switch 4 on digital card.	When ON it internally connects the transmitter output to the local receiver input. If the external clock (DB) is available on the MIL 188 connector (J7) it may be used for transmit and receive timing via the slide switch on the digital card. The effect of the loopback switch at the EIA interface is to loop BB to BA and DD to DA (see Figure 4-2).																				
Output CLOCK	Location C21 on digital card	When this slide switch is in the FORWARD position the internal clock (DA) is used for transmit output timing. The externally supplied clock (DB) is used when the switch is in the AFT position.																				
Tape Recorder IN and OUT switches	Panel	These switches connect either the tape recorder jacks (ON) or the handset (OFF) to the digitizer.																				
FIFO Reset switch	Location A6 Switch 2 of smaller switch dip	Used to reset FIFO buffers after power is applied.																				

The dip switches mounted on the digital card allow selection between different modes of operation, i.e., ARC, CVSD, ADM and internal/external transmit clock. A FIFO reset switch is also provided.

4.3 OPERATING PROCEDURE

Operating the experimental models of the speech digitizer is very simple since an initial handshaking or synchronization procedure is not required. The following steps may be followed:

1. Connect the Speech Digitizer to a modem via a straight through MIL 188 cable to J7 or another ARC through the crossover cable shown in Figure 4-1..
2. Switch the telephone handset on (Tape Recorder off) or connect a tape-recorder output to the RCA phono jack marked IN (Tape Recorder ON).
3. Select the proper transmission rate (9.6/16 kb/s) and mode by pressing the RATE rocker dip switches on the digital card to the appropriate position (see Table 4-2).
4. Depress the FIFO reset switches momentarily.
5. Talk normally into handset.

SECTION 5
ACCEPTANCE TEST PROCEDURE

5.1 GENERAL

All 12 sets of equipment as described in section 2.1.1 of Statement of Work Contract DCA 100-76-c-0026 shall meet the criteria of section 3.1 of the Statement of Work in that all units shall be superior or equal in performance to the "Exploratory Development" modules developed under DCA Contract DCA 100-74-C-0053.

Prior to shipment by the contractor to DCA all units will be evaluated utilizing a tape-recorded segment of spoken English by each of a number of different speakers, male and female. Additionally full electrical and mechanical inspection shall be performed to ensure correct functioning of all modes in all units.

At DCA this tape will be used to verify that all units are superior or equal to the exploratory development module. This test will be witnessed by DCA for the purpose of determining the acceptability of these units per the Statement of Work mentioned above. In addition, the correct functioning of the additional operating modes will be demonstrated.

5.2 PROCEDURE

1. Power on both exploratory development module and unit to be evaluated:
2. Set both units to ARC Mode at 16 KB and place units in loopback (see operation section)
3. Run tape segment through exploratory development module and monitor audio signal quality.
4. Run same tape segment through unit to be evaluated and compare quality.
5. Set both units to ARC mode at 9.6 KB.
6. Run tape segments again and compare quality.

7. In addition, run unit to be evaluated in CVSD and ADM modes at 9.6 KB and 16 KB and monitor quality.

8. Test handset on unit to be evaluated by turning off tape recorder input and output switches and speaking into handset normally.

9. To verify MIL 188 interface function on units to be evaluated, use crossover cable as described in operation section. With units out of loopback pass data from one unit to the other and monitor quality.

SECTION 6

PARTS LIST

6.1 GENERAL

This section provides a list of replaceable parts for the Codex speech digitizer.

ARC CARRYING CASE

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
SK4066	Carrying Case	1
65000C2	Handset, Black	1
TR202-1	Power Supply	1
SK4067	Chassis	1
SR-11-2	Strain, Relief	1
SR-4N-4	Strain, Relief	2
SK4068	Plate, Component	1
SK4069	Mtg. Strap	2
45-000-S-H	Circuit, Breaker	1
9223-SS-140-0	Spacer (.19 Lg.)	2
6 x 3/4 Type B	Pan Hd. Screw	2
8 x 3/4 Type B	Flat Hd. Screw	4
SK4070	Barrier, Plate	1
22-01-2251	P.C. Connector	2
08-56-0110	Contact, Crimp	2
1-87175-7	Housing	2
1-87175-9	Housing	1
1-87175-5	Housing	1
7201-P3-P-D-2-0	Switch (DPDT)	2
33-804	Phone Jack	2
17-304-01	Connector, 25 Pin	1
17-763-02	Contact	8
2252	Grommet	1
8094-N-0440	Standoff	4
106821-603	Microphone	1
87165-1	Contact, Crimp	22
324597	Term. Slotted	4

ARC ANALOG

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
35867-01	Circuit Card	1
NA55D3830F	383-Ohm-1/8W 1%	2
NA55D3241F	3.24K 1/8W 1%	2
NA55D1302F	13.0K 1/8W 1%	2
NA55D9531F	9.53K 1/8W 1%	2
NA55D49R9F	49.9-Ohm 1/8W 1%	2
RC07GF102J	1K 1/4W 5%	3
RC07GF301J	300-Ohm 1/4W 5%	1
NA55D1002F	10K 1/8W 1%	5
NA55D2372F	23.7K 1/8W 1%	2
NA55D1692F	16.9K 1/8W 1%	2
NA55D1332F	13.3K 1/8W 1%	4
NA55D2052F	20.5K 1/8W 1%	2
NA55D3322F	33.2K 1/8W 1%	2
NA55D2672F	26.7K 1/8W 1%	2
NA55D1242F	12.4K 1/8W 1%	2
NA55D2742F	27.4K 1/8W 1%	2
NA55D3922F	39.2K 1/8W 1%	2
NA55D2002F	20K 1/8W 1%	4
NA55D8661F	8.66K 1/8W 1%	2
NA55D1872F	18.7K 1/8W 1%	2
NA55D2612F	26.1K 1/8W 1%	2
NA55D1303F	130K 1/8W 1%	2
NA55D6491F	6.49K 1/8W 1%	2
NA55D6342F	63.4K 1/8W 1%	2
NA55D4532F	45.3K 1/8W 1%	1

ARC ANALOG (Cont)

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
NA55D6492F	64.9K 1/8W 1%	2
NA55D4122F	41.2K 1/8W 1%	2
NA55DD2552F	25.5K 1/8W 1%	2
NA55DD8252F	82.5K 1/8W 1%	2
NA55DD1132F	11.3K 1/8W 1%	1
NA65D	Select Valve at Test	1
NA65D	Select Valve at Test	1
NA65D	Select Valve at Test	1
NA65D	Select Valve at Test	1
NA65D	Select Valve at Test	1
NA65D	Select Valve at Test	1
RC07GF166J	16M 1/4W 5%	1
RC07GF395J	3.9M 1/4W 5%	1
RC07GF681J	680-Ohm 1/4W 5%	2
NA55D6341F	6.34K 1/8W 1%	1
NA55D9310F	931-Ohm 1/8W 1%	1
NA55D1821F	1.82K 1/8W 1%	1
RC07GF821J	820-Ohm 1/4W 5%	4
2DDU60D103M	.01 MF, 20V	1
T362B106M025AS	10 MF, 25V	18
2DDU60M503M	.05 MF, 25V	38
PP-11-.01-100-1	.01 MF, 100V	4
2DDU60D104M	.1 MF, 20V	2
PP-11-.0047-100-1	.0047 MF, 100V	9

ARC ANALOG (Cont)

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
PP-11-.001-100-1	.001 MF, 100V	3
2SA-.00047-100-1	.00047 MF, 100V	2
PP-11-0027-100-1	.0027 MF, 100V	1
8131-050-651-474M	.47 MF, 50V	2
IN5233B	Diode, Zener	1
IN914	Diode, Zener	4
DAC-HY12DC	Integrated, Circuit	1
IF356H	Integrated, Circuit	2
LM311D	Integrated, Circuit	1
DG200BA	Integrated, Circuit	2
SN74LS174	Integrated, Circuit	2
LM1160	Integrated, Circuit	2
LM310D	Integrated, Circuit	1
MC1458CP1	Integrated, Circuit	12
LM1150	Integrated, Circuit	1
IM320H12	Voltage Regulator 12V	1
324-AG2D	Socket, 24 Contact	1
22-01-2231	Conn. Wafer 25 Cir.	1
87232-8	Header Assy, 8 Pos.	2
87232-2	Header Assy, 2 Pos.	1
87232-4	Header Assy, 4 Pos.	1
3643-2-05	Terminal, Turret	21
JO.250 x 0.125T22	Wire, Jumper (Fancourt)	8

ARC DIGITAL BOARD

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
8136-PG13-90	Augat, Panel	1
SN74LS00	Integrated, Circuit	4
SN74LS02	Integrated, Circuit	2
SN74LS04	Integrated, Circuit	5
SN74LS08	Integrated, Circuit	2
SN74LS10	Integrated, Circuit	2
SN74LS51	Integrated, Circuit	2
SN74LS74	Integrated, Circuit	1
SN74LS83	Integrated, Circuit	1
SN74LS86	Integrated, Circuit	2
SN74LS93	Integrated, Circuit	3
SN74LS95A	Integrated, Circuit	1
SN74LS107	Integrated, Circuit	3
SN74LS151	Integrated, Circuit	2
SN74LS153	Integrated, Circuit	3
SN74LS157	Integrated, Circuit	7
SN74LS163	Integrated, Circuit	10
SN74LS164	Integrated, Circuit	2
SN74LS194	Integrated, Circuit	7
SN74LS195	Integrated, Circuit	4
SN74S287	Integrated, Circuit	2
SN74C89	Integrated, Circuit	3
SN74H183	Integrated, Circuit	1
SN74S471	Integrated, Circuit	2
AM2841	Integrated, Circuit	5
82S123	Integrated, Circuit	2

ARC DIGITAL BOARD (Cont)

<u>Part No.</u>	<u>Description</u>	<u>Qty</u>
898-1-R3.3K	Resistor, Network	1
CO-238B	Oscillator	1
435166-1	Dip Switch, Toggle	1
8531-B	SPST Pushbutton	1
.01MF, 16V	Capacitor	23
10MF, 25V	Capacitor	6
SN7425	Integrated, Circuit	1
2345-3	Term. Solder	2
435626-8	Dip Switch	1
435673-2	Dip Switch	1

AD-A031 509

CODEX CORP NEWTON MASS

F/G 17/2

CODEX SPEECH DIGITIZER ADVANCED DEVELOPMENT MODEL.(U)

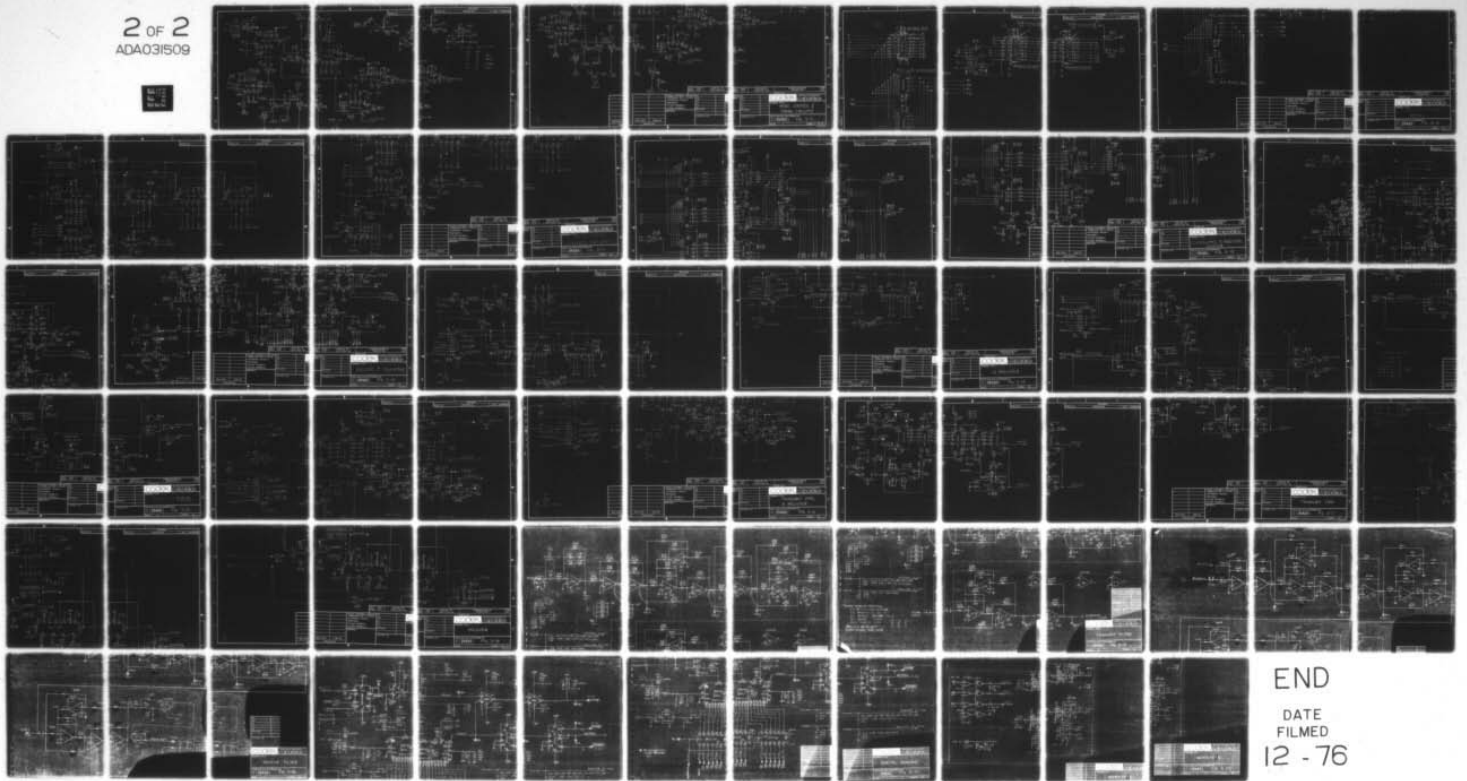
JUN 76 G D FORNEY, S QURESHI

DCA100-76-C-0026

UNCLASSIFIED

NL

2 OF 2
ADA031509

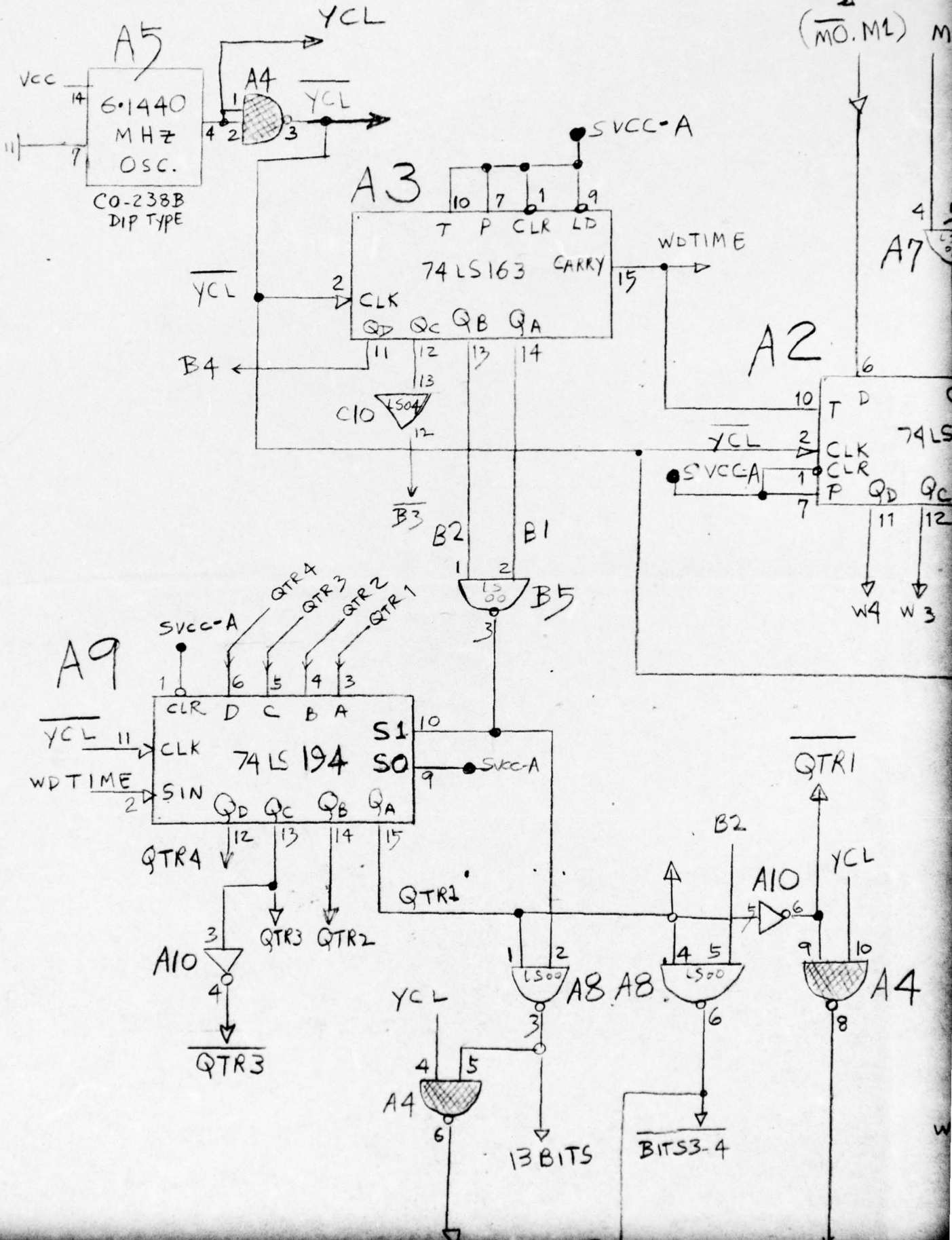


END

DATE
FILMED
12 - 76

From Fig. 3-15

(MO.ML) M

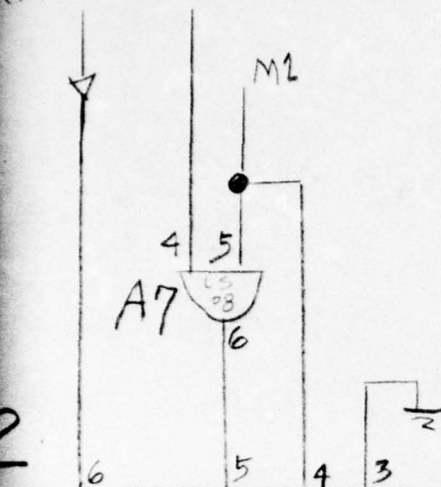


D

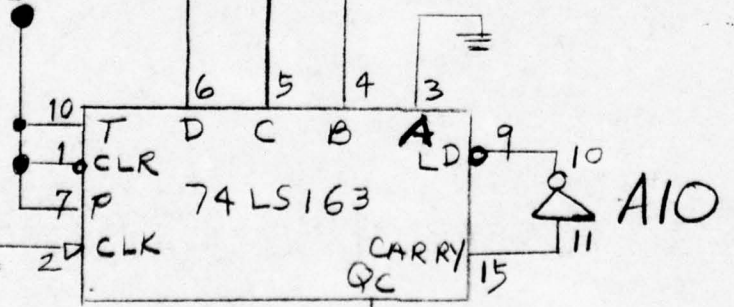
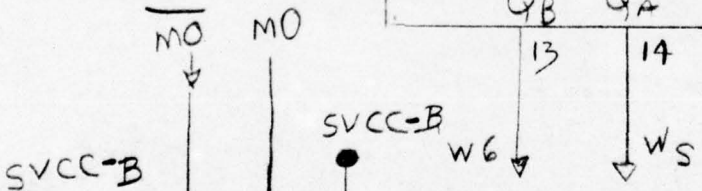
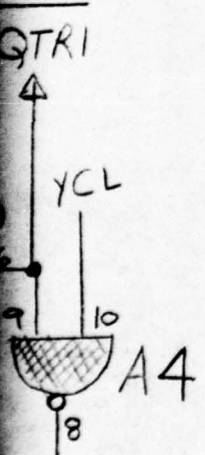
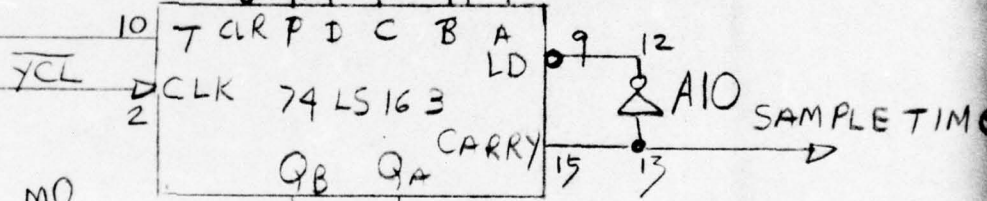
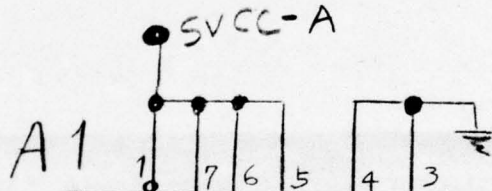
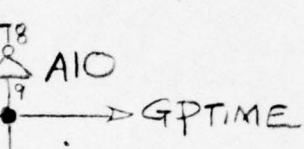
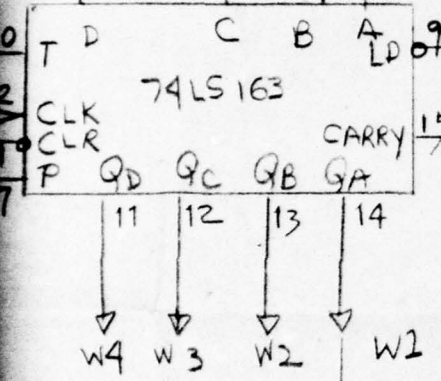
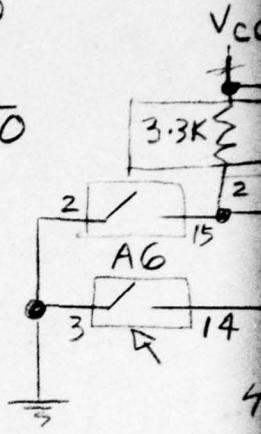
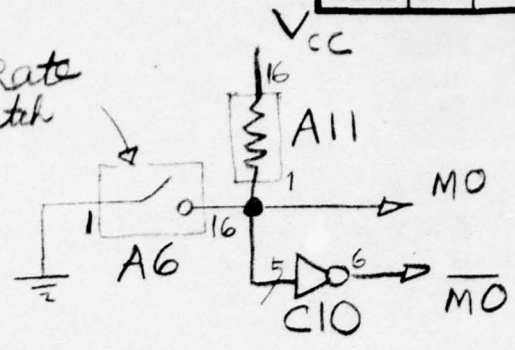
C

B

Fig 3-15
(MO.ML) MD



Rate switch
closed - 16K
open - 9.6K



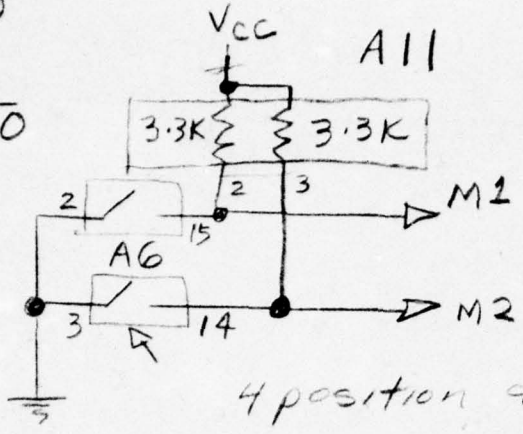
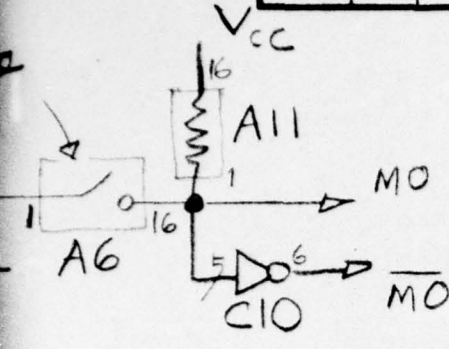
A15

BITCLK X2

WDTIME YCL

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

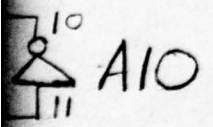
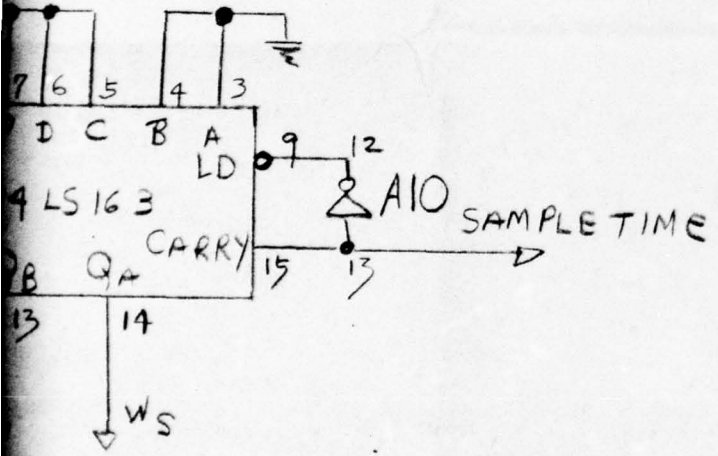
3



4 position dip switch

M 1	M 2	
0	0	ARC
0	1	ARC _b
1	0	CVSD _b
1	1	ADM

VCC-A

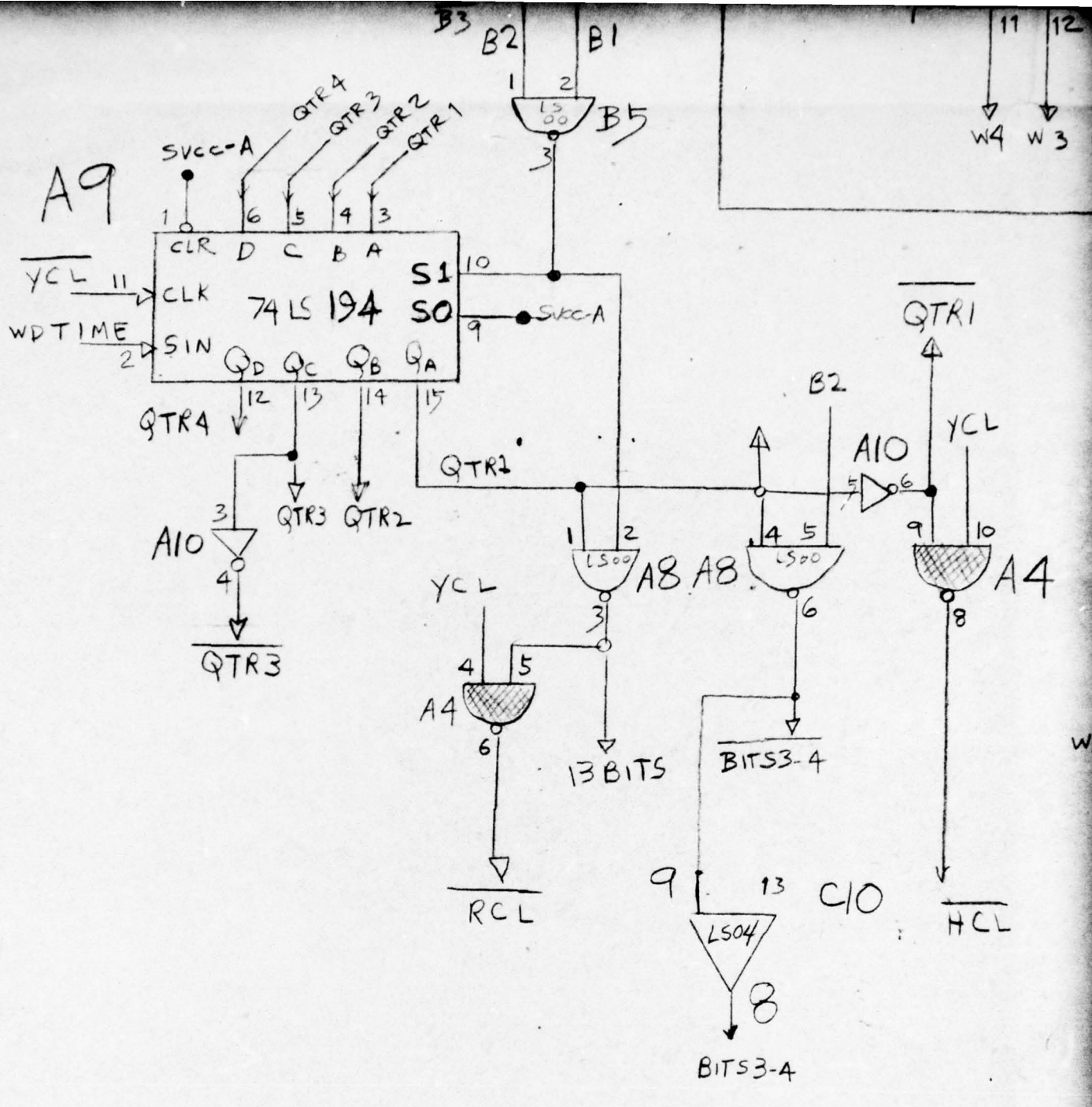


ITCLK X2

D

C





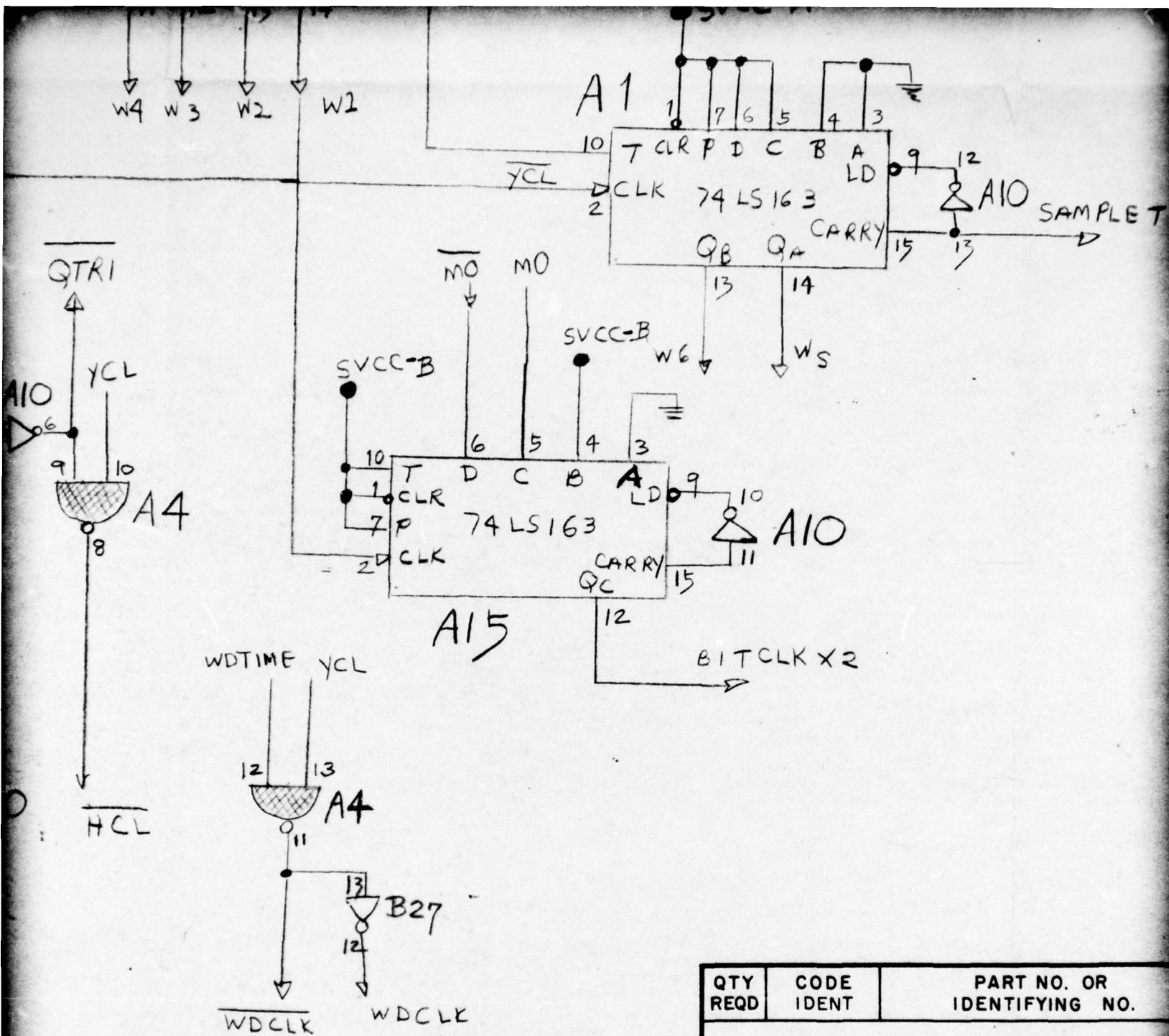
A9



B

4

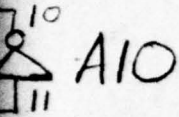
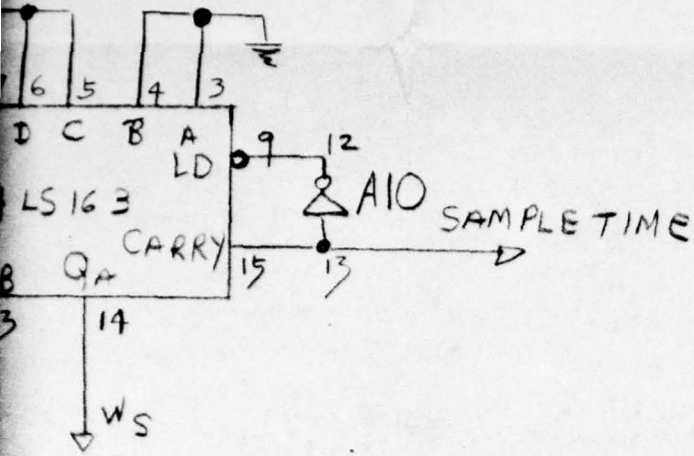
NEXT ASSY
APPL



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.

		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES: ANGLES ± FRACTIONS ± 3 PLACE DECIMALS ± 2 PLACE DECIMALS ±	DR	
			CHK	
		MATERIAL:	A	
			P	
			D	
			RELEASED	
		NEXT ASSY USED ON APPLICATION		
			CONTRACT NO.	

1 0 CVSD_b
 1 1 ADM



ITCLK X2

C

B

A

G

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
----------	------------	-----------------------------	-----------------------------	----------

PARTS LIST

APPROVED	DR	
	CHK	
	A	
	P	
	D	
RELEASED		
CONTRACT NO.		

NEWTON, MASSACHUSETTS 02195

codex corporation

DRAWING TITLE

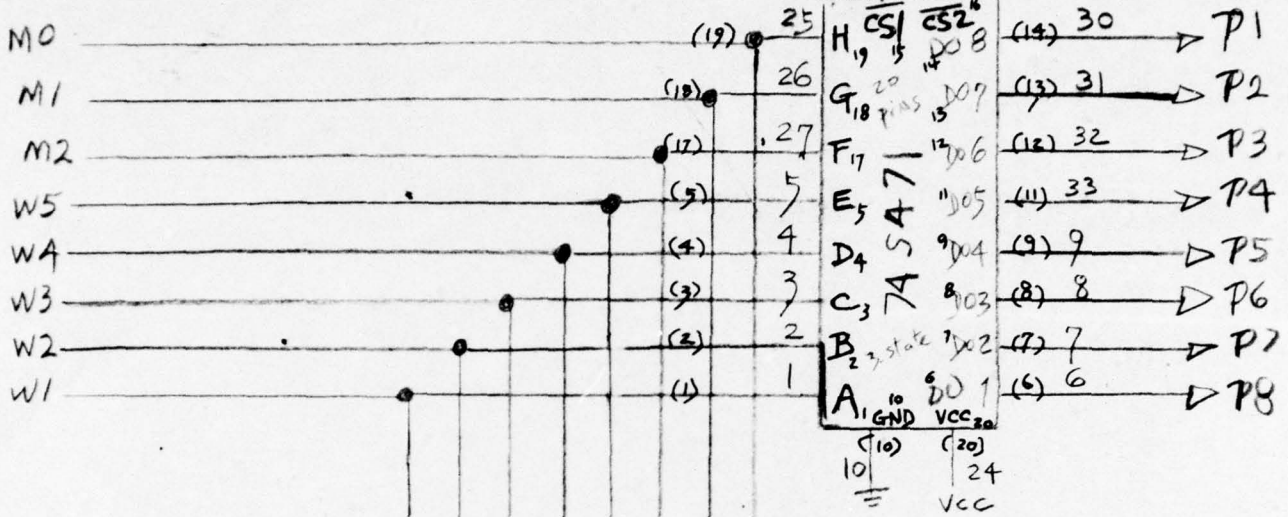
MODE CONTROL & TIMING CIRCUITS

SIZE	CODE IDENT NO.	DRAWING NO.
C	25420	FIG. 3-9
SCALE	SHEET 1 OF 10	

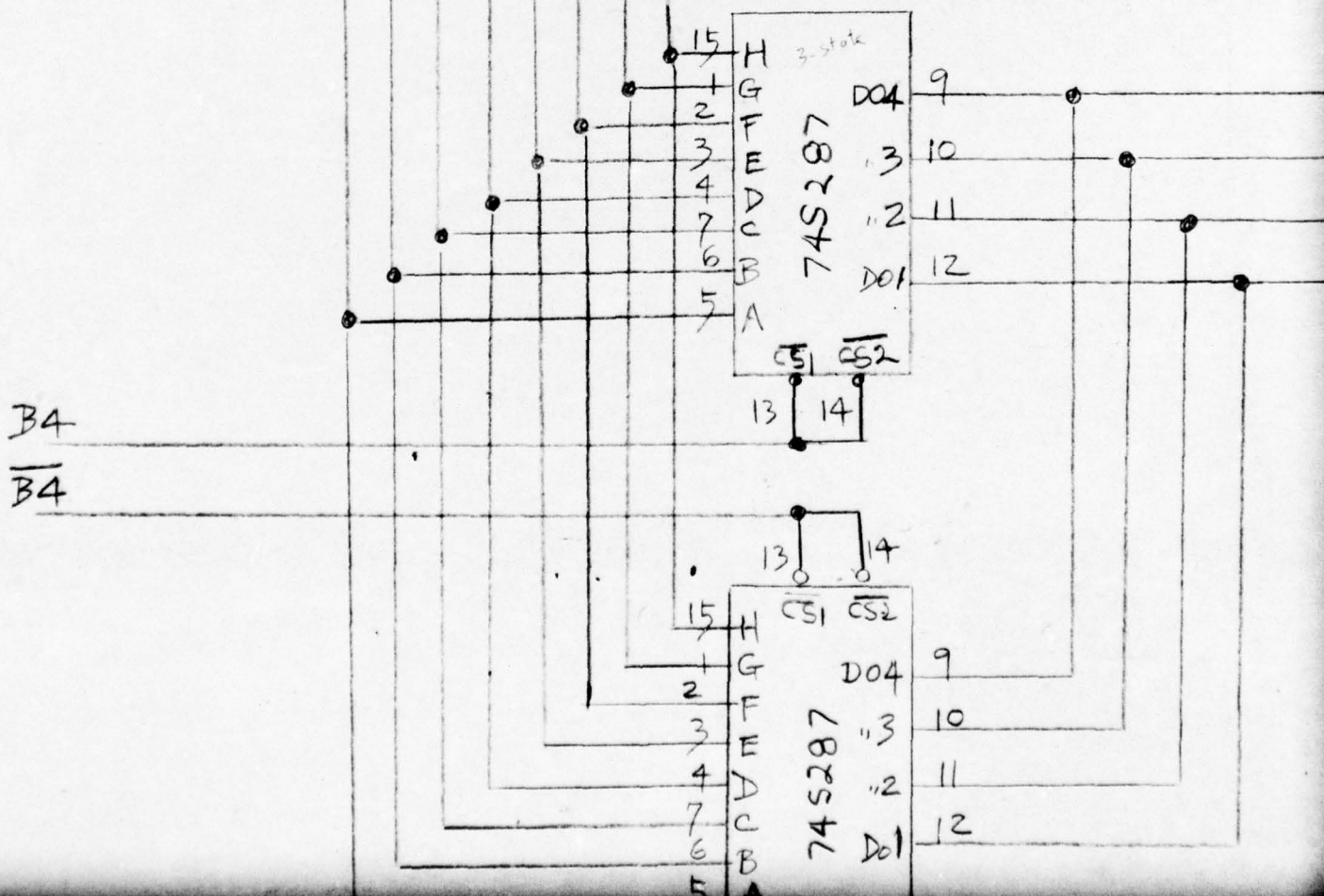
74S471
VCC = 20
GND = 10

J1

PROGRAM ROM



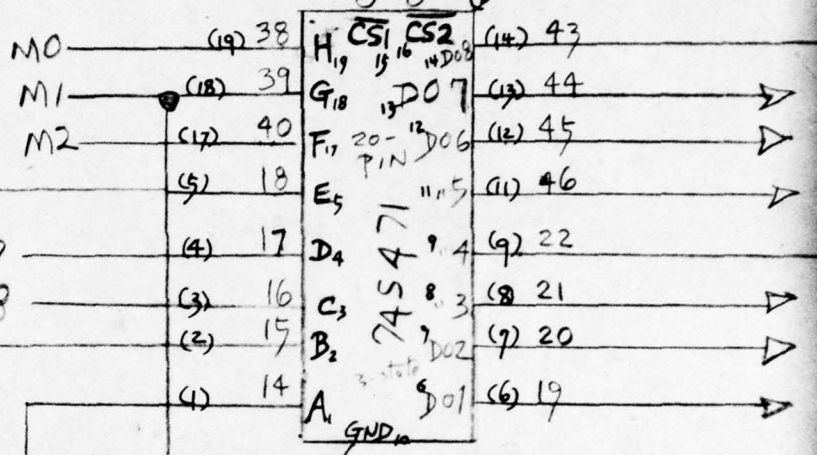
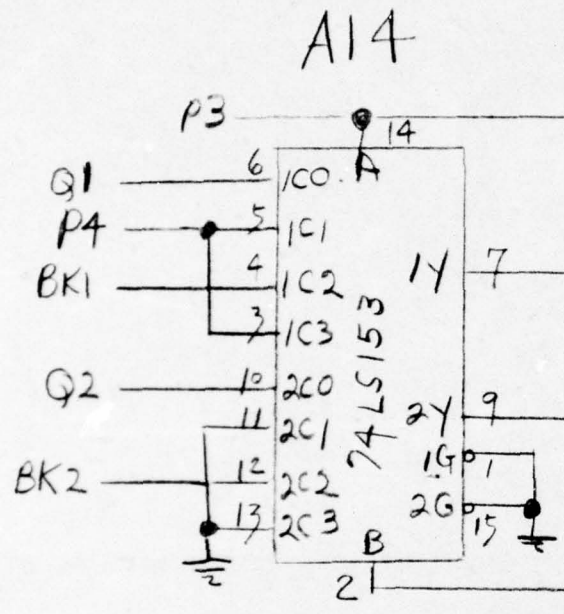
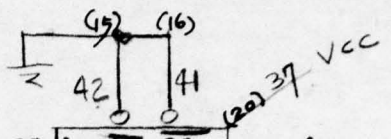
A13 WRITE ADDRESS ROM



2

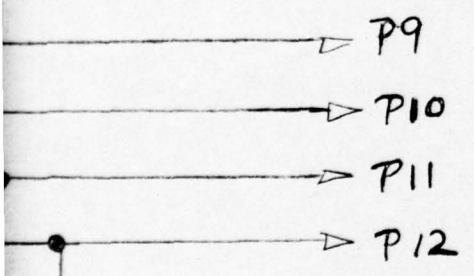
Q5/BS
P5

J1



PARAMETER ROM

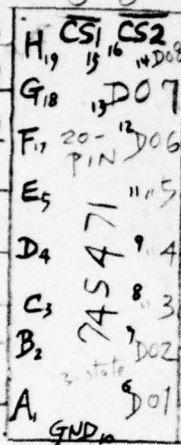
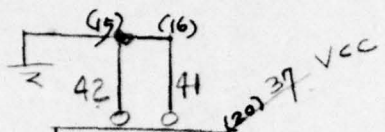
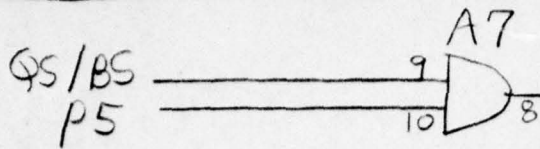
ROM



REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

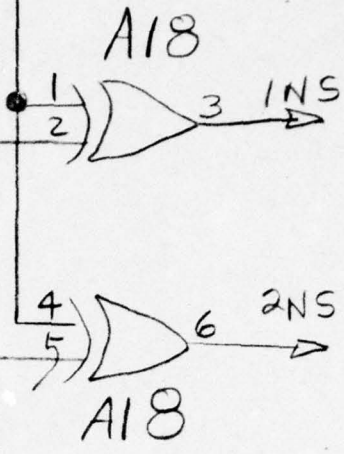
3

J1



PARAMETER ROM

- IN1
- IN2
- IN3
- 2N1
- 2N2
- 2N3

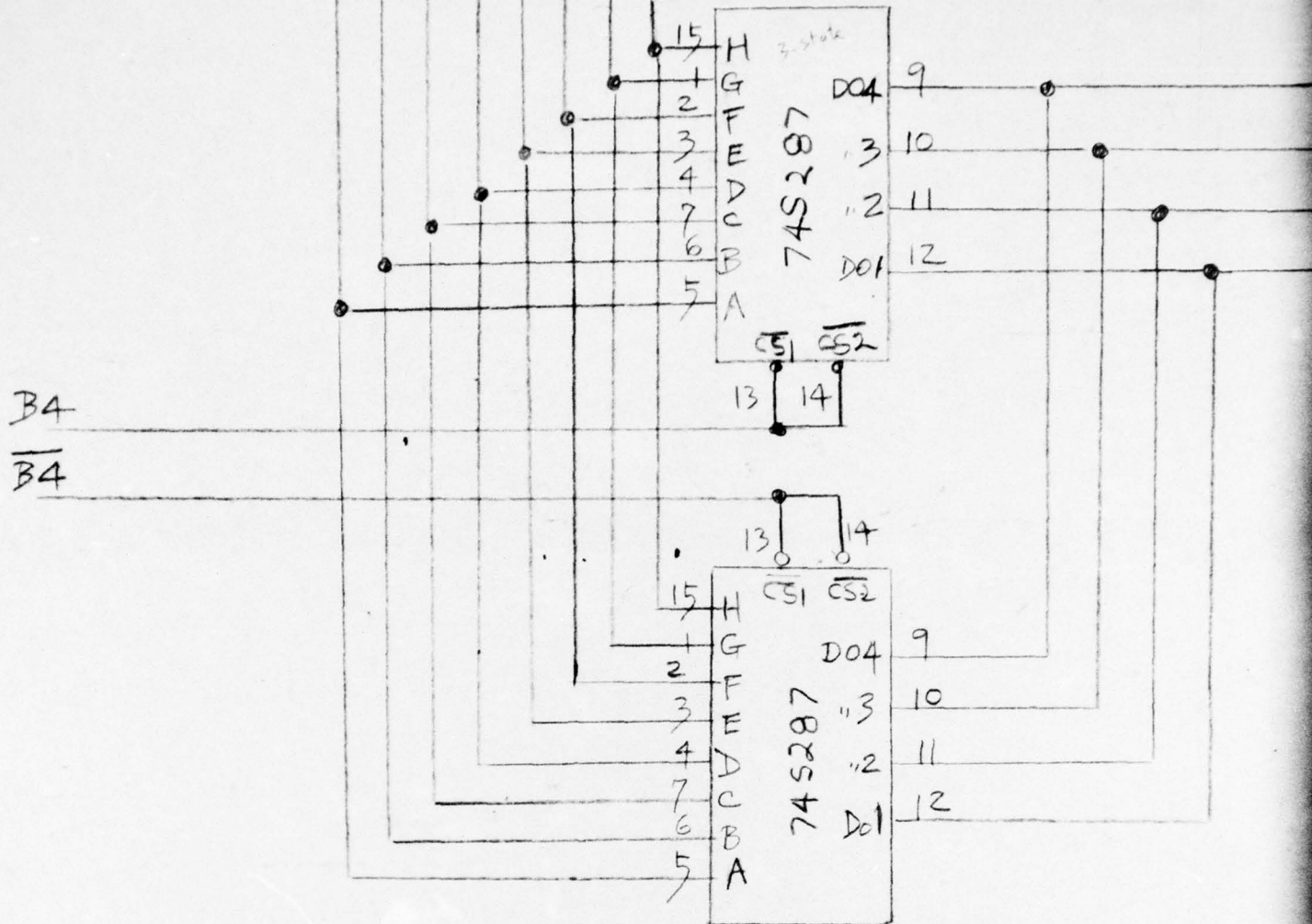


D

C



A13 WRITE ADDRESS ROM

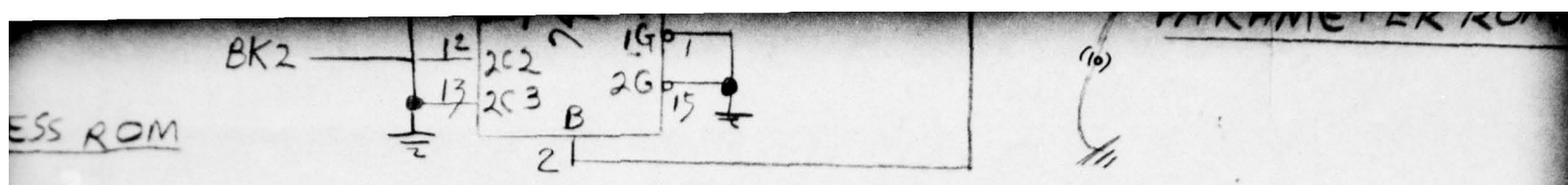


A12 READ ADDRESS ROM

4

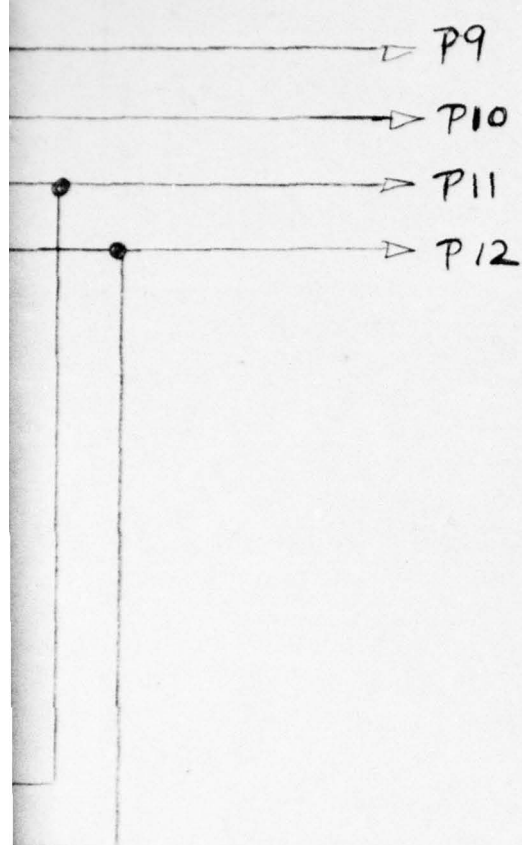
NEXT ASSY

APP



ESS ROM

PARAMETER ROM



S ROM

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	
			PAF

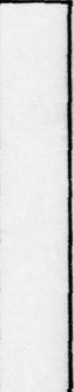
PARAMETER ROM

(10)

C

B

A



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
----------	------------	-----------------------------	-----------------------------	----------

PARTS LIST

SPECIFIED INCHES	DR	
	CHK	
	A	
	P	
	P	
	RELEASED	

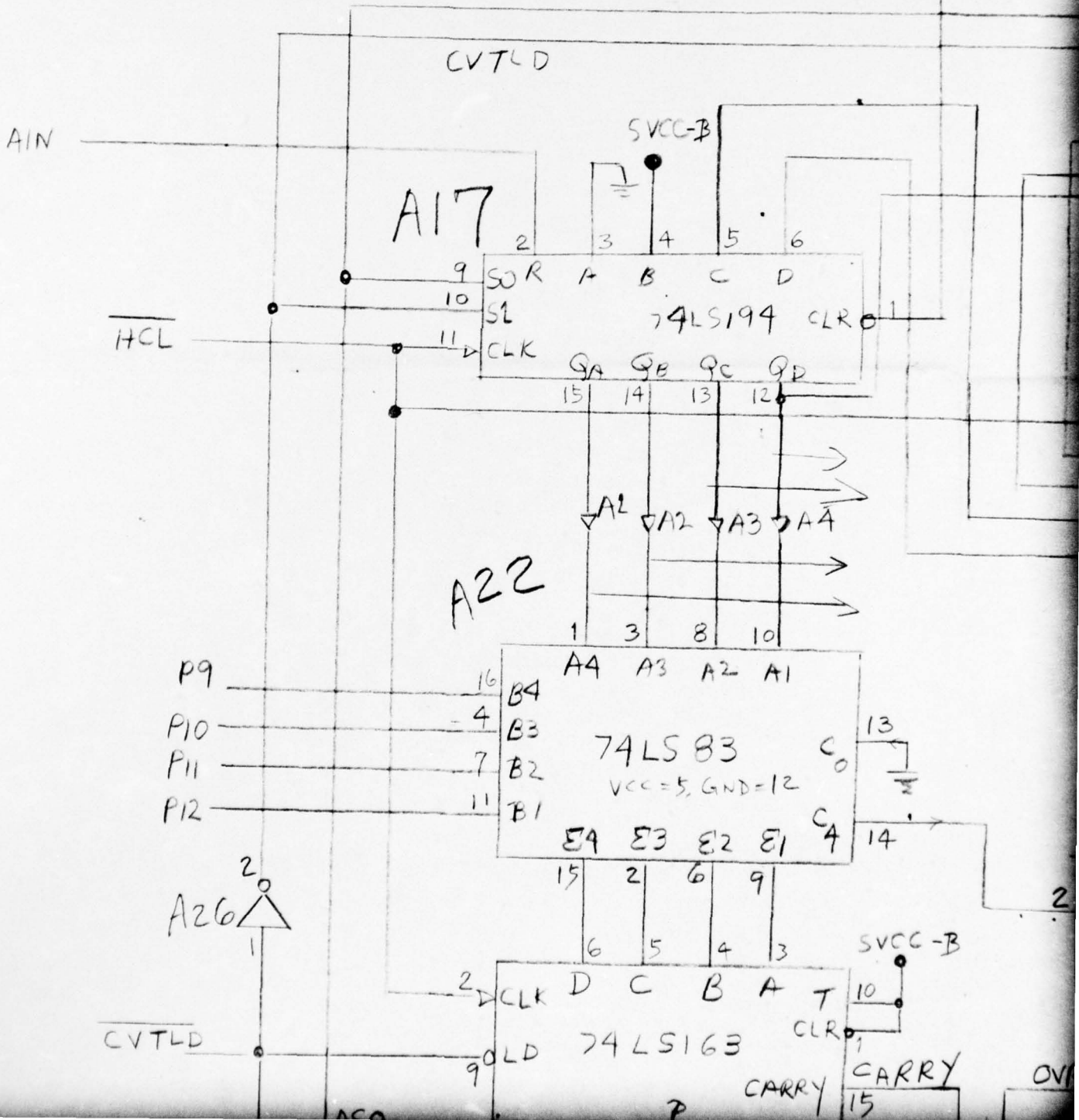
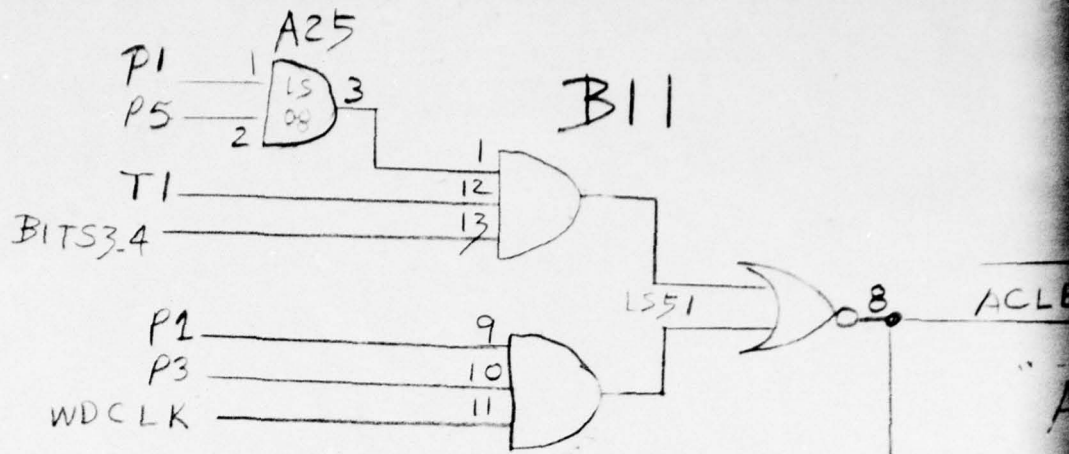
NEWTON, MASSACHUSETTS 02195

DRAWING TITLE
 PROGRAM & PARAMETER
 ROM's

CONTRACT NO.	SIZE C	CODE IDENT NO. 25420	DRAWING NO. FIG. 3-10
	SCALE		SHEET OF

6

D



3

2

2

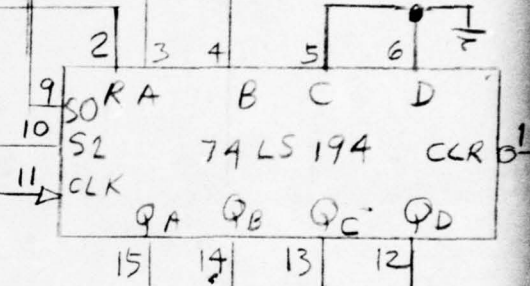
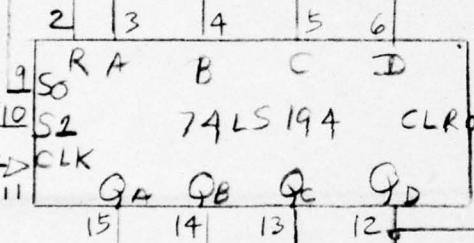
ZONE LTR

ACLEAR

"A" REGISTER

ASO

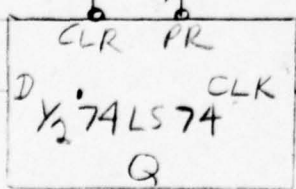
A16



A5 A6 A7 A8

A9 A10 A11 A12

SVCC-B



A19

A20

YCL

CVTLD

OVRFLO

ARRY

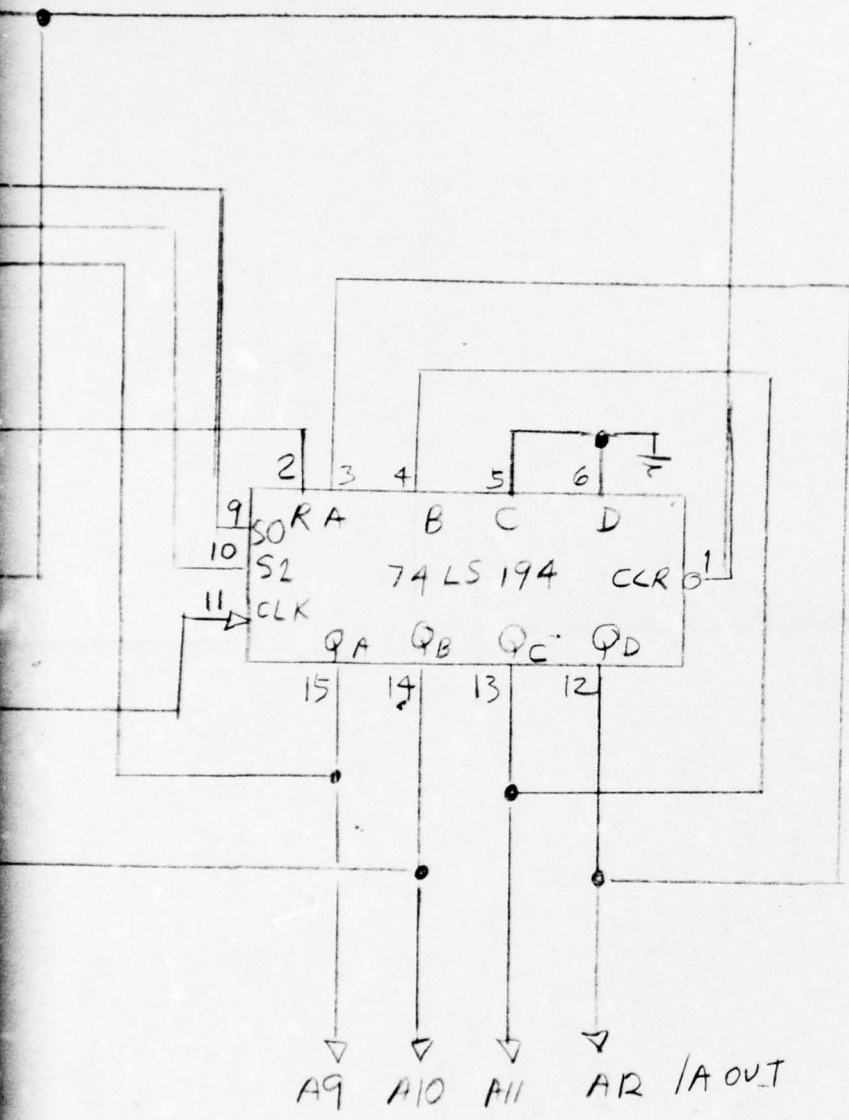
REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

3

D

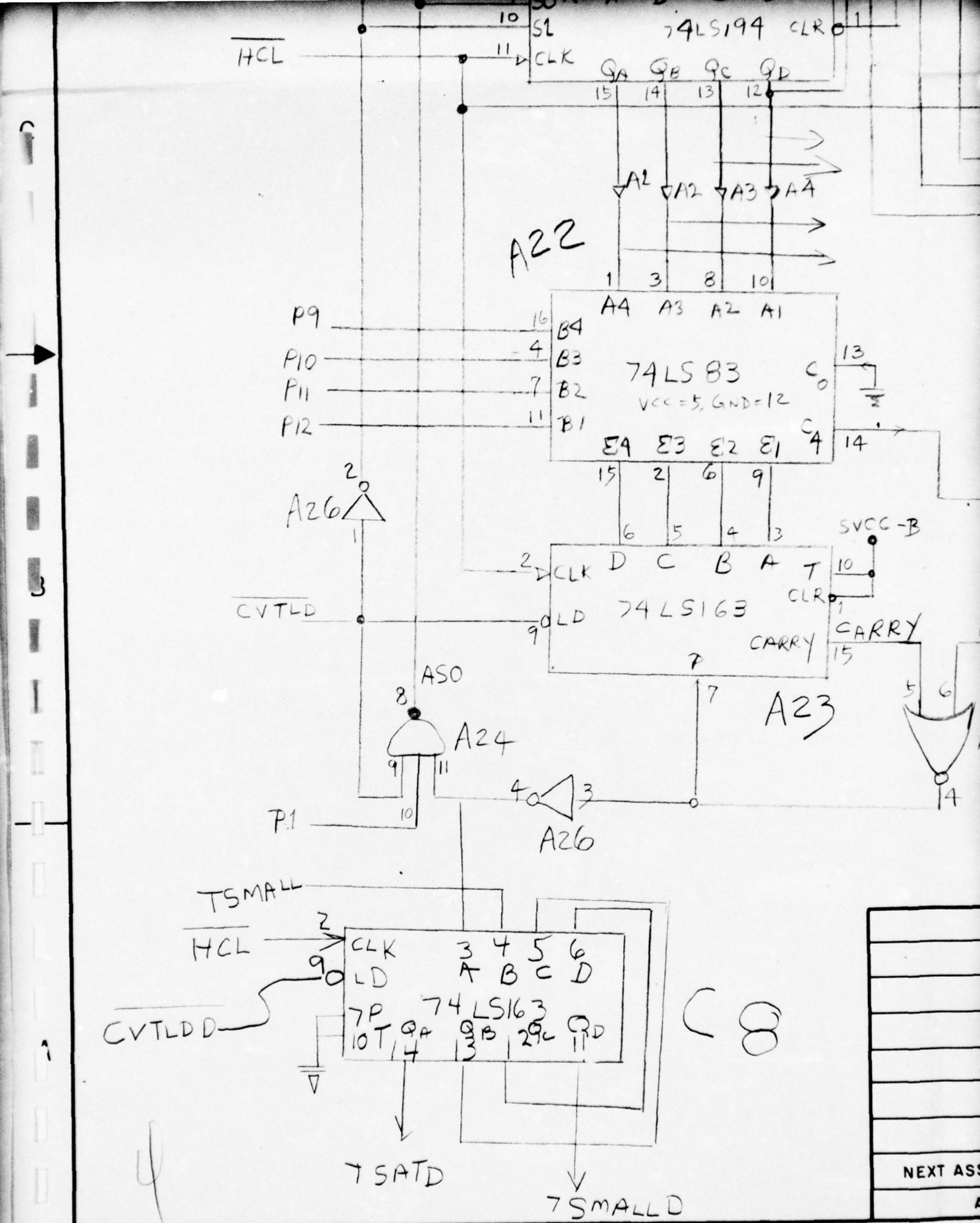
C

A21

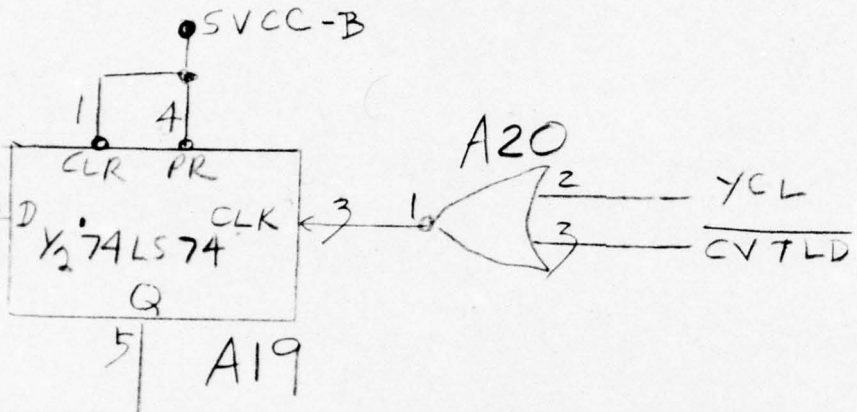
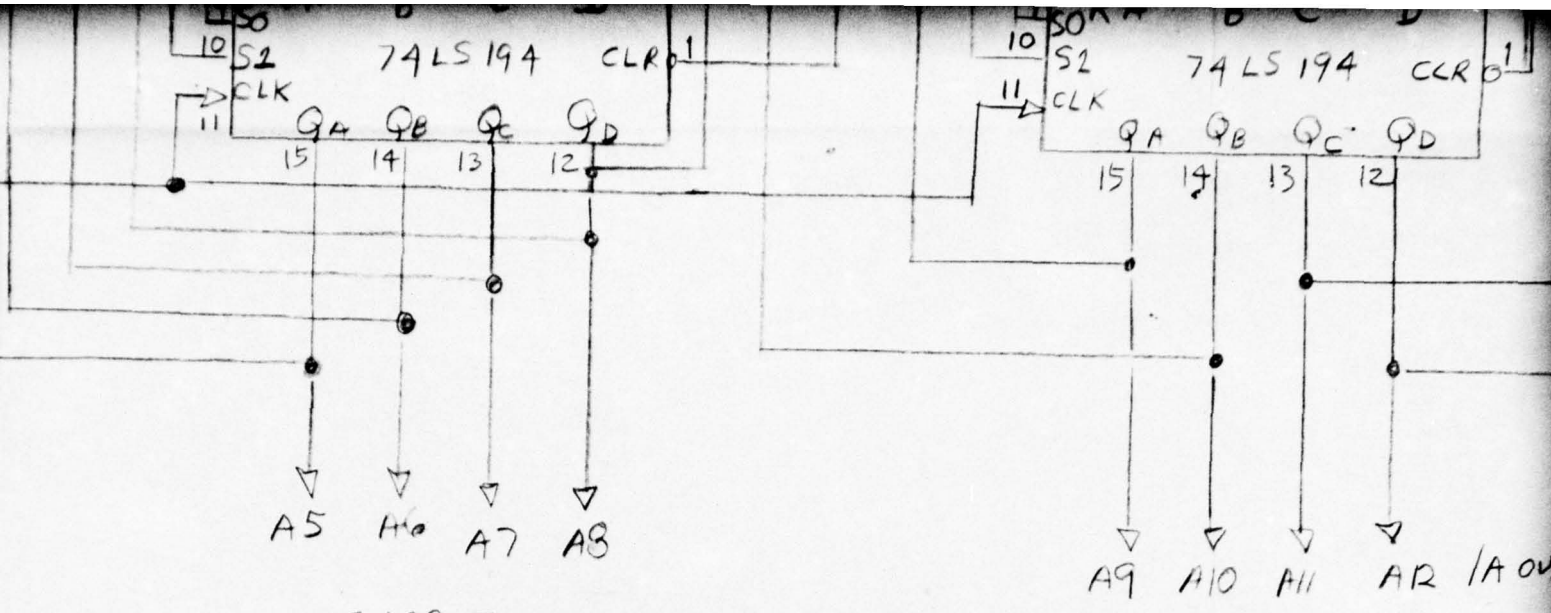


YCL
 CVTLD



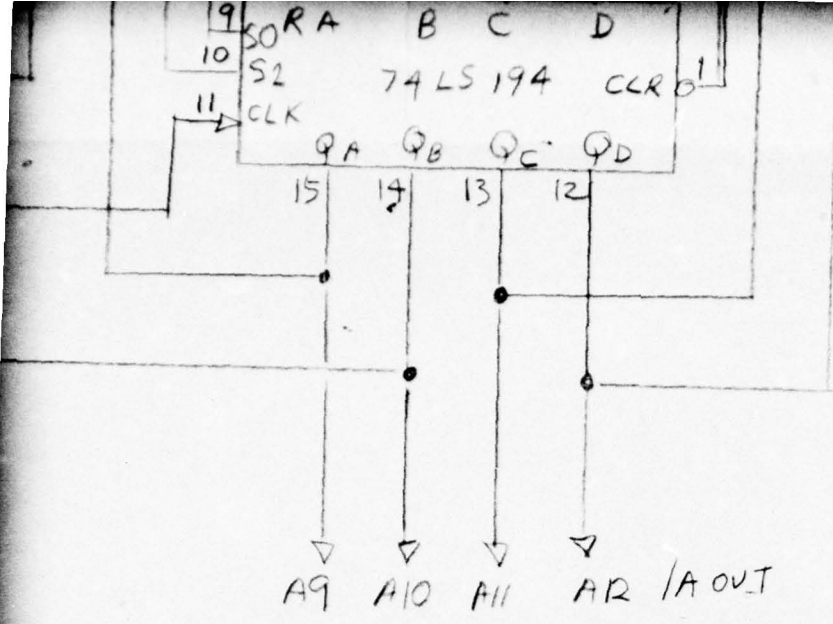


NEXT ASSY	
AP	



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	PART
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES: ANGLES ± FRACTIONS ± 3 PLACE DECIMALS ± 2 PLACE DECIMALS ±		DR	DRAWING C
MATERIAL:		CHK	
NEXT ASSY		A P P D	
USED ON		RELEASED	
APPLICATION		CONTRACT NO.	SIZE C
			CODE 2
			SCALE

5



A21

YCL
CVTLD

C
B

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
----------	------------	-----------------------------	-----------------------------	----------

PARTS LIST

SPECIFIED INCHES	DR	
	CHK	
	A	
	P	
	D	
	RELEASED	

NEWTON, MASSACHUSETTS 02195

codex corporation

DRAWING TITLE

A REGISTER,
CONVERT CIRCUITRY

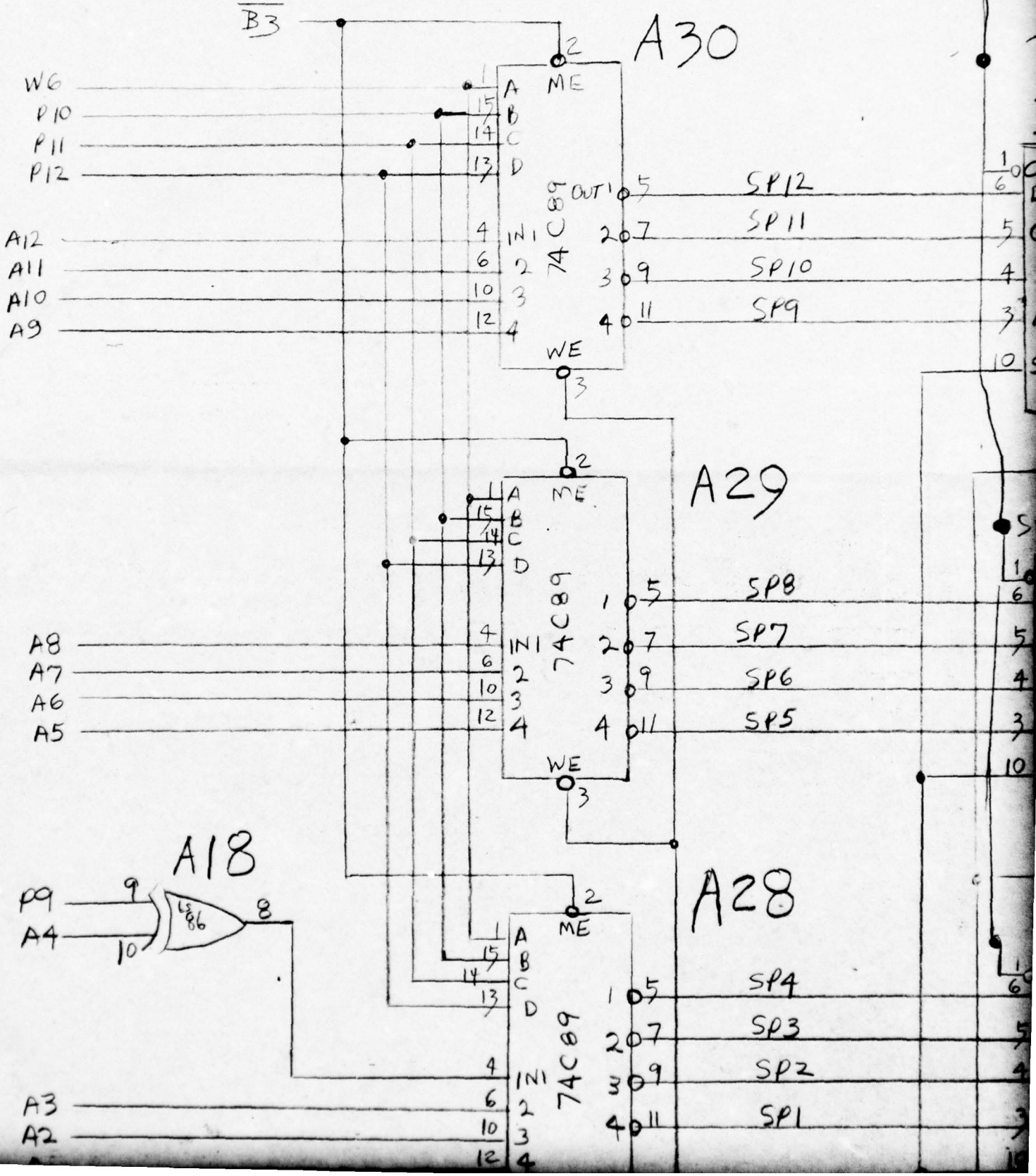
CONTRACT NO.	SIZE	CODE IDENT NO.	DRAWING NO.
	C	25420	FIG. 3-11
	SCALE		SHEET OF

A
6

1

D

R C



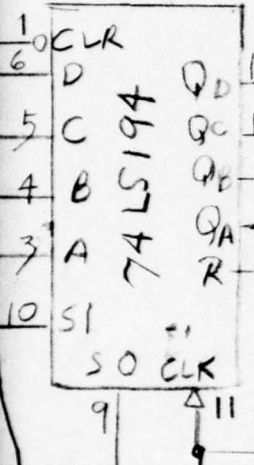
B

2

R CLEAR

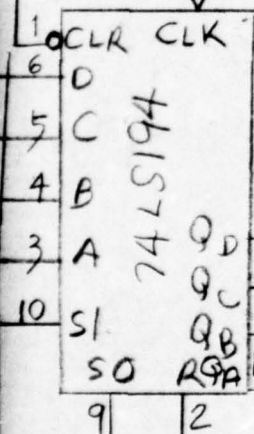
B30

B25

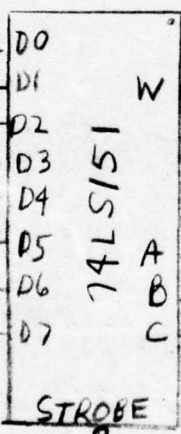
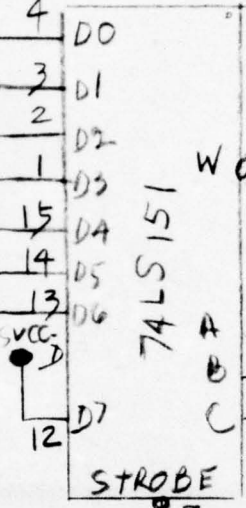


R12

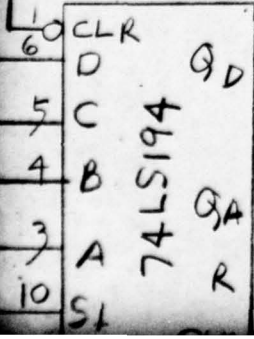
R13



B29



B24



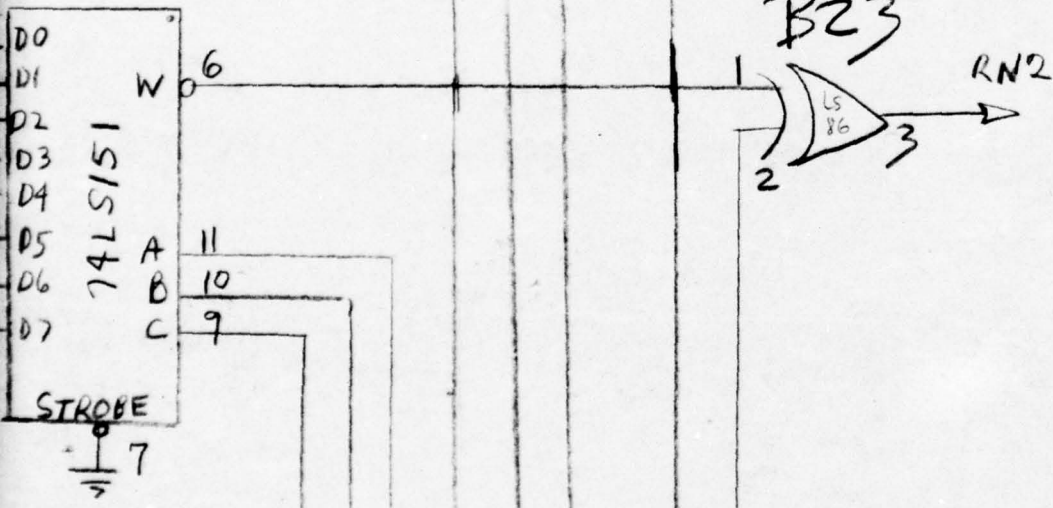
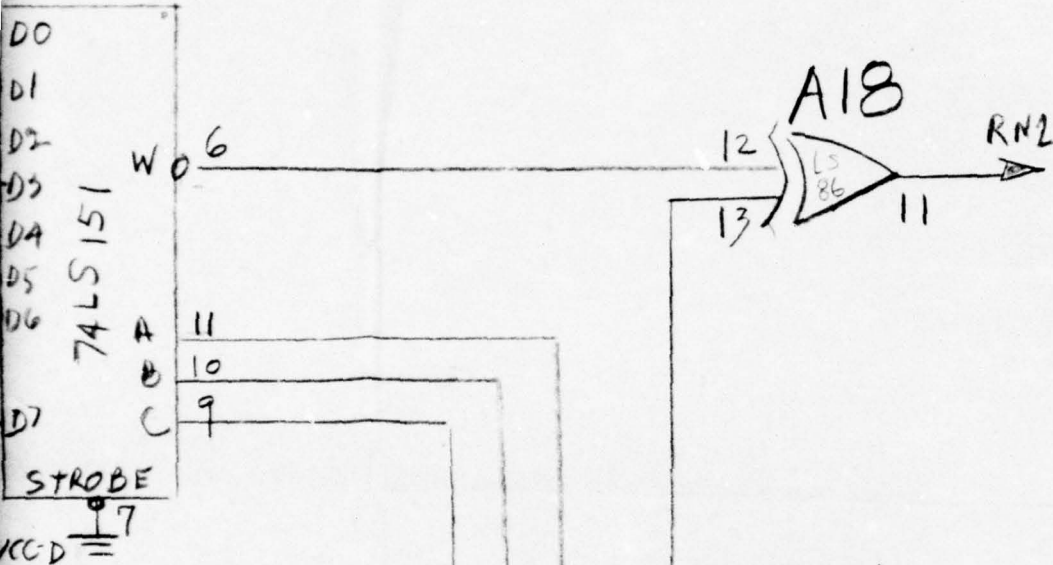
B28



- 2N1
- 2N2
- 2N3
- 1N1
- 1N2
- 1N3
- 1NS
- 2NS

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

B25



B24

RN1
 RN2
 RN3
 IN1
 IN2
 IN3
 INS
 2NS

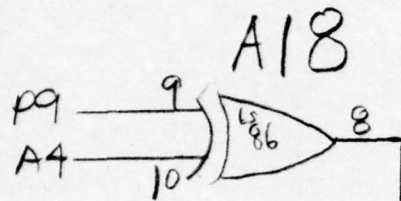
3

D

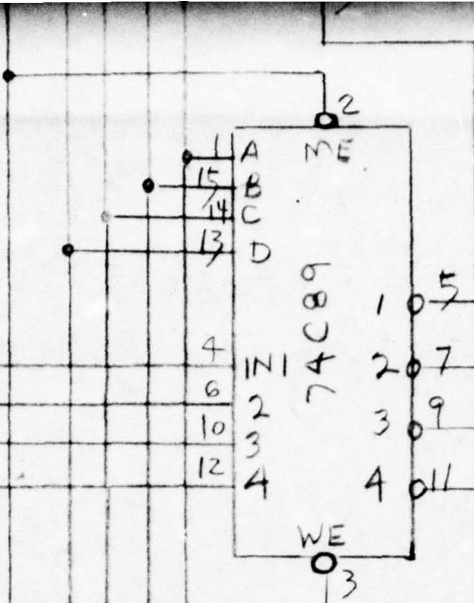
C

←

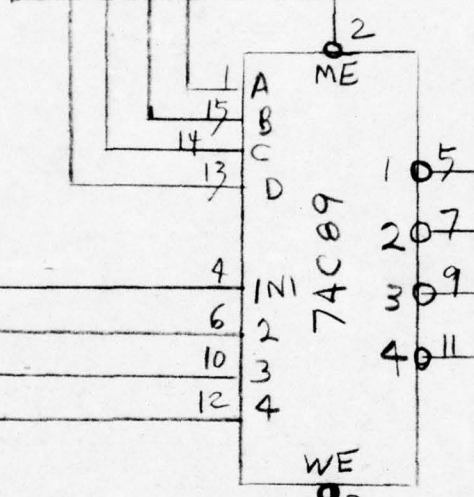
A8
A7
A6
A5



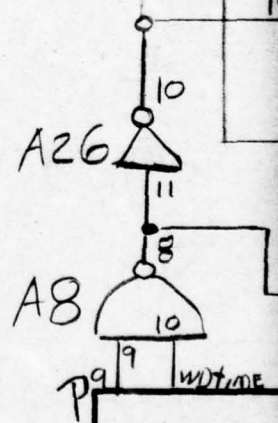
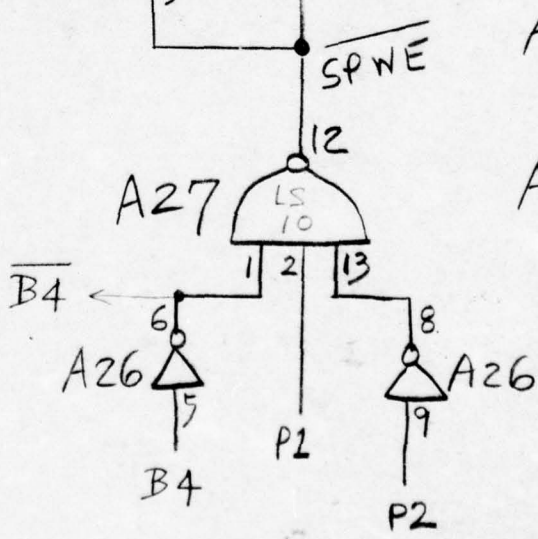
A3
A2
A1



A29

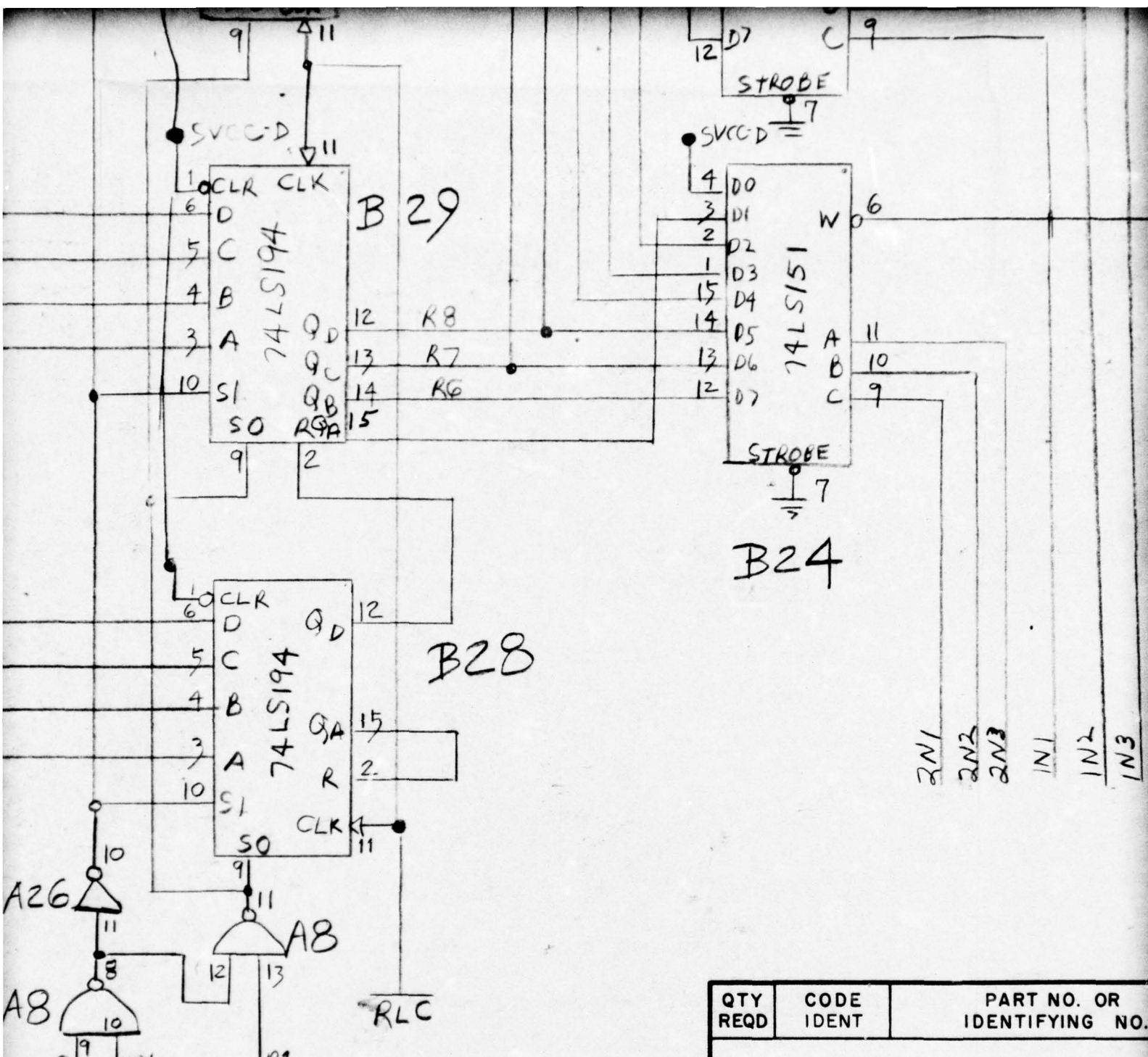


A28



NEXT ASSY

AF



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.

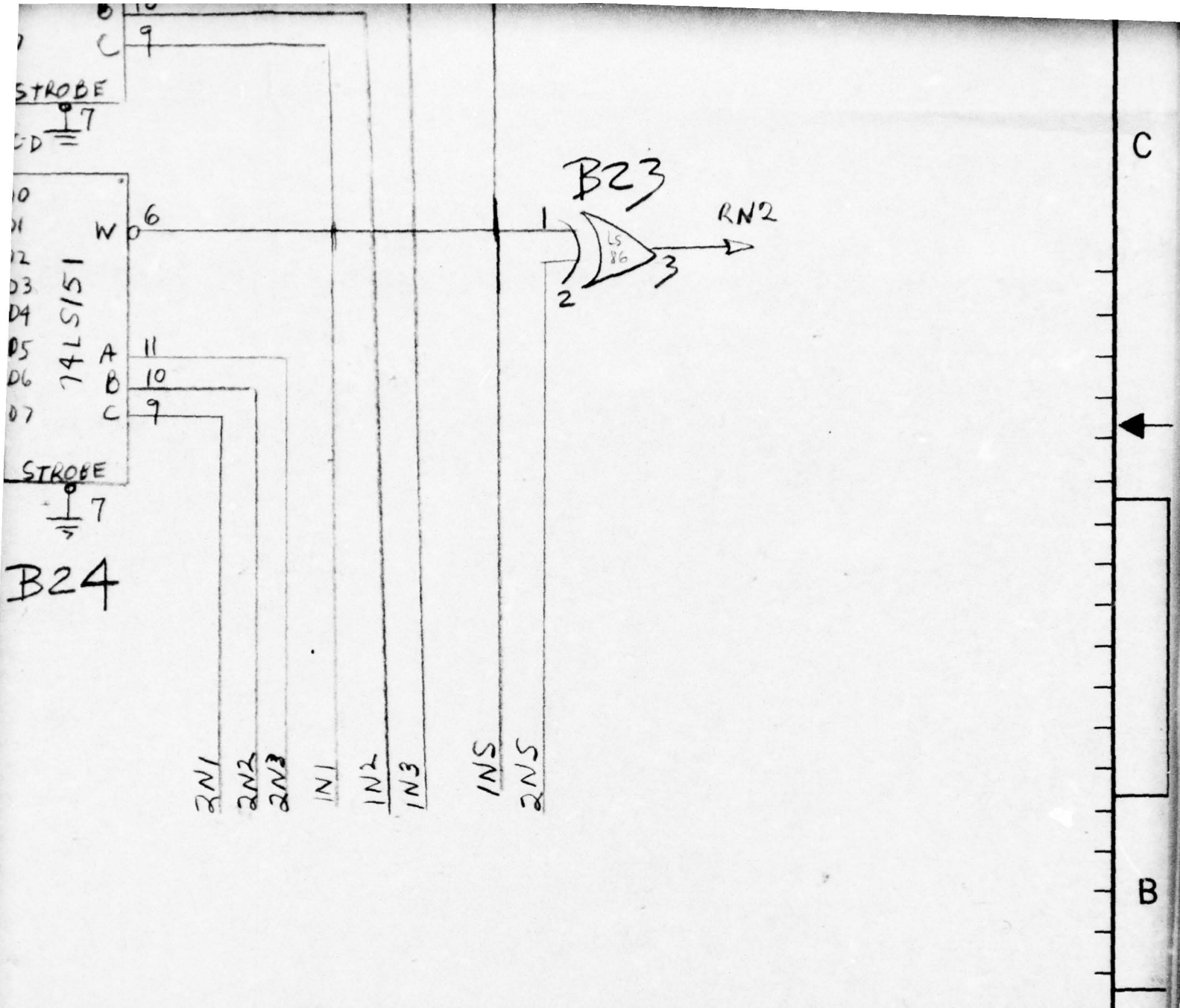
UNLESS OTHERWISE SPECIFIED
 DIMENSIONS ARE IN INCHES
 TOLERANCES: ANGLES ±
 FRACTIONS ±
 3 PLACE DECIMALS ±
 2 PLACE DECIMALS ±

MATERIAL:

DR	
CHK	
A	
P	
P	
D	
RELEASED	
CONTRACT NO.	

NEXT ASSY USED ON

APPLICATION



C

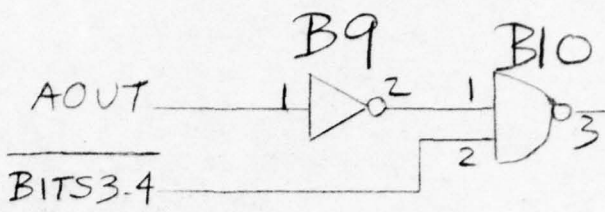
B

A

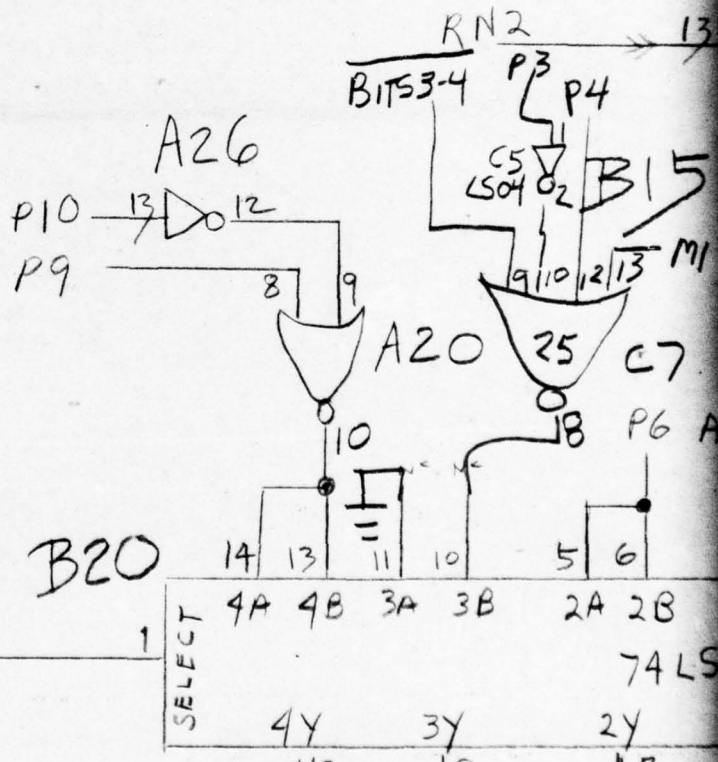
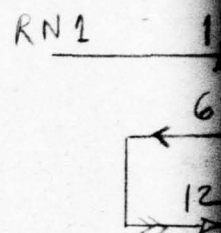
6

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED SHEETS	DR			NEWTON, MASSACHUSETTS 02195 codex corporation
	CHK			
	A P P D			
	RELEASED			
DRAWING TITLE			SCRATCHPAD, R REGISTER, MUXES	
CONTRACT NO.		SIZE	CODE IDENT NO.	DRAWING NO.
		C	25420	FIG. 3-12
SCALE			SHEET	OF

D



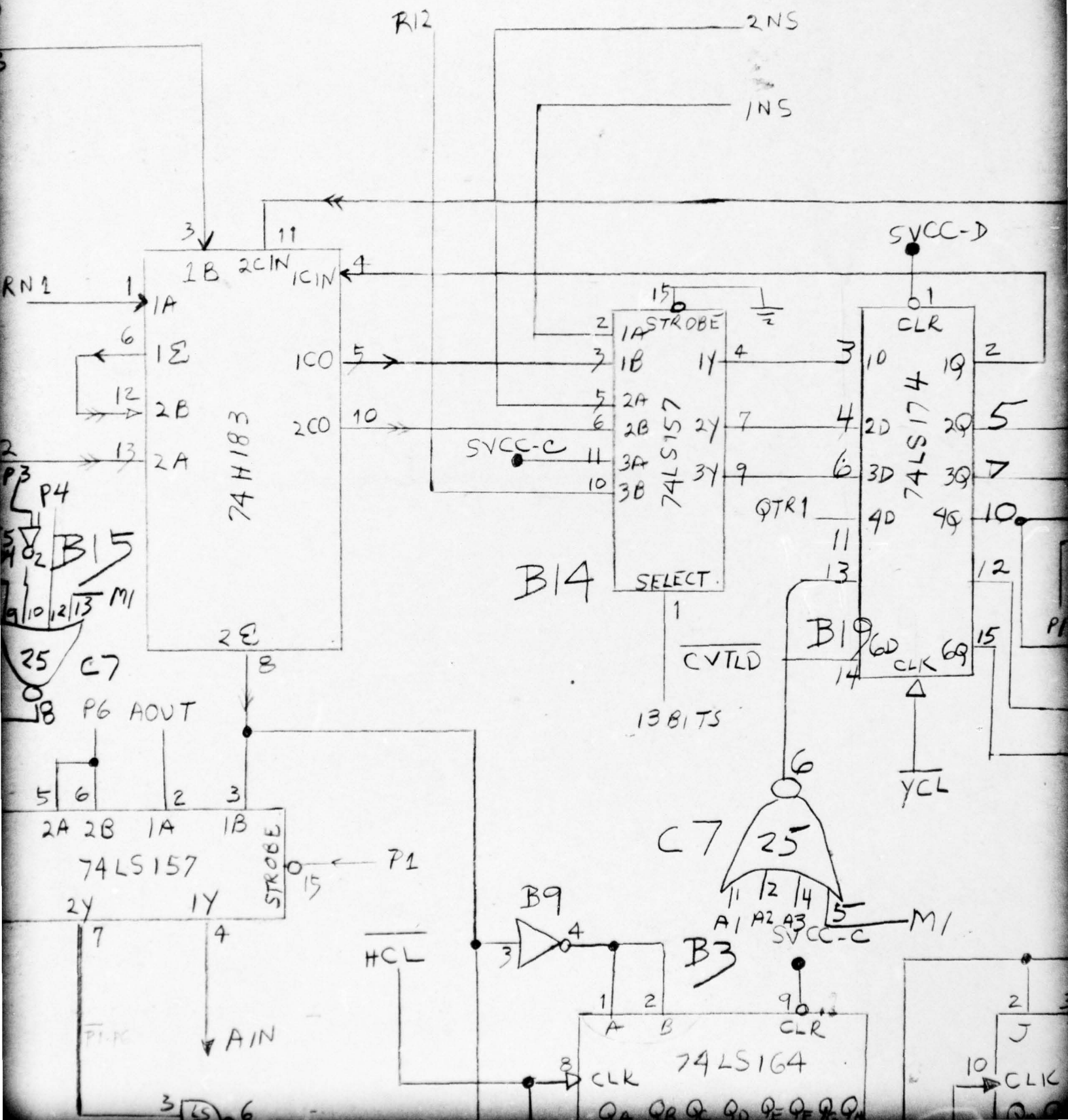
C



B



2

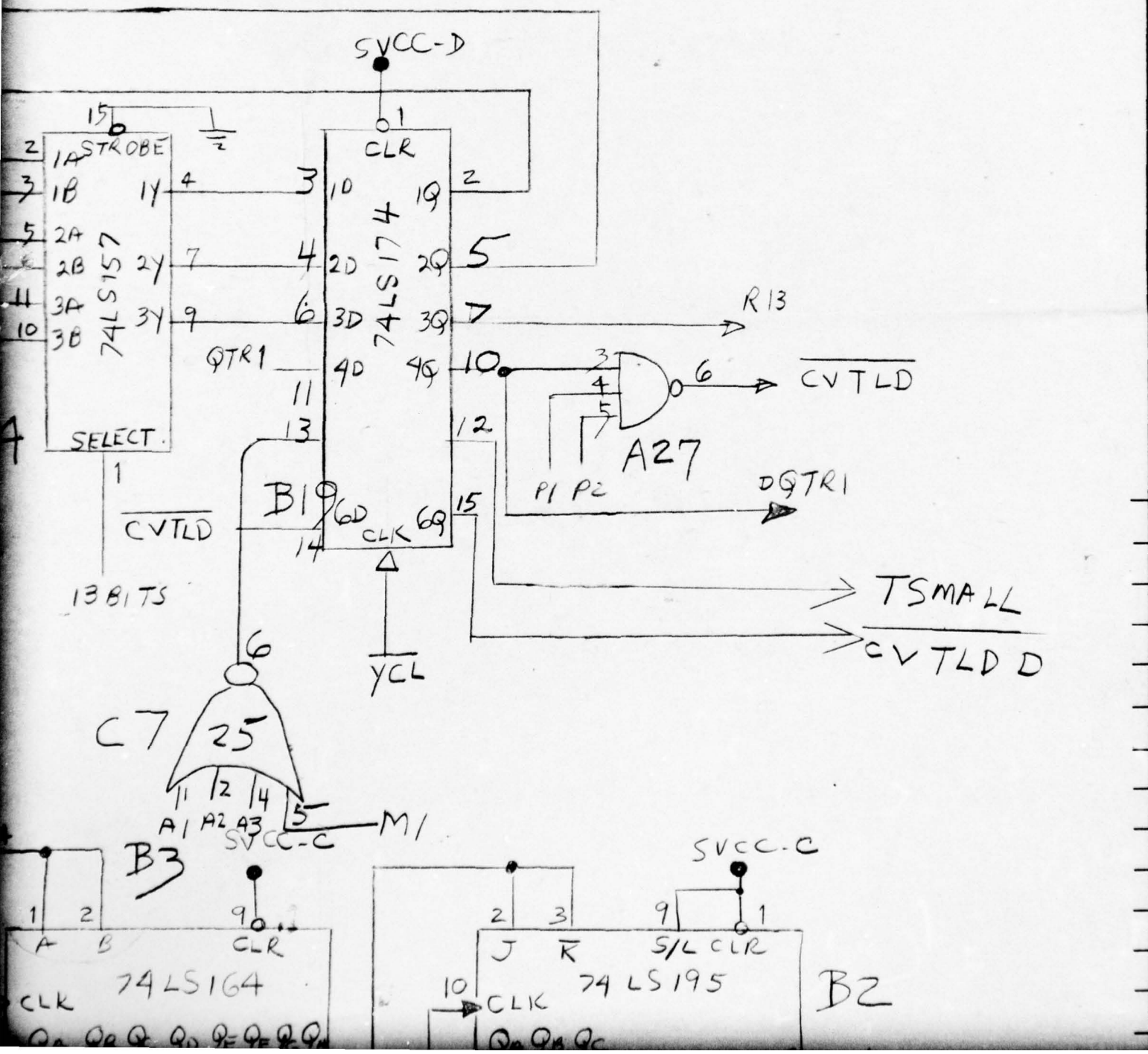


REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

3

2NS

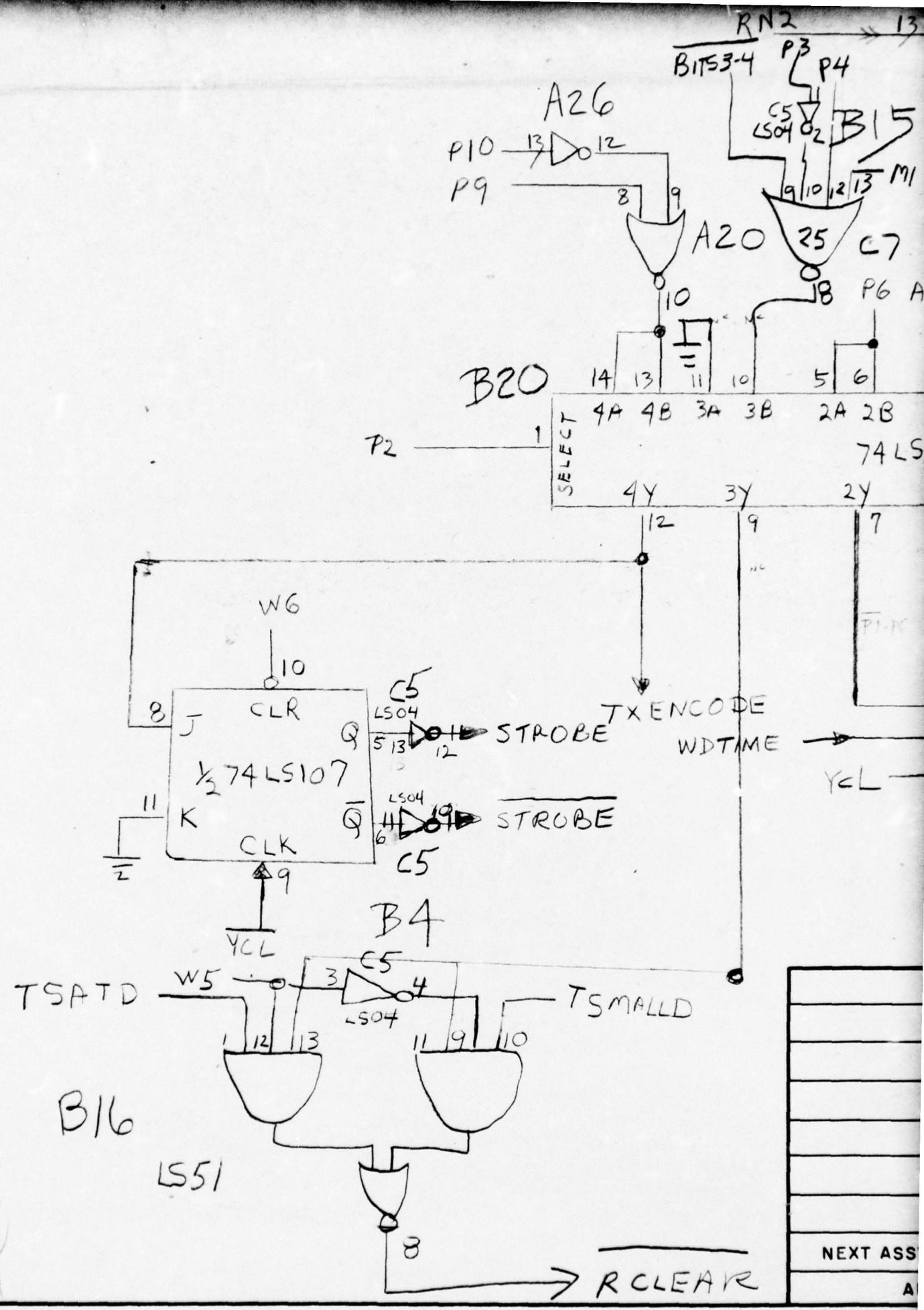
1NS



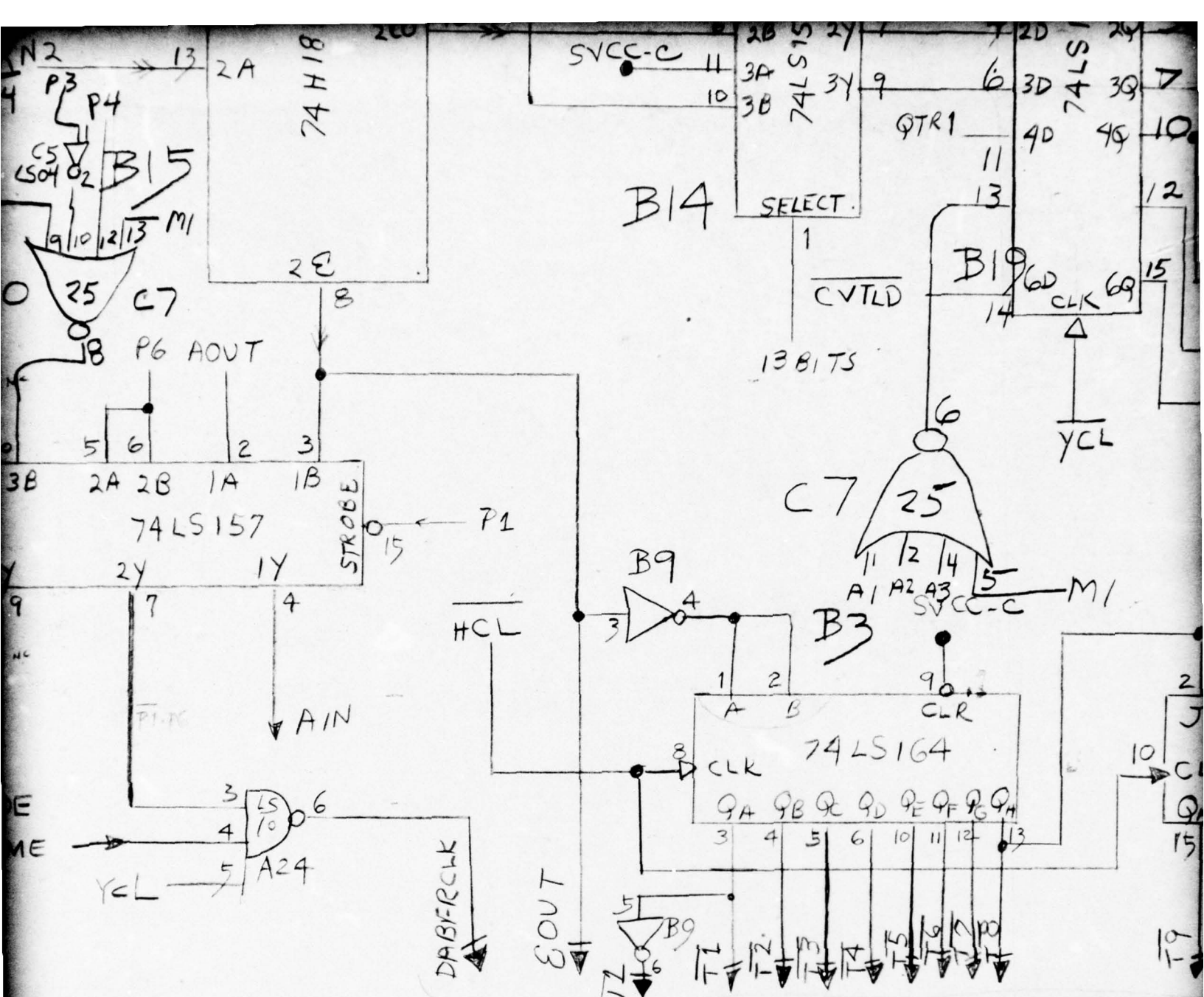
C

B

A



NEXT ASS
A



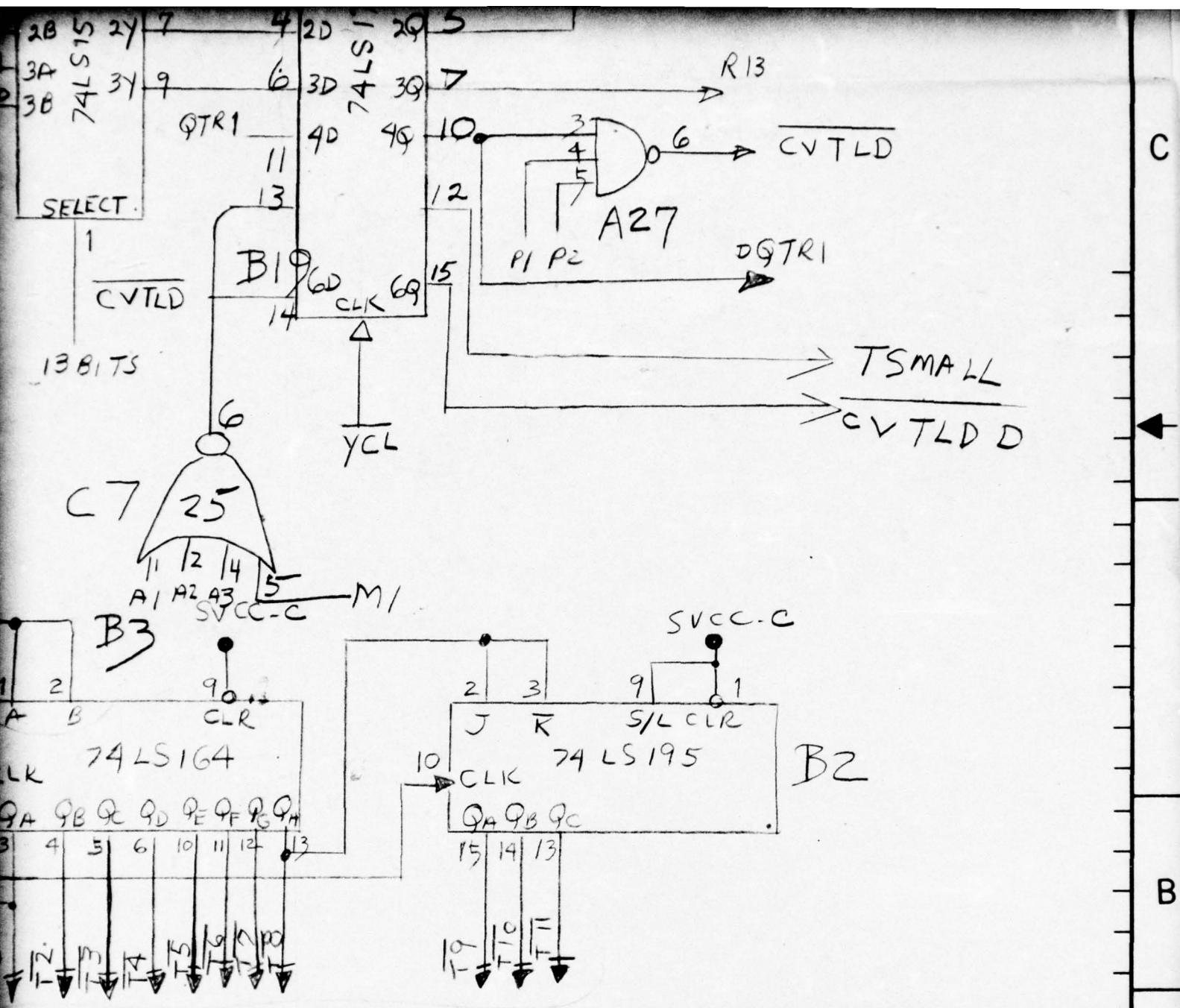
QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.
----------	------------	-----------------------------

TO V/A BUFFER

		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES: ANGLES ± FRACTIONS ± 3 PLACE DECIMALS ± 2 PLACE DECIMALS ±	DR	
			CHK	
		MATERIAL:	A	
			P	
			D	
			RELEASED	
			CONTRACT NO.	
NEXT ASSY	USED ON			
APPLICATION				

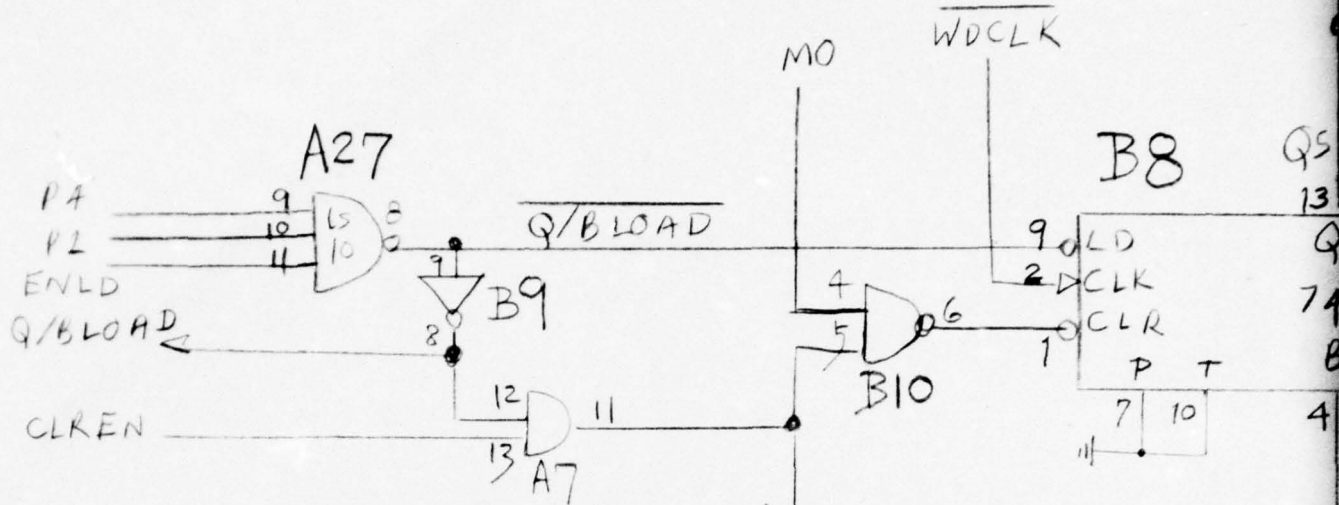
5



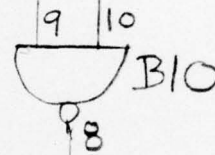


QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED DIMENSIONS	DR		 DRAWING TITLE ADDER, T REGISTER	
	CHK			
	A P P D			
	RELEASED			
CONTRACT NO.		SIZE C	CODE IDENT NO. 25420	DRAWING NO. FIG. 3-13
		SCALE	SHEET OF	

D

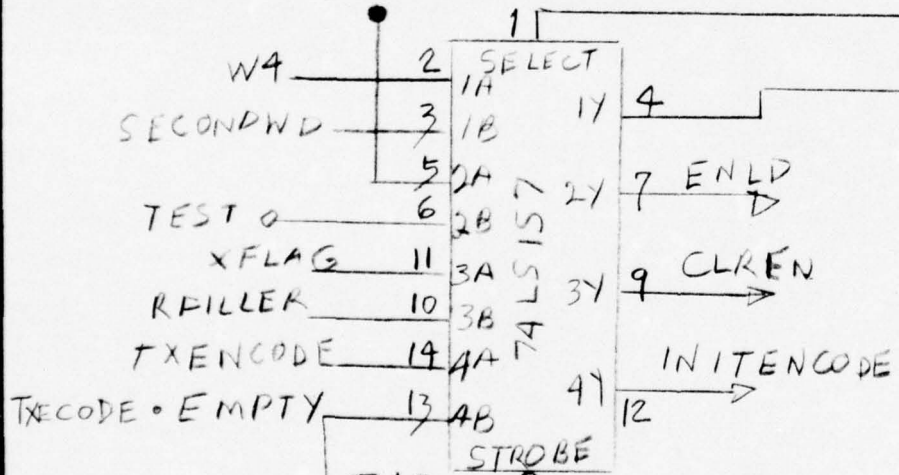


P6



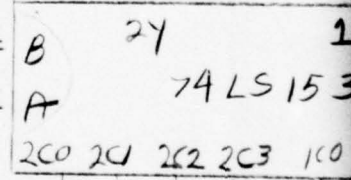
BIN

SVCC-D



W4
SECONDDWD
TEST 0
XFLAG
RILLER
TXENCODE
TXENCODE · EMPTY

ENLD
CLREN
INITENCODE
STROBE



B17

RFIFOUT

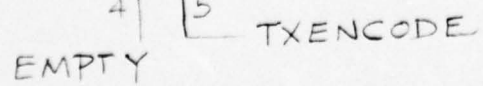
C

B

COMP

B18

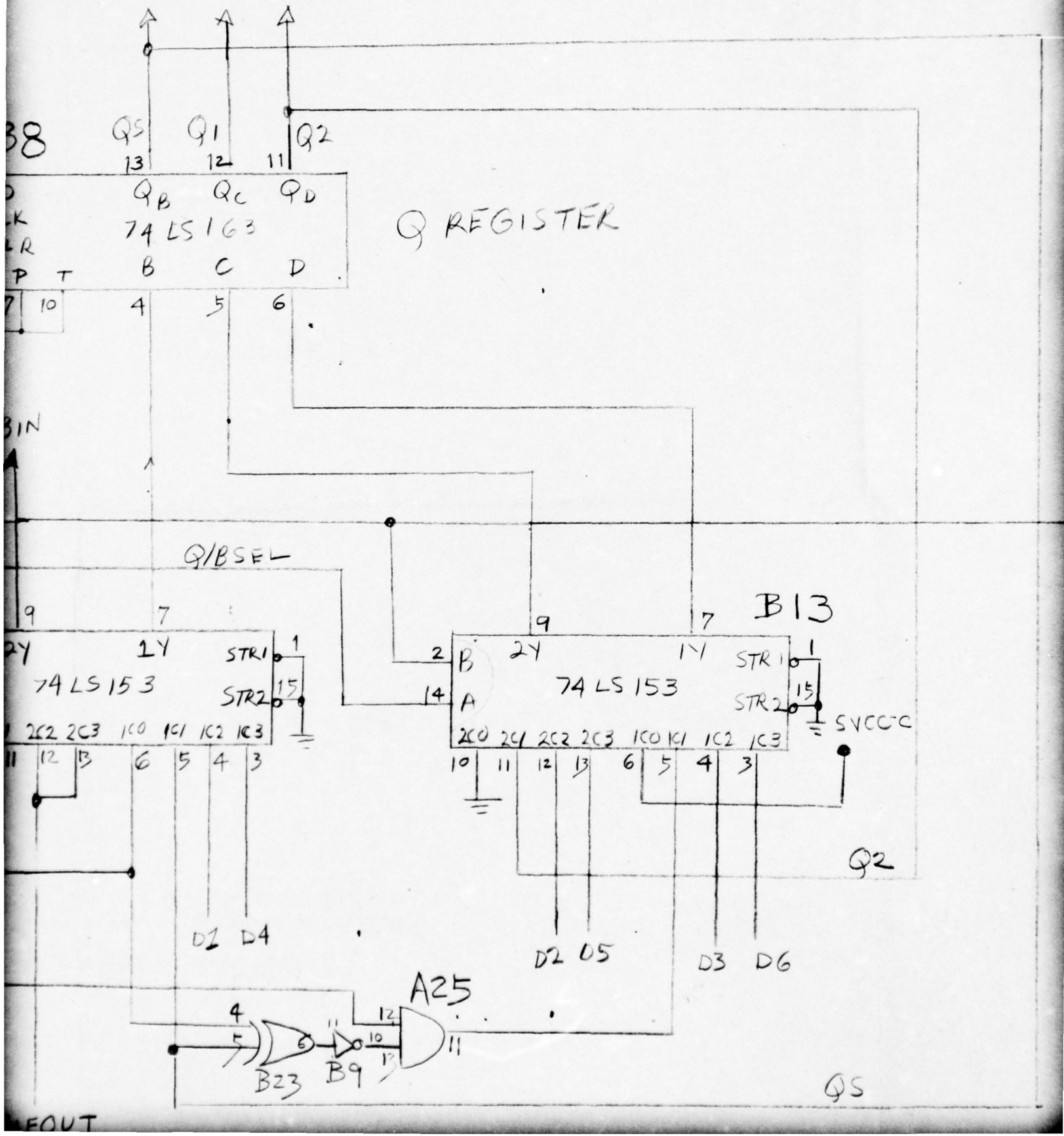
A25



EMPTY

TXENCODE

2

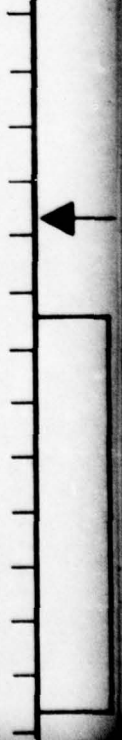
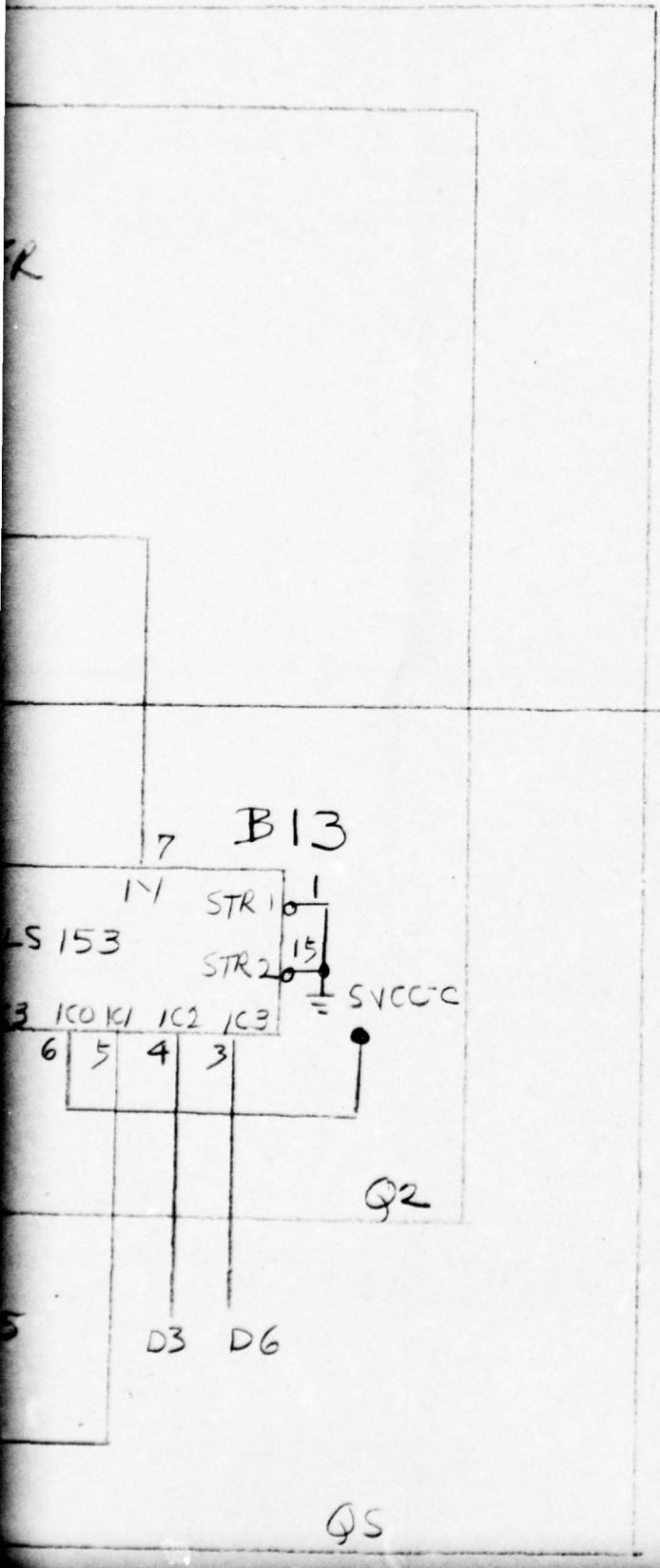


REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

3

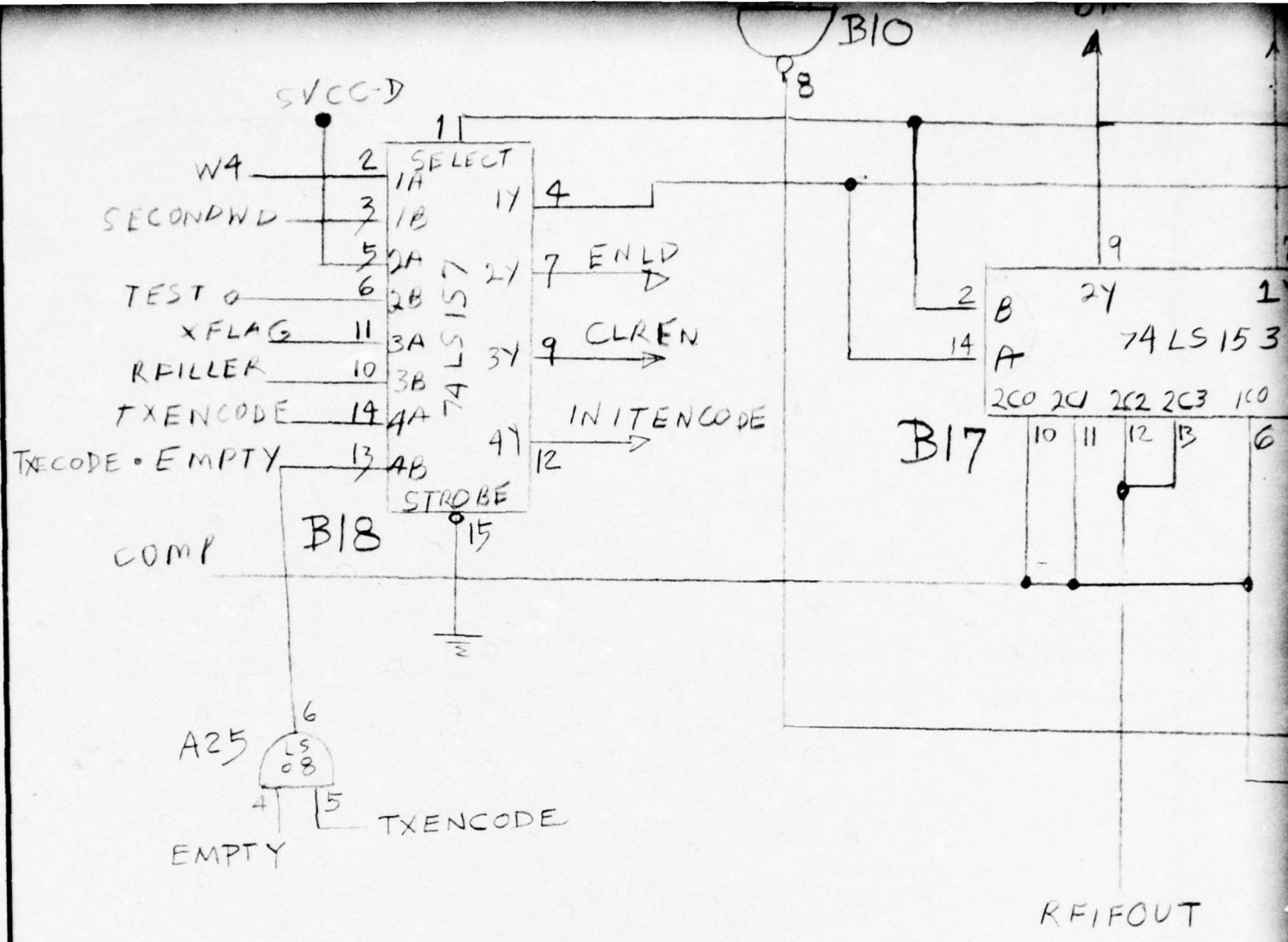
D

C



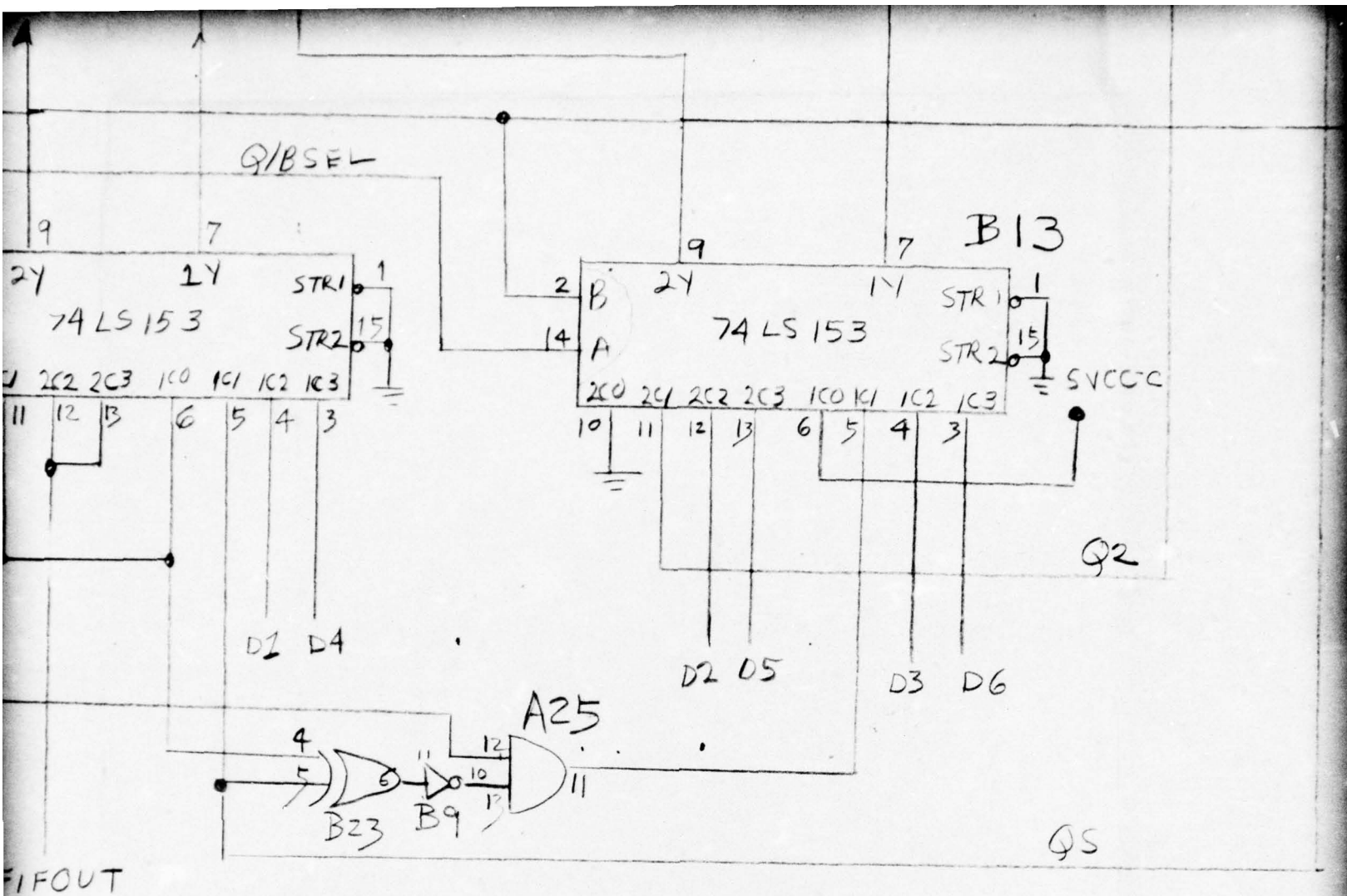
Q5

C
B
A



4

NEXT ASSY
AF



FIFOUT

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.
----------	------------	-----------------------------

NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED
 DIMENSIONS ARE IN INCHES
 TOLERANCES: ANGLES ±
 FRACTIONS ±
 3 PLACE DECIMALS ±
 2 PLACE DECIMALS ±

MATERIAL:

DR	
CHK	
A	
P	
P	
D	
RELEASED	
CONTRACT NO.	

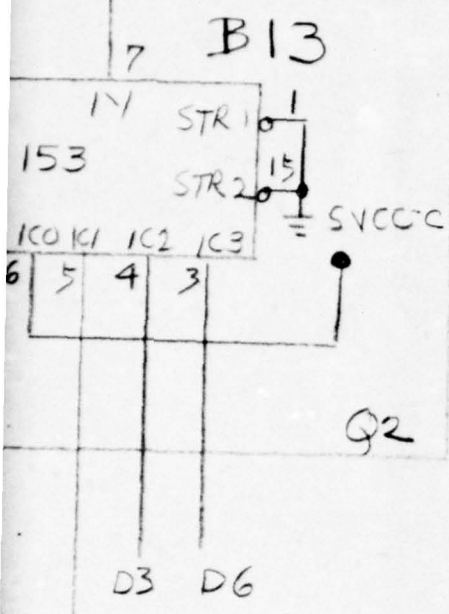
DRAW

SIZE
C

SCALE

← W6

C




Q5

←

B

A

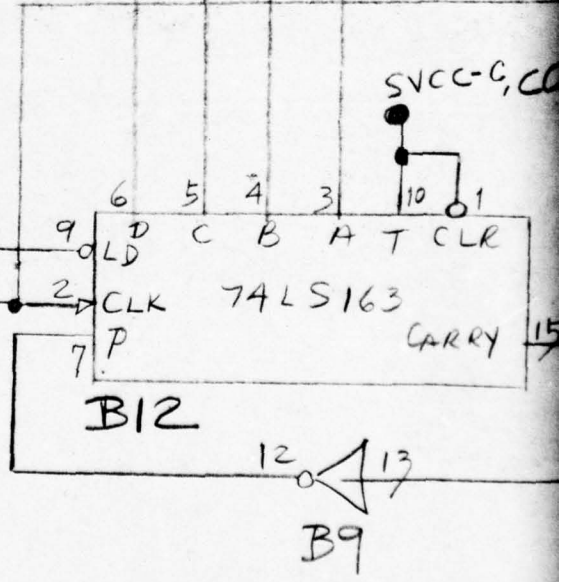
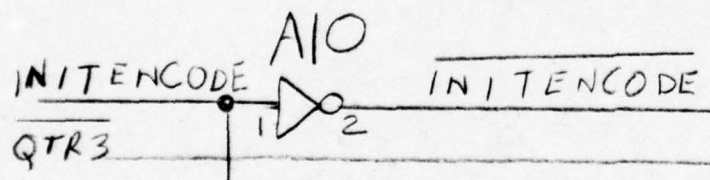
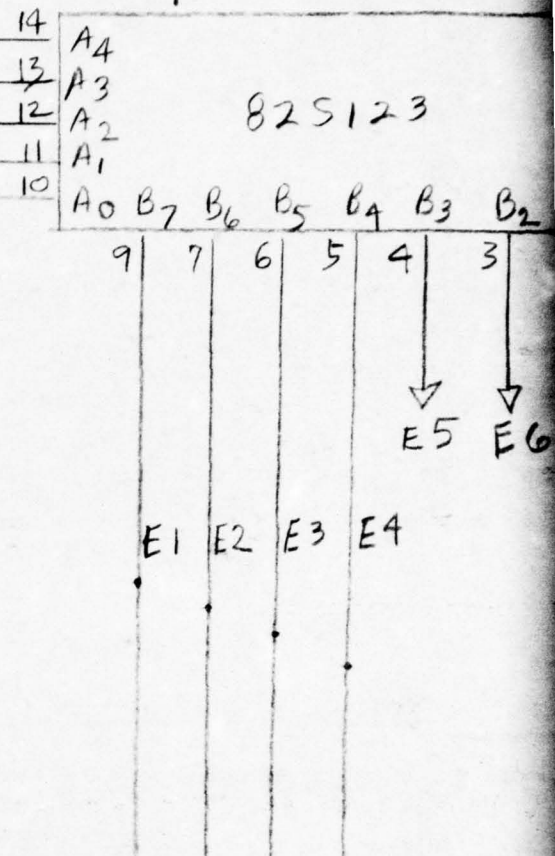
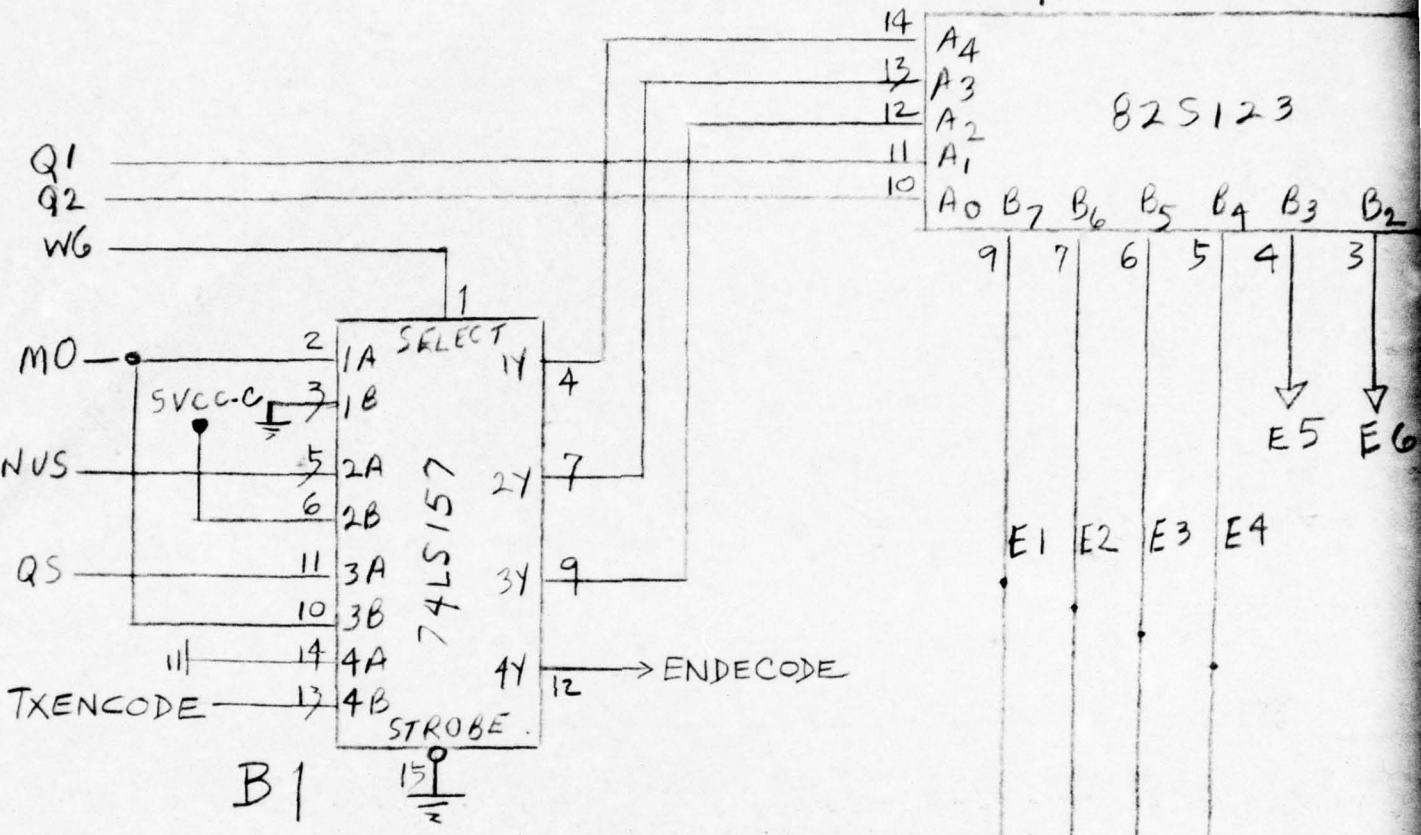
6

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED CHECKS	DR		 DRAWING TITLE Q REGISTER	
	CHK			
	A			
	P			
	D			
RELEASED				
CONTRACT NO.		SIZE	CODE IDENT NO.	DRAWING NO.
		C	25420	FIG. 3-14
		SCALE		SHEET OF

ENCODER ROM

B7

82S123



D

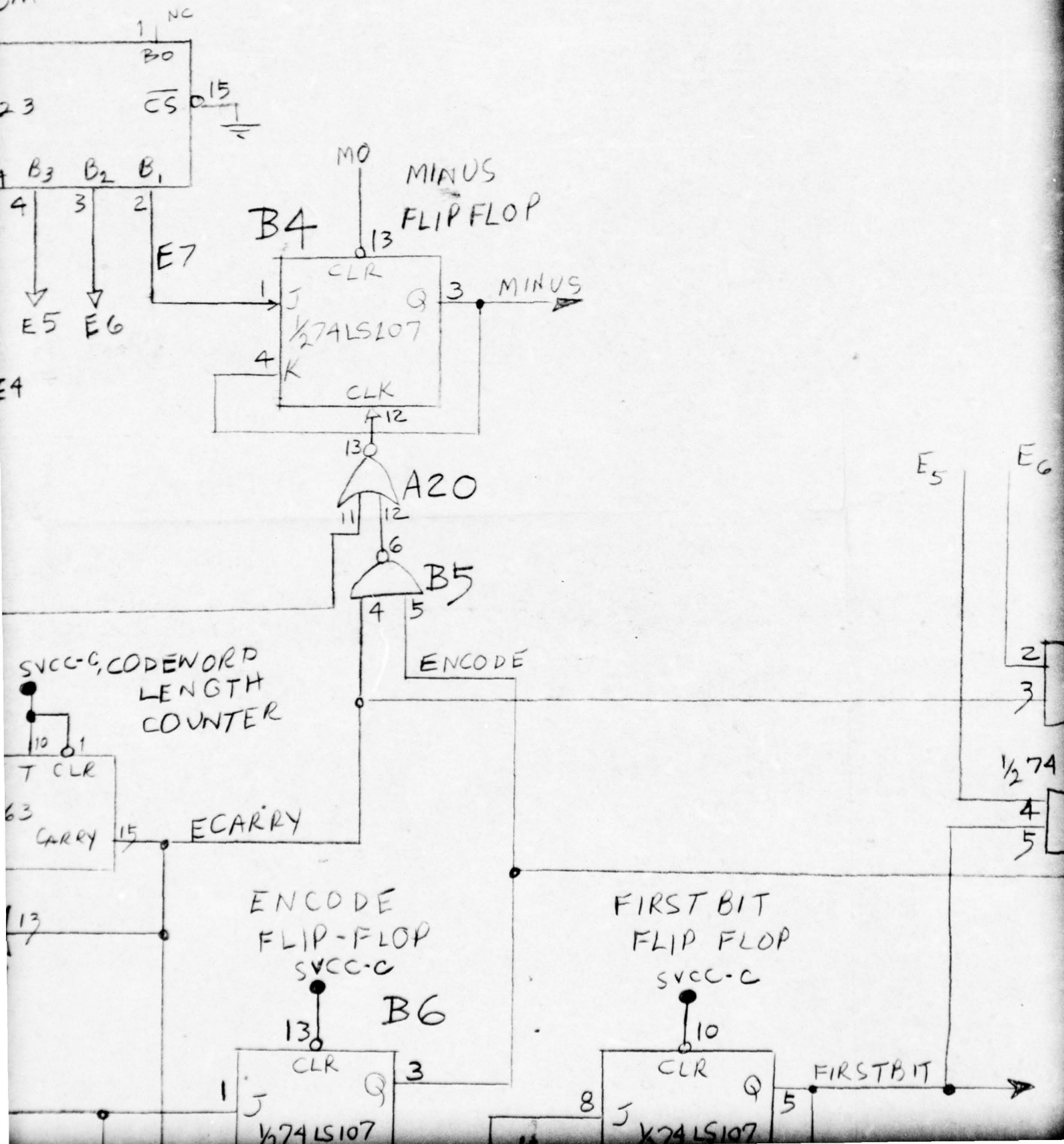
B

3

2

ZONE	LTR
------	-----

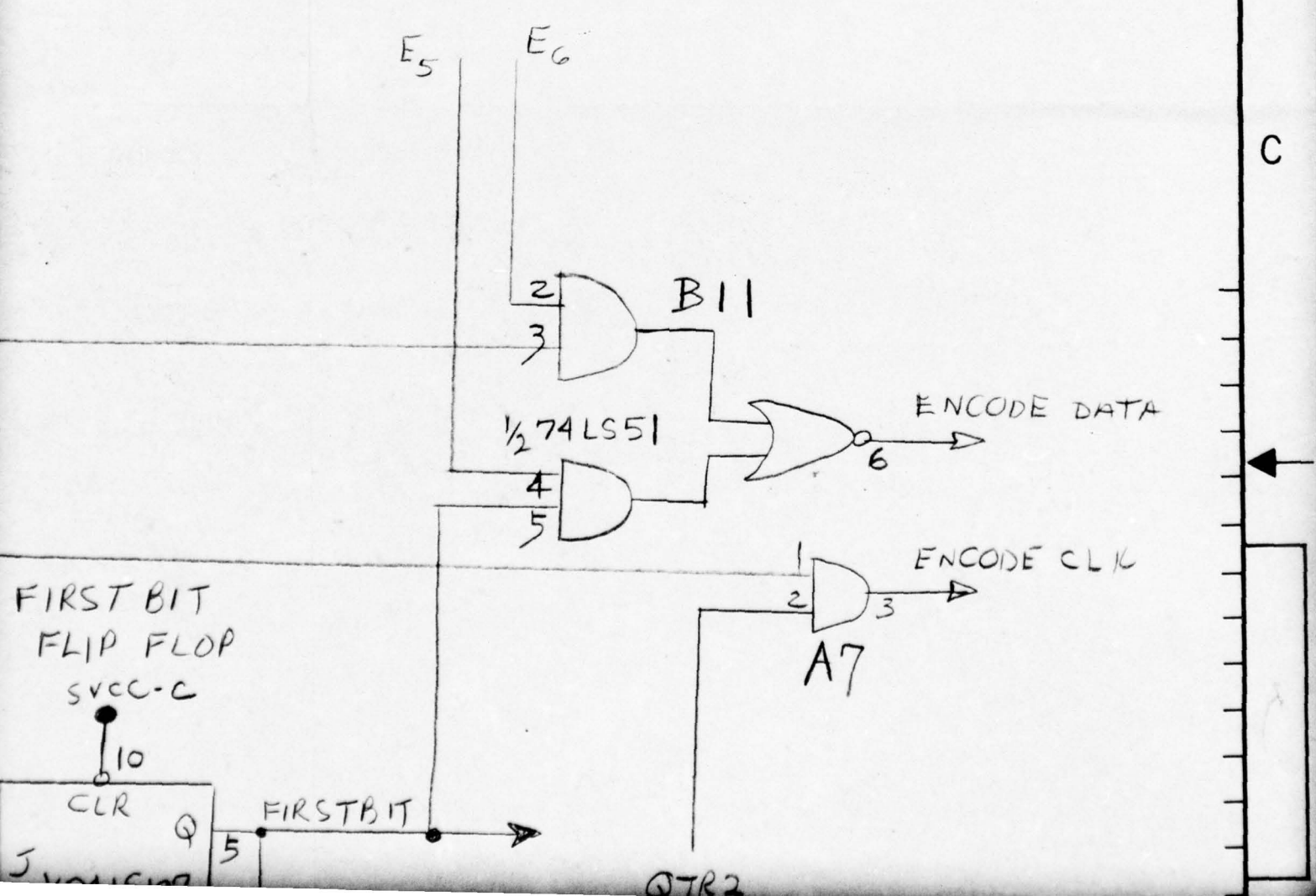
ORDER
DM

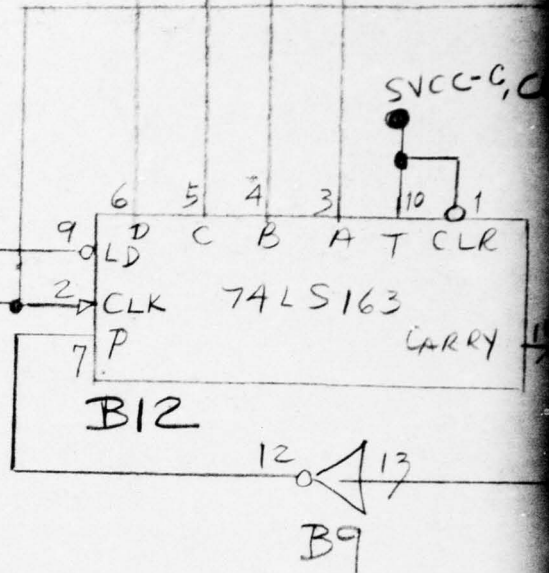
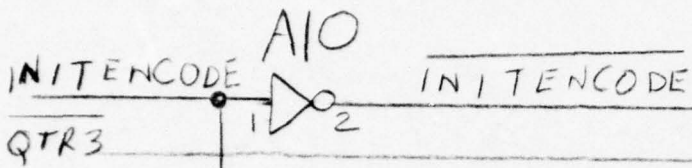
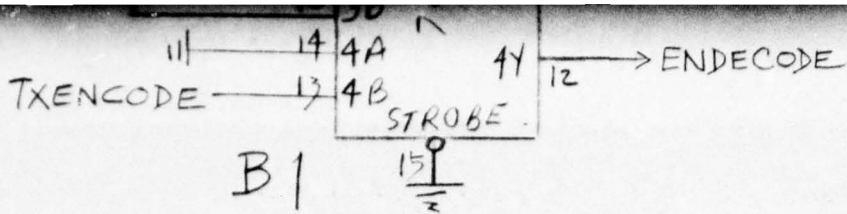


REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
				3

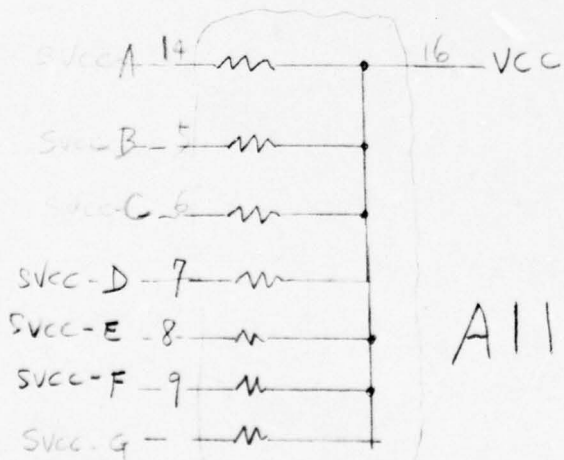
D

C



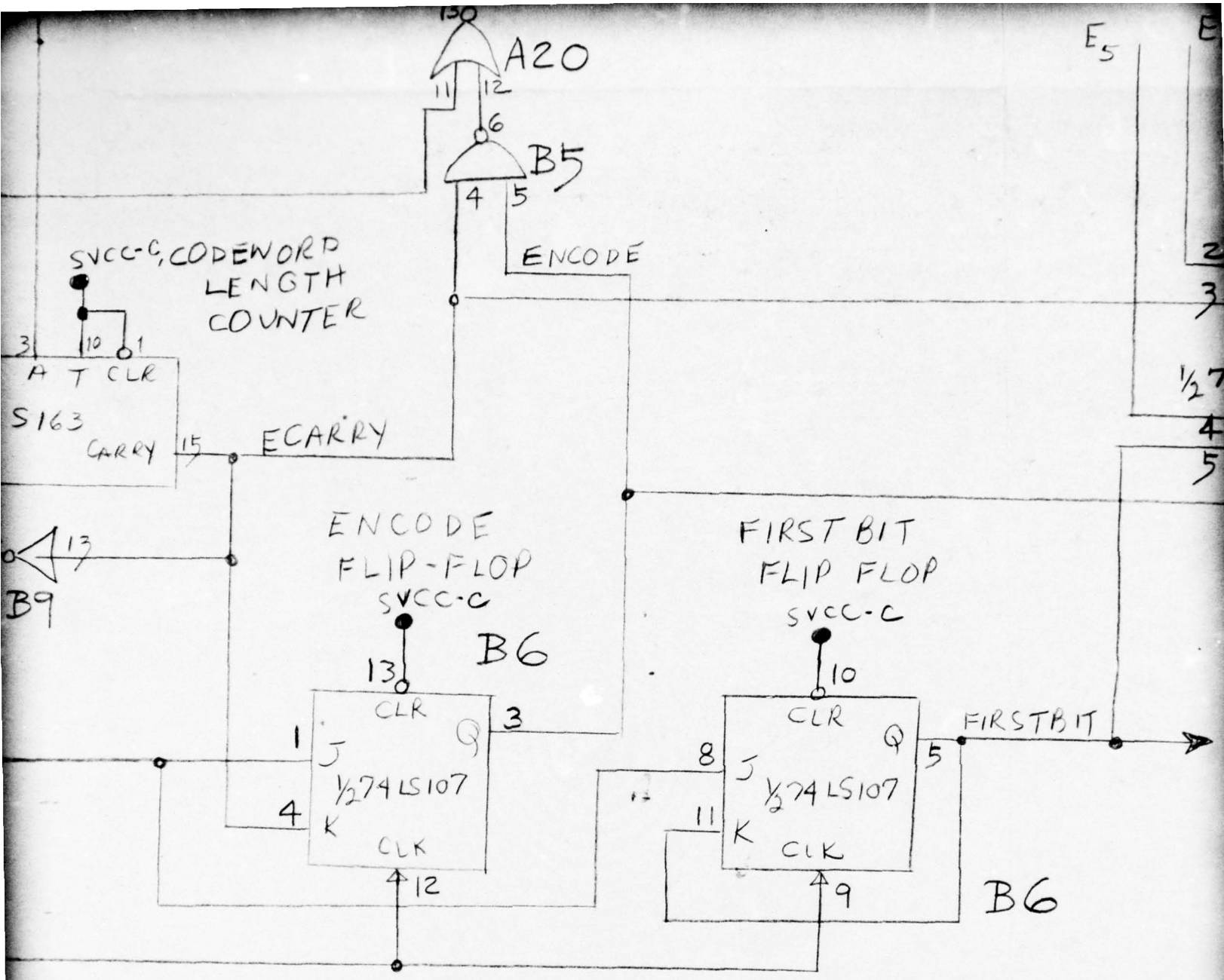


QTR3



NEXT ASSY

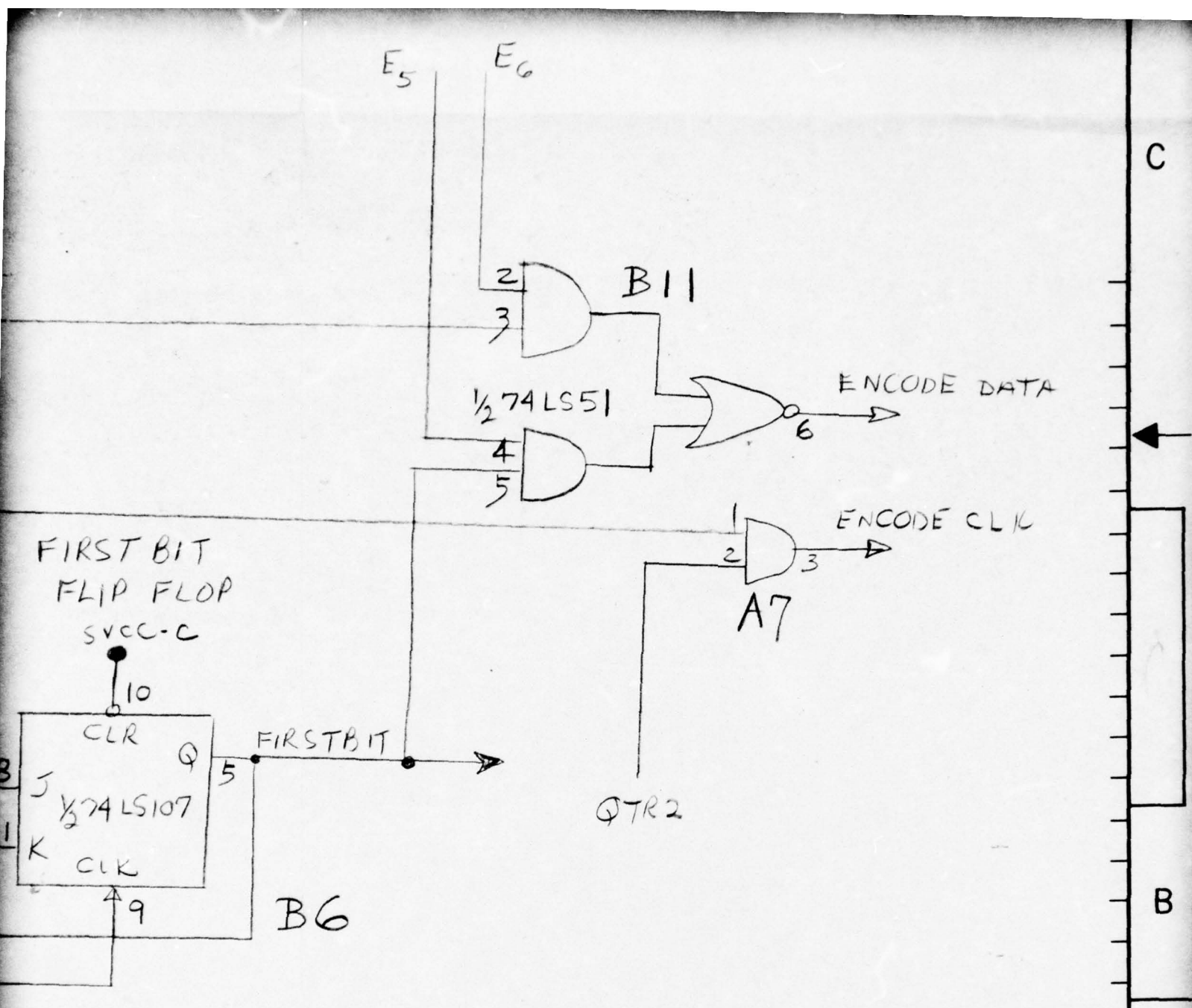
APP




QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.

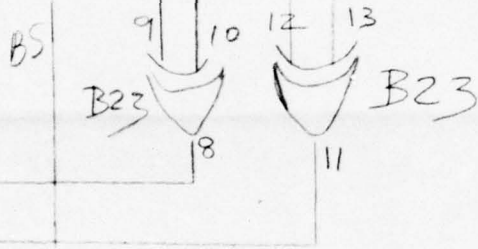
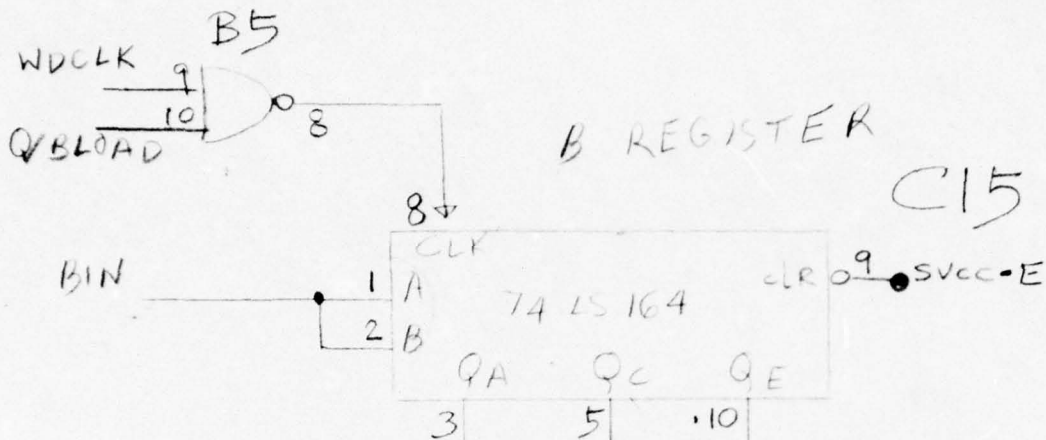
		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES	DR	
		TOLERANCES: ANGLES ±	CHK	
		FRACTIONS ±	A	
		3 PLACE DECIMALS ±	P	
		2 PLACE DECIMALS ±	D	
MATERIAL:		RELEASED		

DRAW

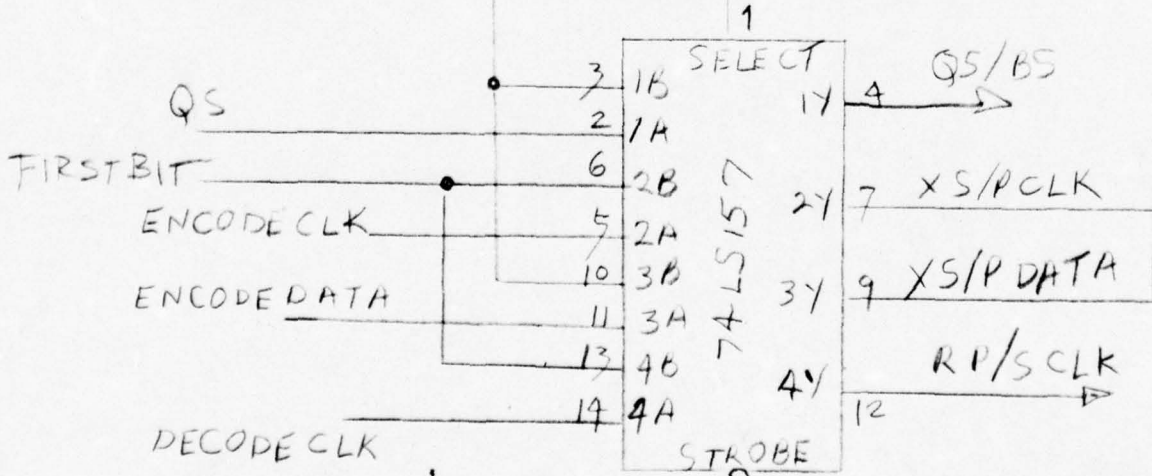


QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED DIMENSIONS	DR		 NEWTON, MASSACHUSETTS 02195	
	CHK			
	A			
	P			
	D			
	RELEASED		DRAWING TITLE	
			ENCODER	
CONTRACT NO.		SIZE	CODE IDENT NO.	DRAWING NO.
		C	25420	FIG. 3-15
SCALE			SHEET OF	

D



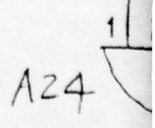
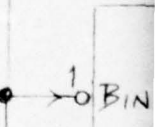
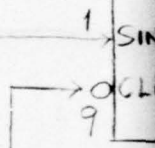
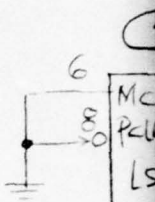
BK1
 BK2
 MI



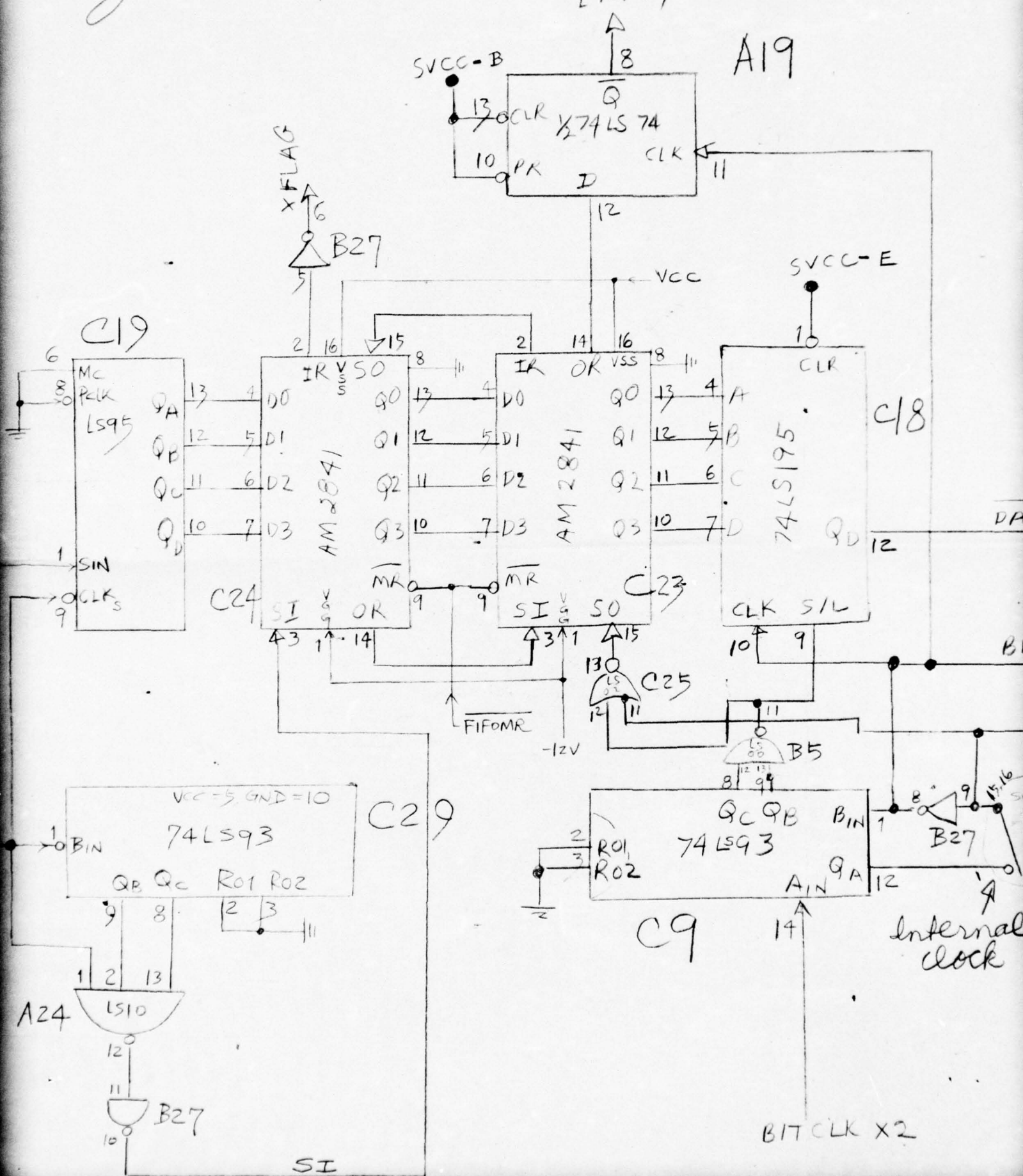
C20

C

B

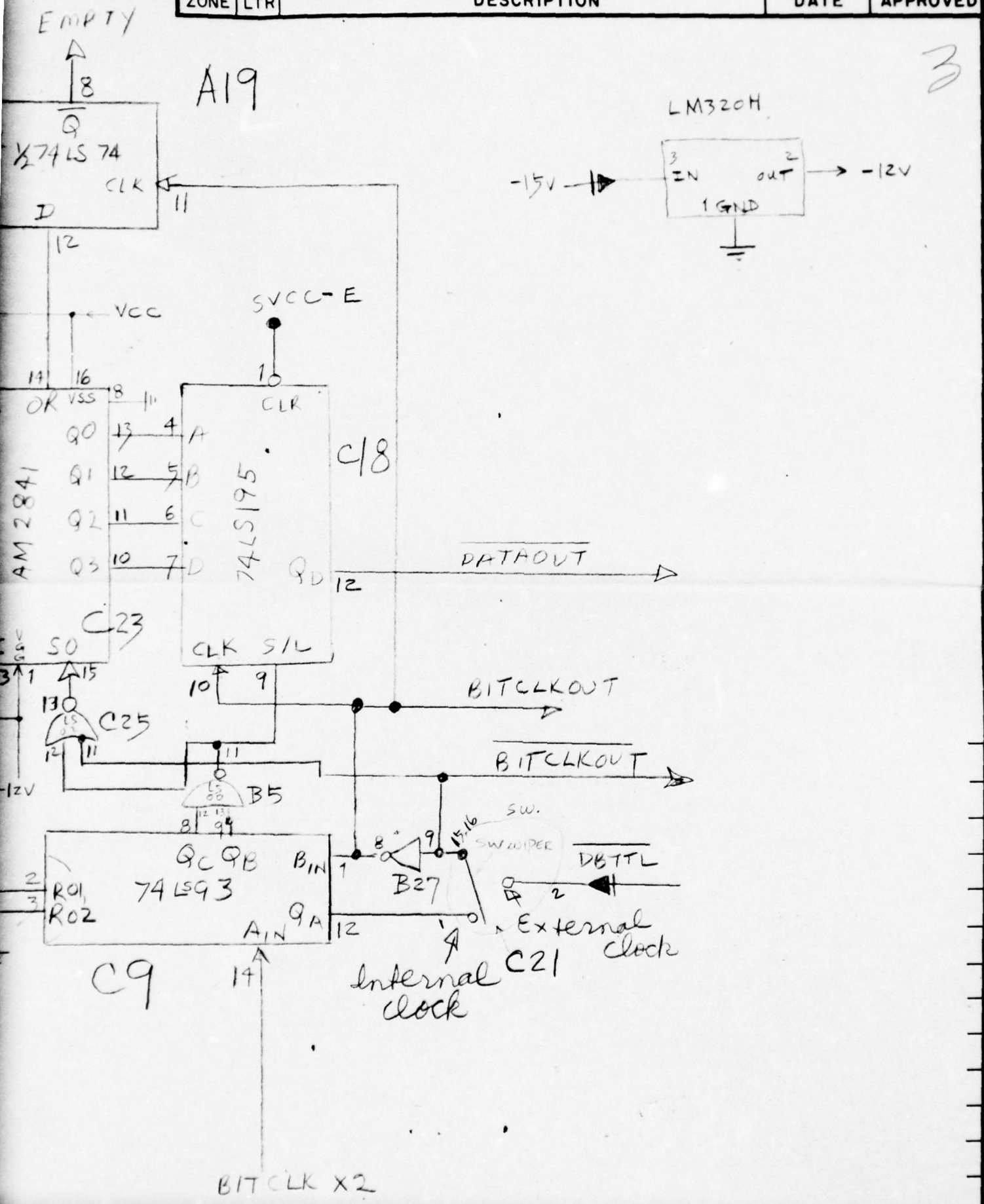


2



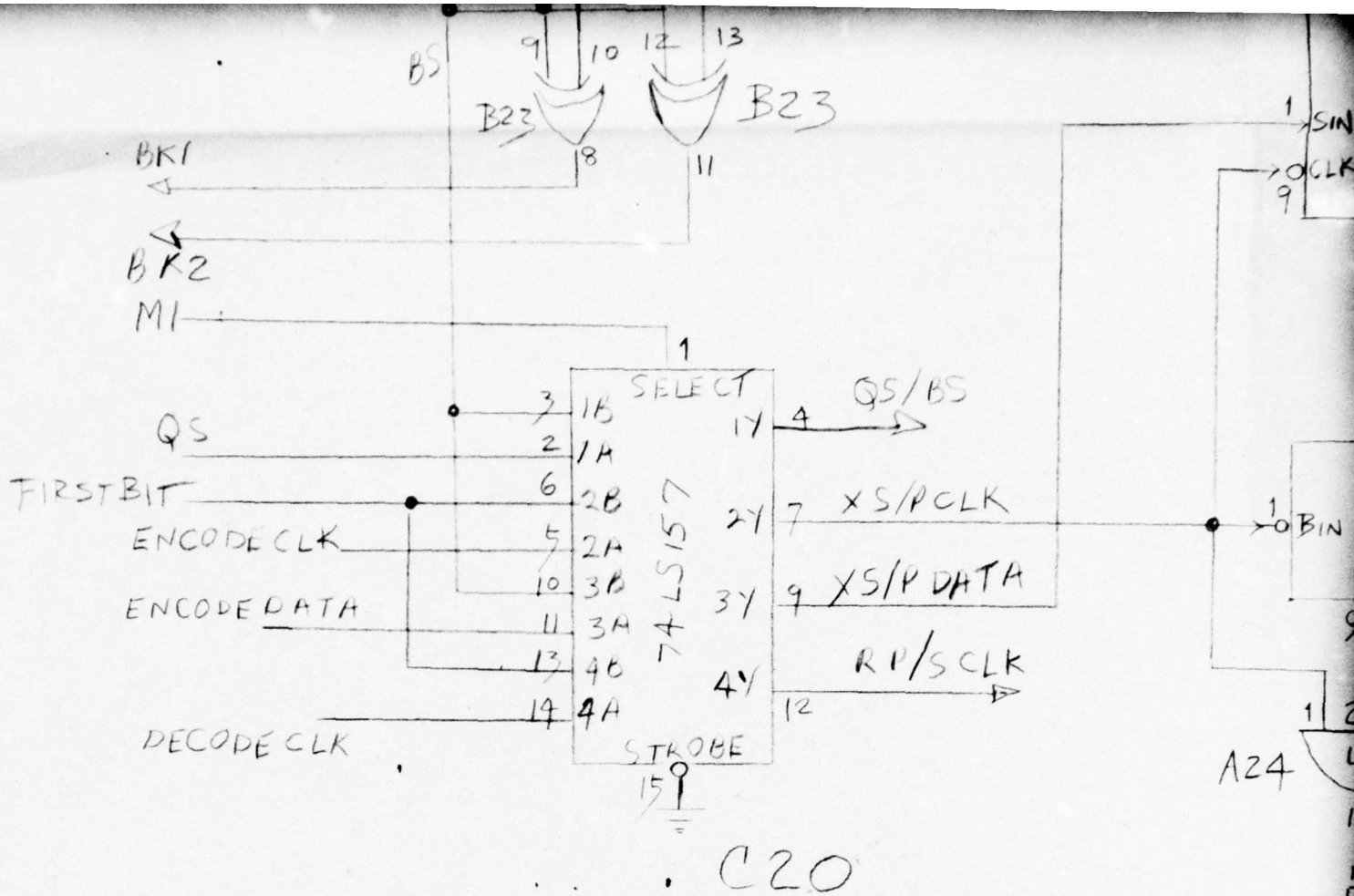
REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

3

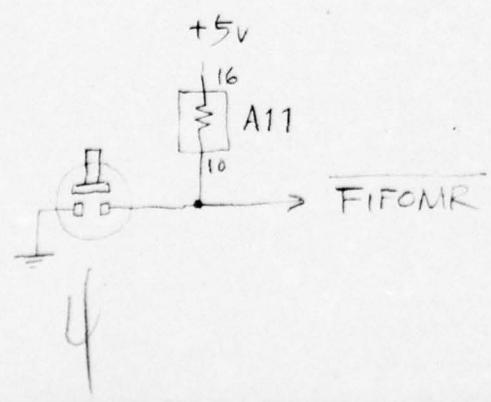


D

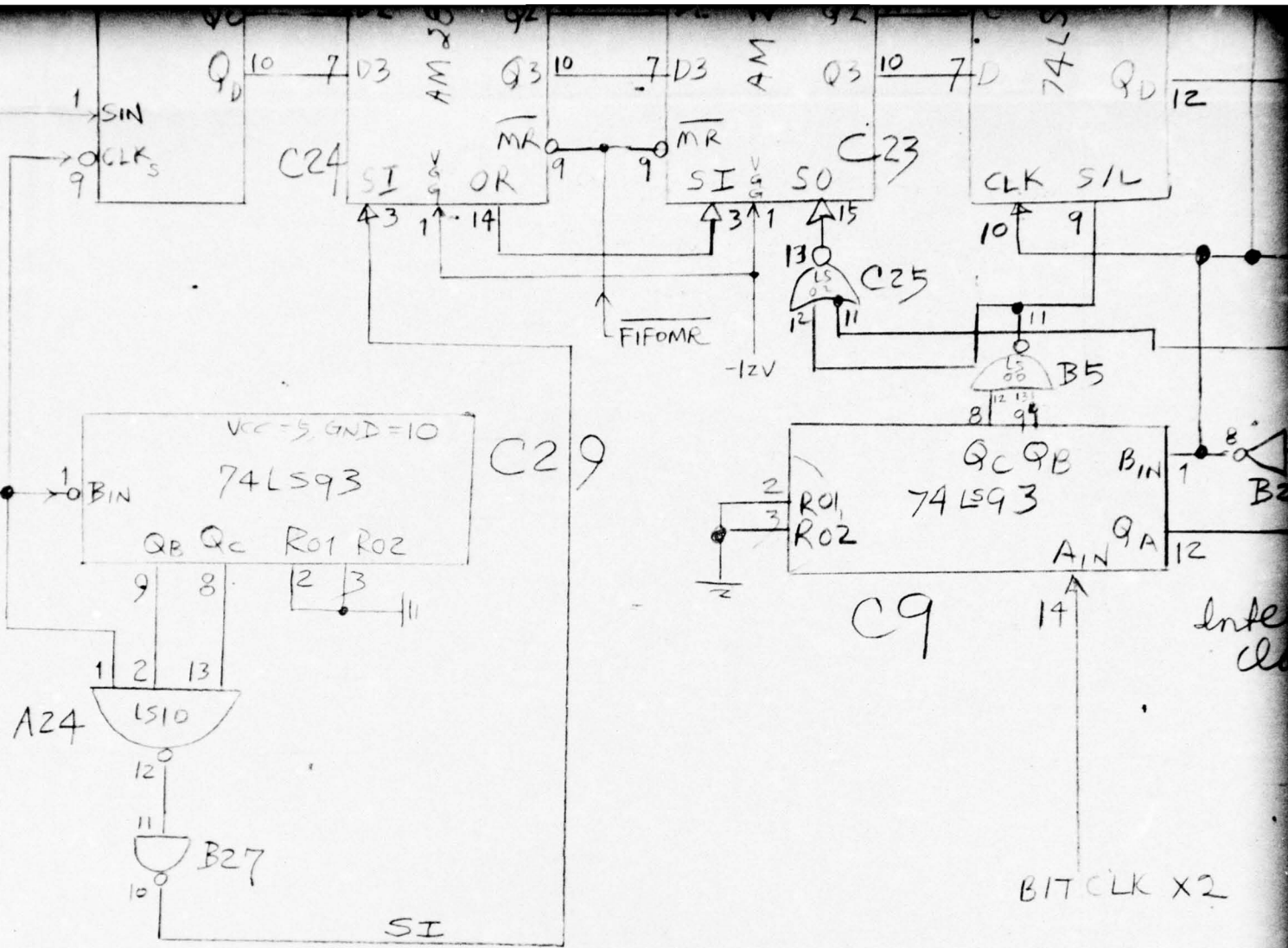
C



C20

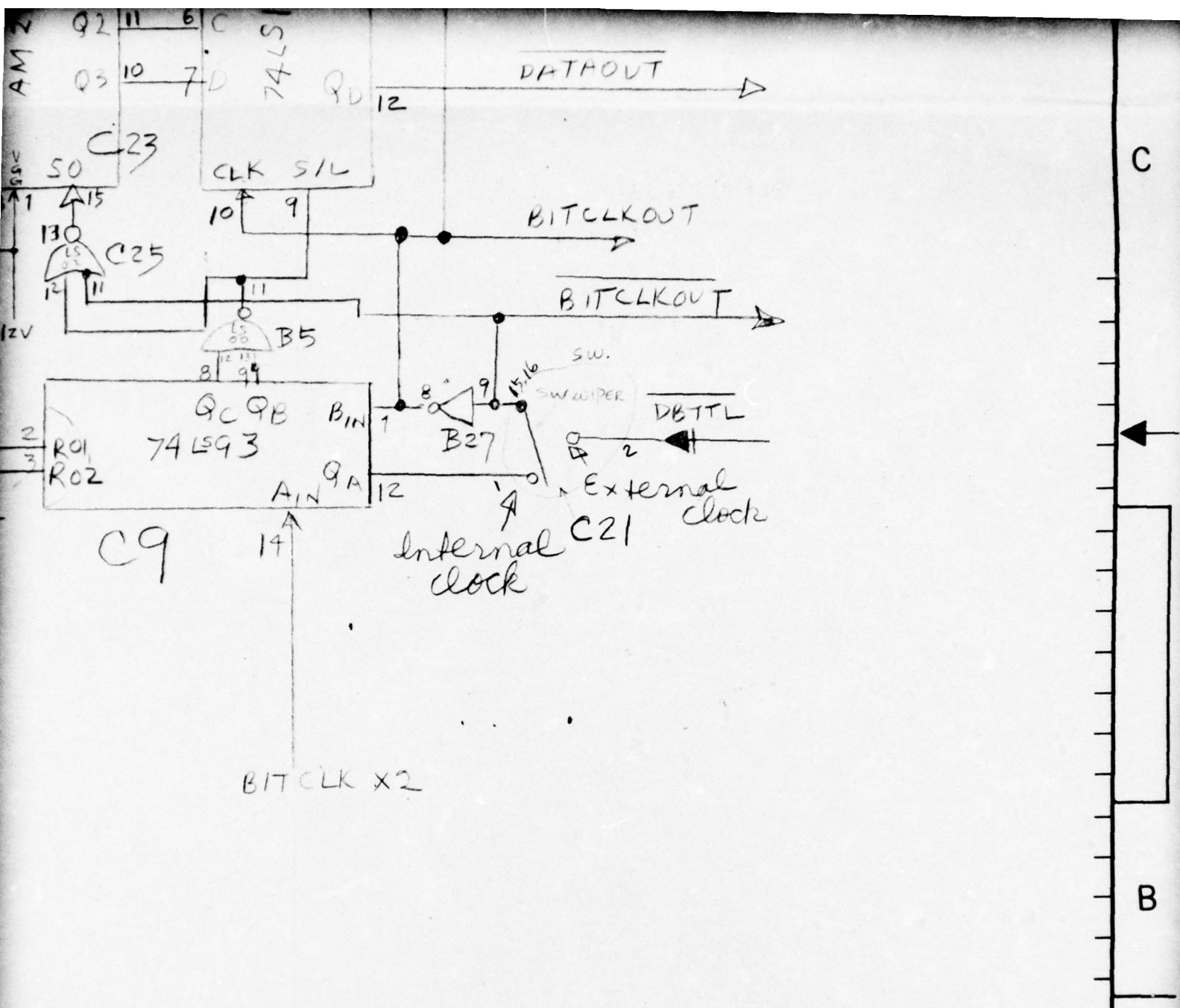



NEXT ASSY
APP



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.
----------	------------	-----------------------------

		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES: ANGLES ± FRACTIONS ± 3 PLACE DECIMALS ± 2 PLACE DECIMALS ±	DR		DR
			CHK		
		MATERIAL:	A		SIZE
			P		
			D		SCALE
			RELEASED		
			CONTRACT NO.		
NEXT ASSY	USED ON				
APPLICATION					



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED CHES	DR		 DRAWING TITLE TRANSMIT FIFO, B REGISTER	NEWTON, MASSACHUSETTS 02195 FIG. 3-16
	CHK			
	A P P D			
	RELEASED			
CONTRACT NO.		SIZE	CODE IDENT NO.	DRAWING NO.
		C	25420	FIG. 3-16
		SCALE	SHEET OF	

C

B

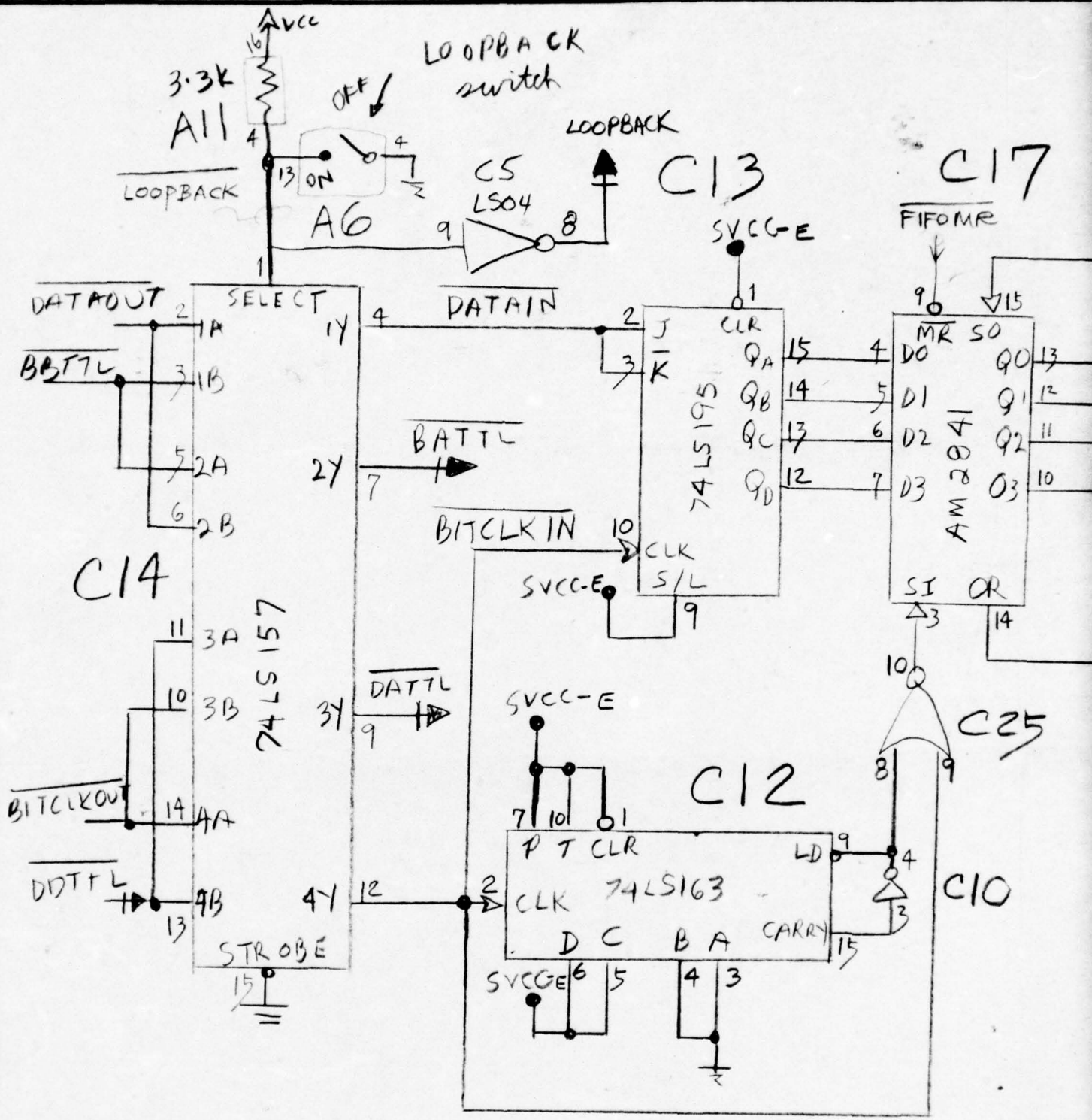
A

6

I
D

C

3



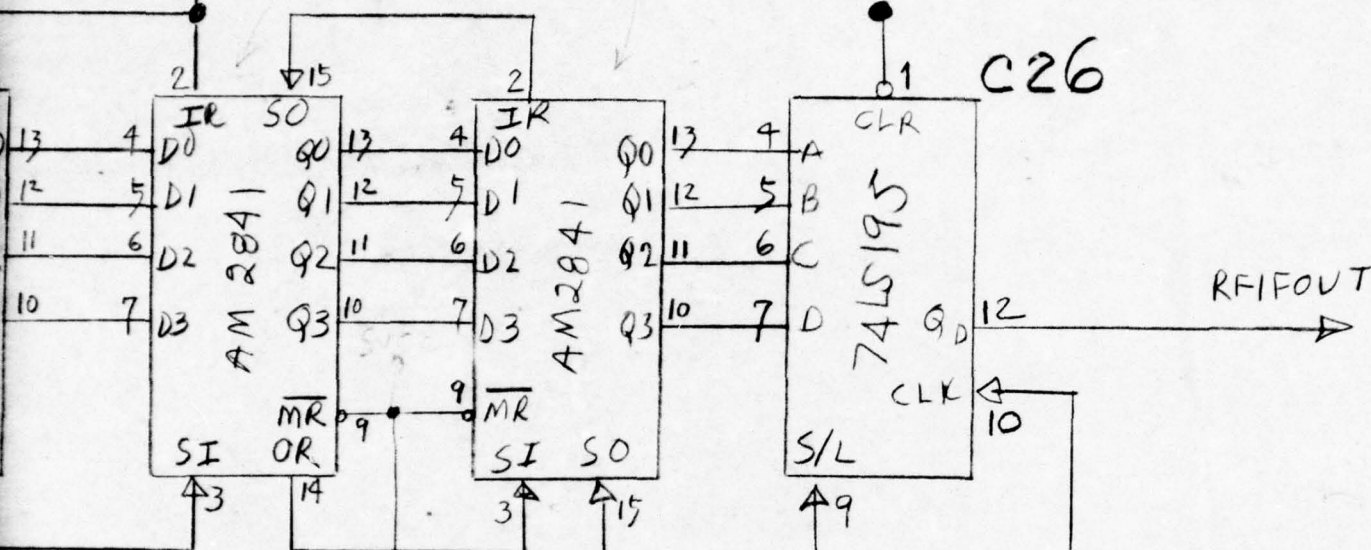
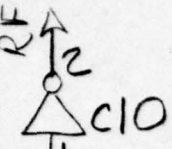
2

7

RFLAG

AM2841
GND=8
VCC=16
-12V=1

SVCC-F



C22

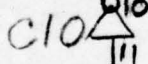
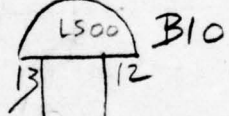
C27

C26

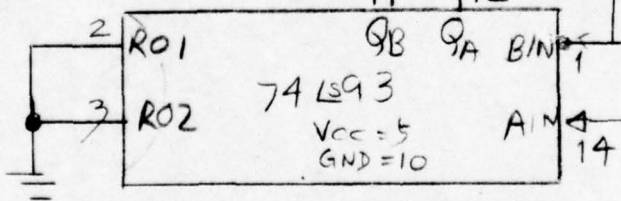
RFI FOUT

FIFOMR

RP/sdW/4



RP/sclW/2



RP/SCLK

C28

REVISIONS

ZONE LTR

DESCRIPTION

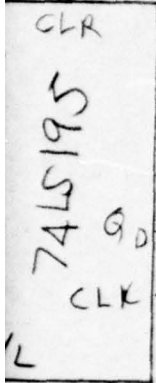
DATE

APPROVED

3

SVCC-F

C26



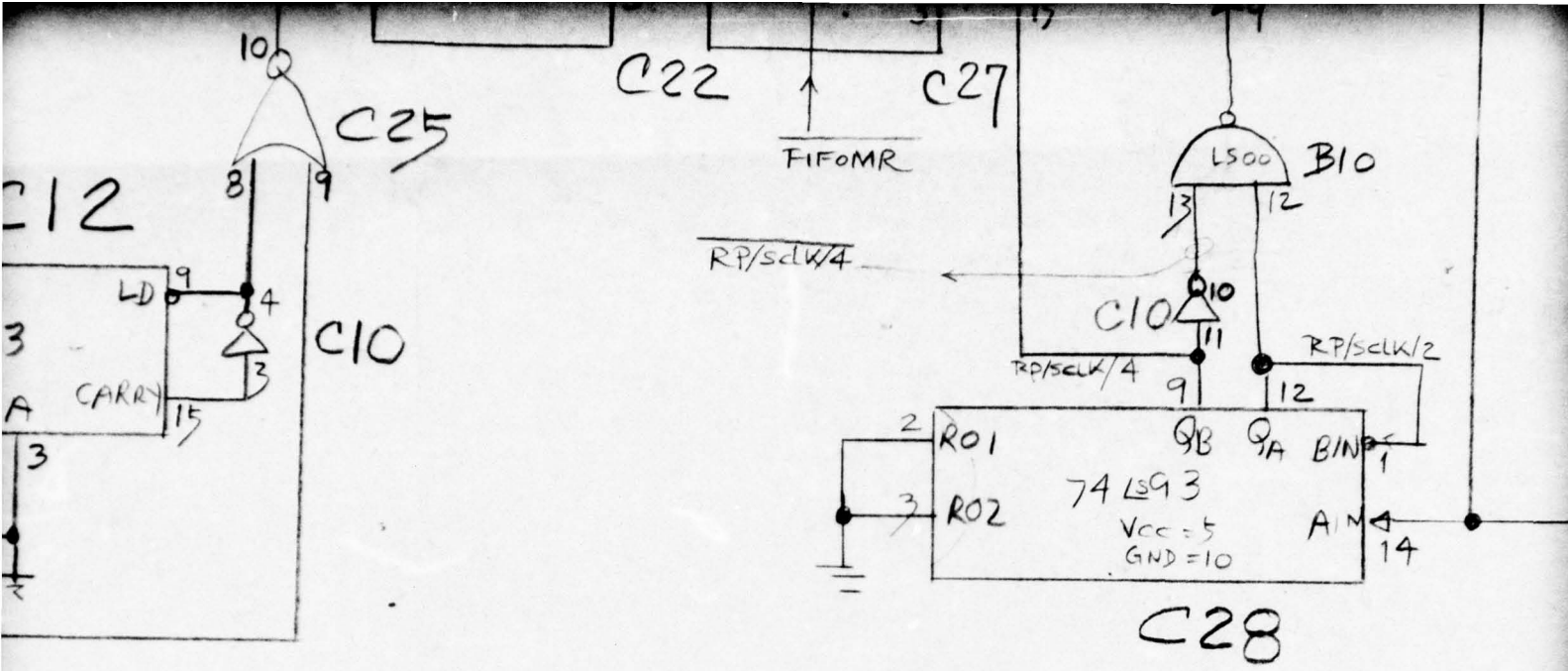
RFIFOUT

B10

RP/SCLK/2

RP/SCLK

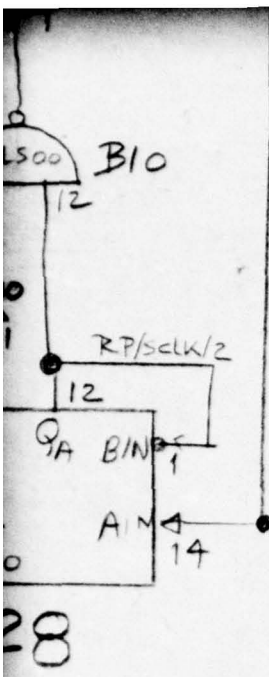
8



QTY REQD	CODE IDENT	
----------	------------	--

		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES: ANGLES ± FRACTIONS ± 3 PLACE DECIMALS ± 2 PLACE DECIMALS ± MATERIAL:	DR
			CHK
			A
			P
			P
			D
			RELEASED
			CONTRACT NO.
NEXT ASSY	USED ON		
APPLICATION			

4




28

C

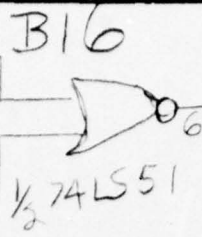
B

A

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
PARTS LIST				
SPECIFIED CHECKS	DR		 DRAWING TITLE TRANSMIT FIFO	
	CHK			
	A			
	P			
	D			
	RELEASED			
CONTRACT NO.		SIZE	CODE IDENT NO.	DRAWING NO.
		C	25420	FIG. 3-17
SCALE			SHEET OF	

5

ENDECODE 2
 SECONDDWD 3
 RFILLER 4
 RFLAG 5

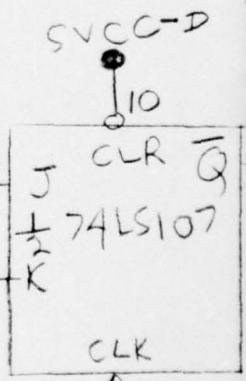
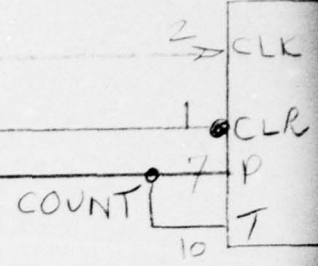
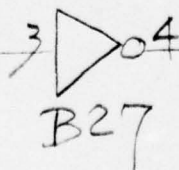
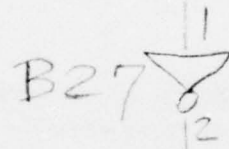


QTR3

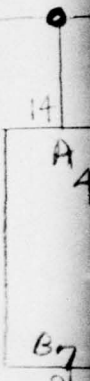
MO

RFIFOUT

QTR 1



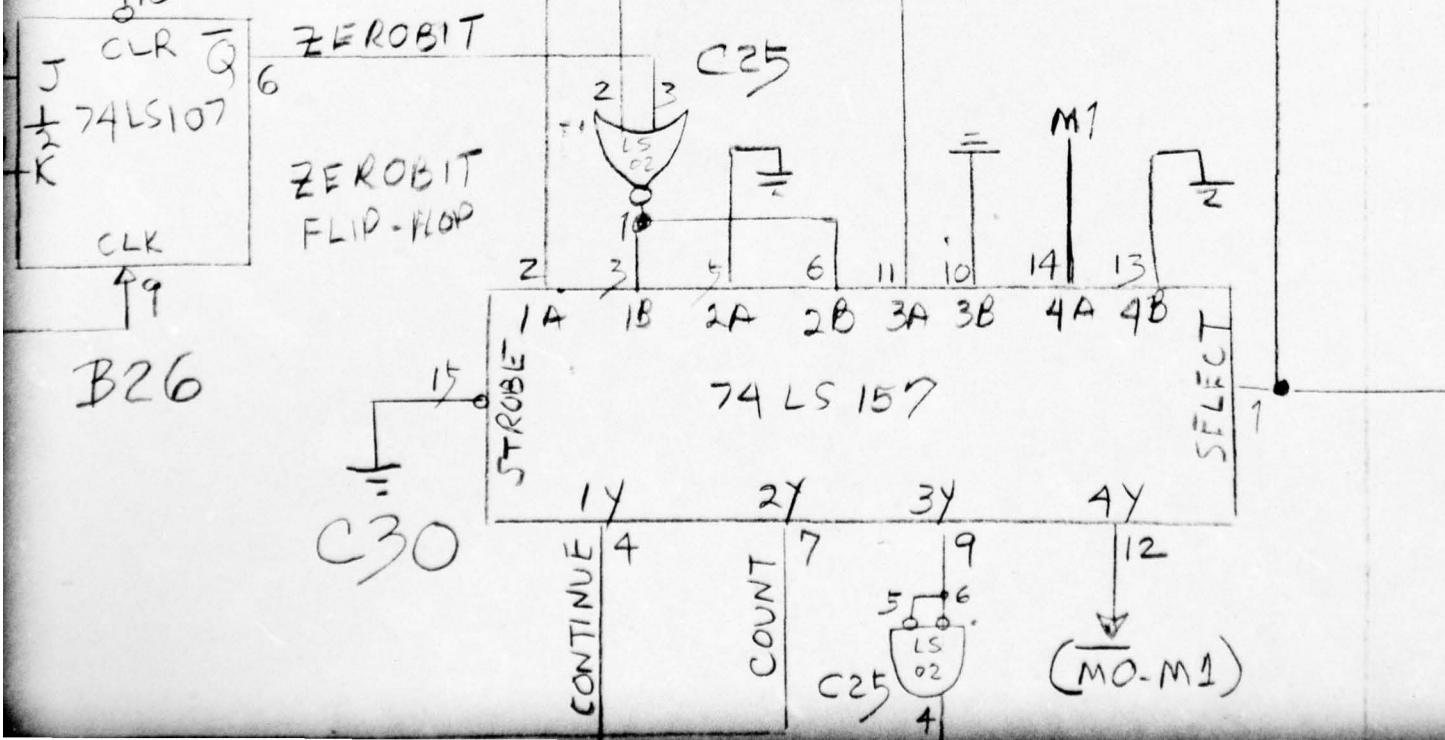
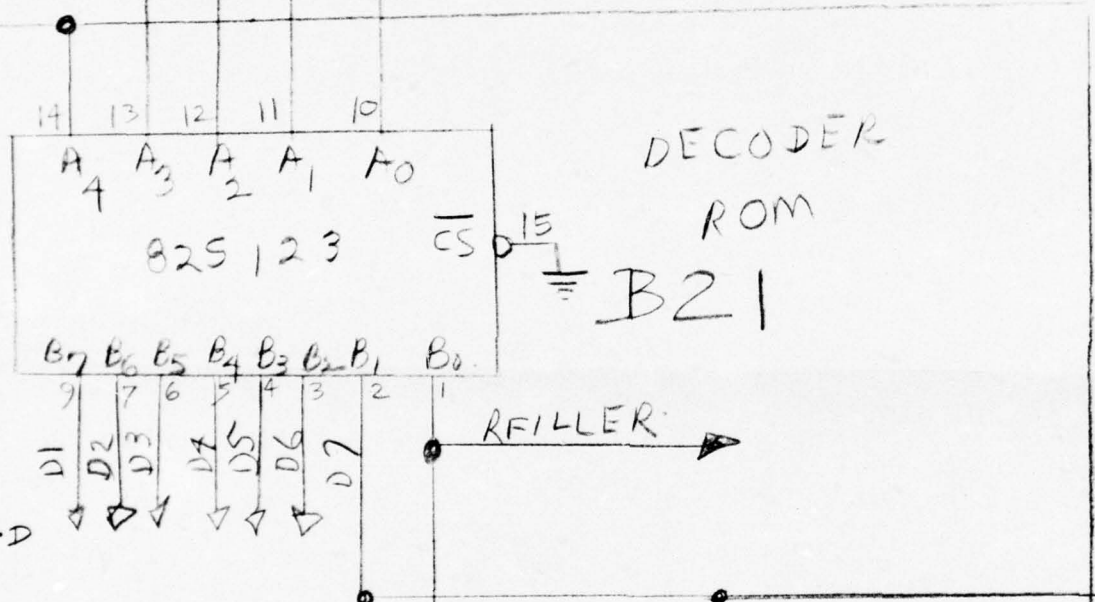
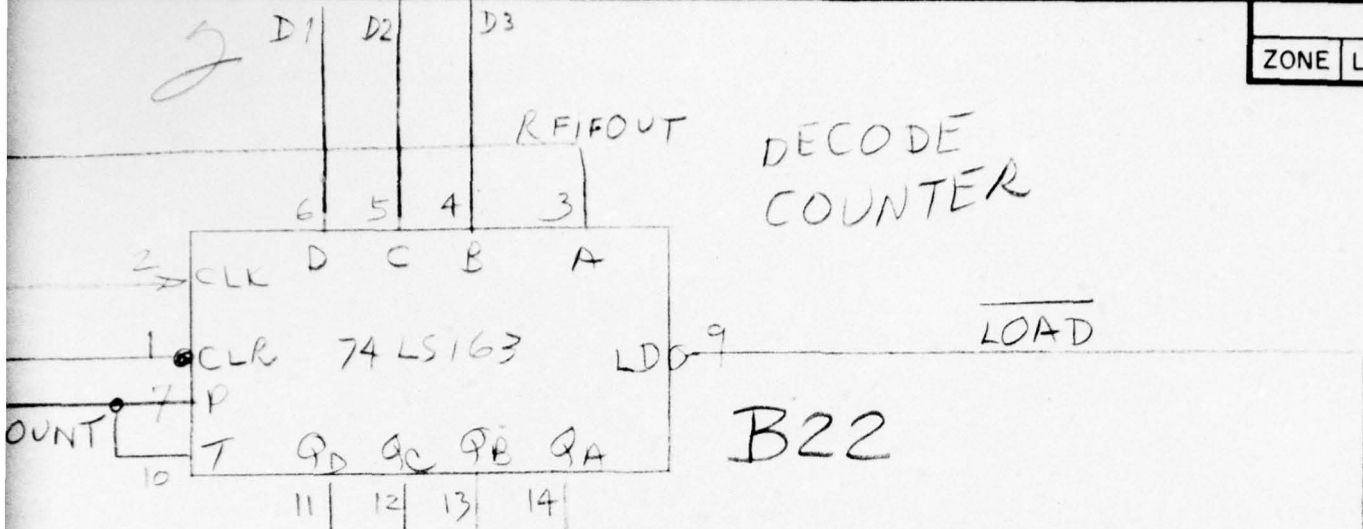
B26



D

C

E



REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

3

D

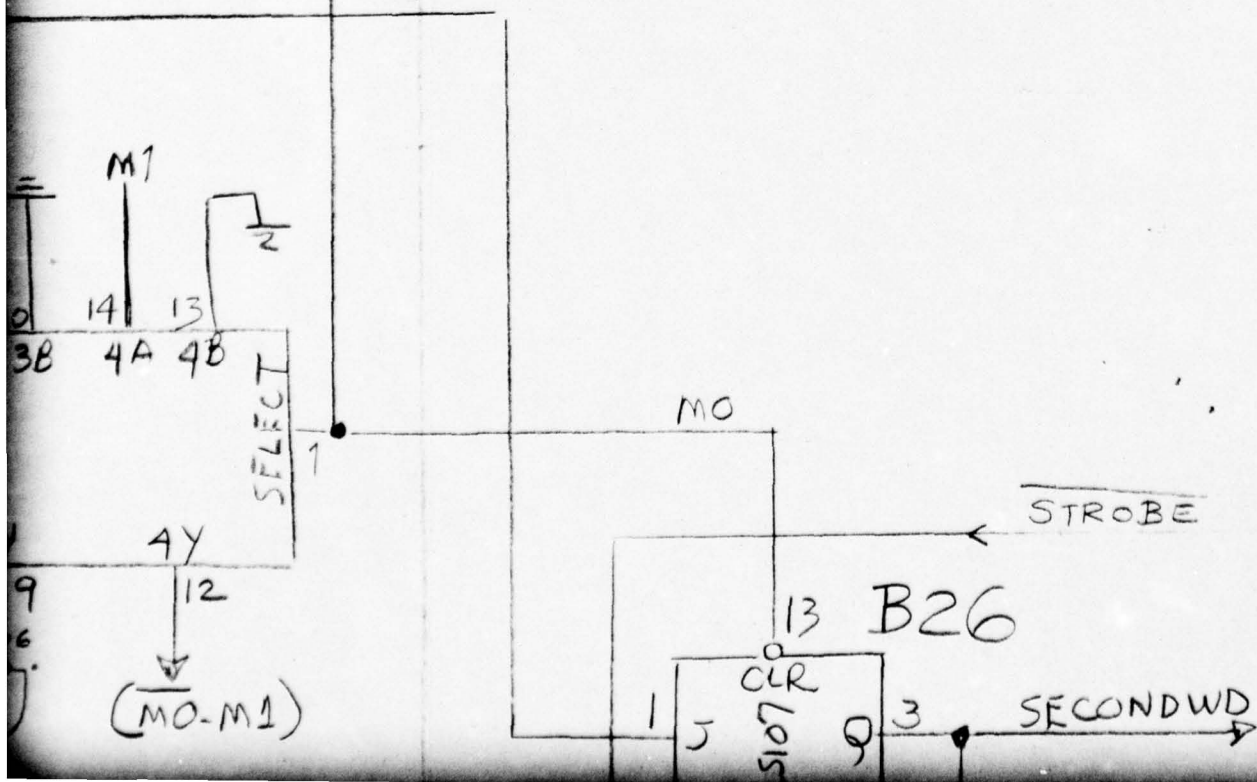
C

ER

LOAD

DEK

M



C

E

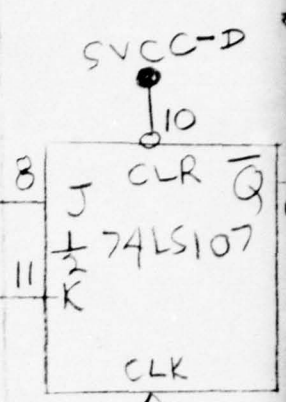
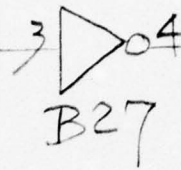
A

KFIFOUT

QTR 1

B27

INITDECODE

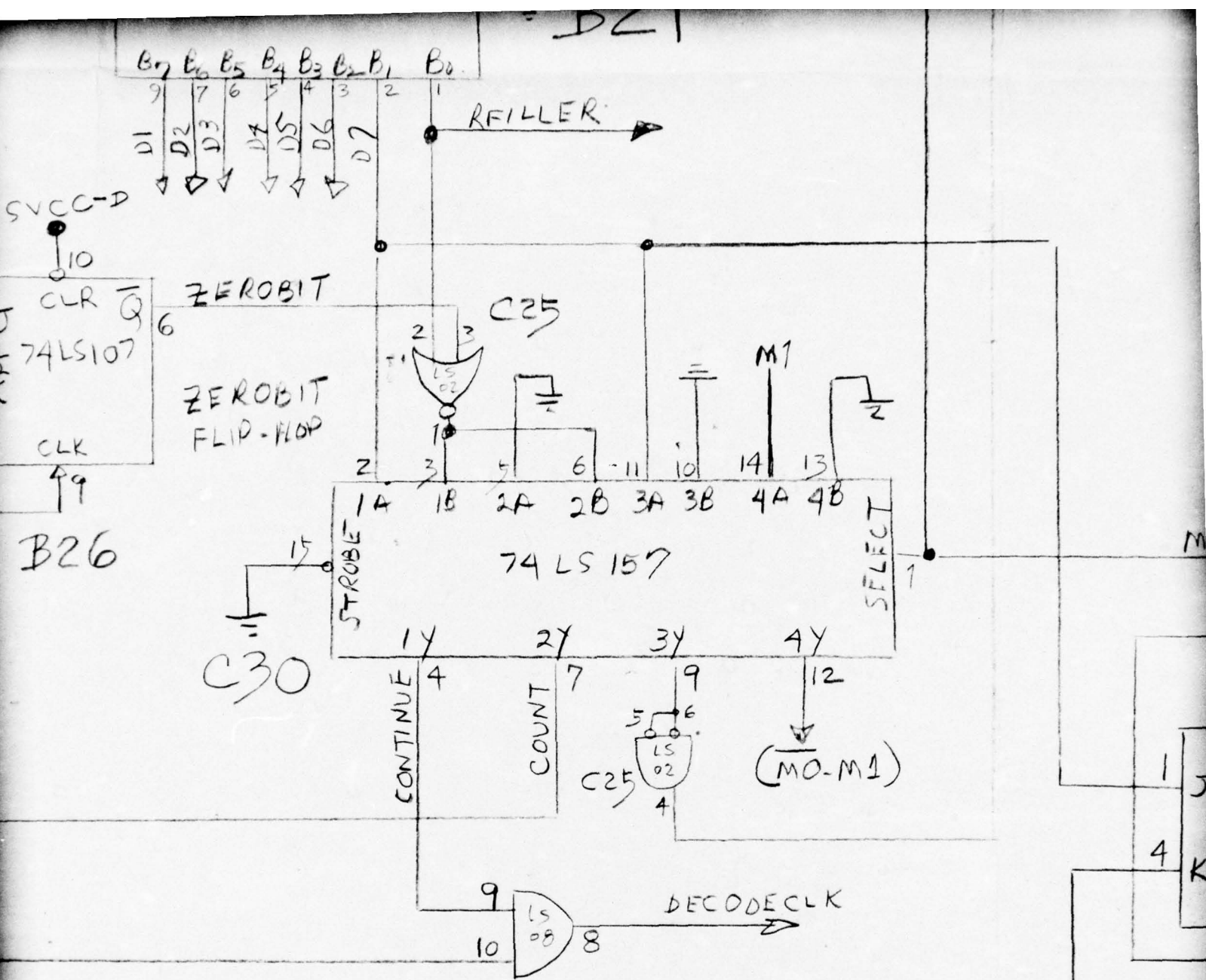


B26

4

NEXT ASSY

AP



QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.

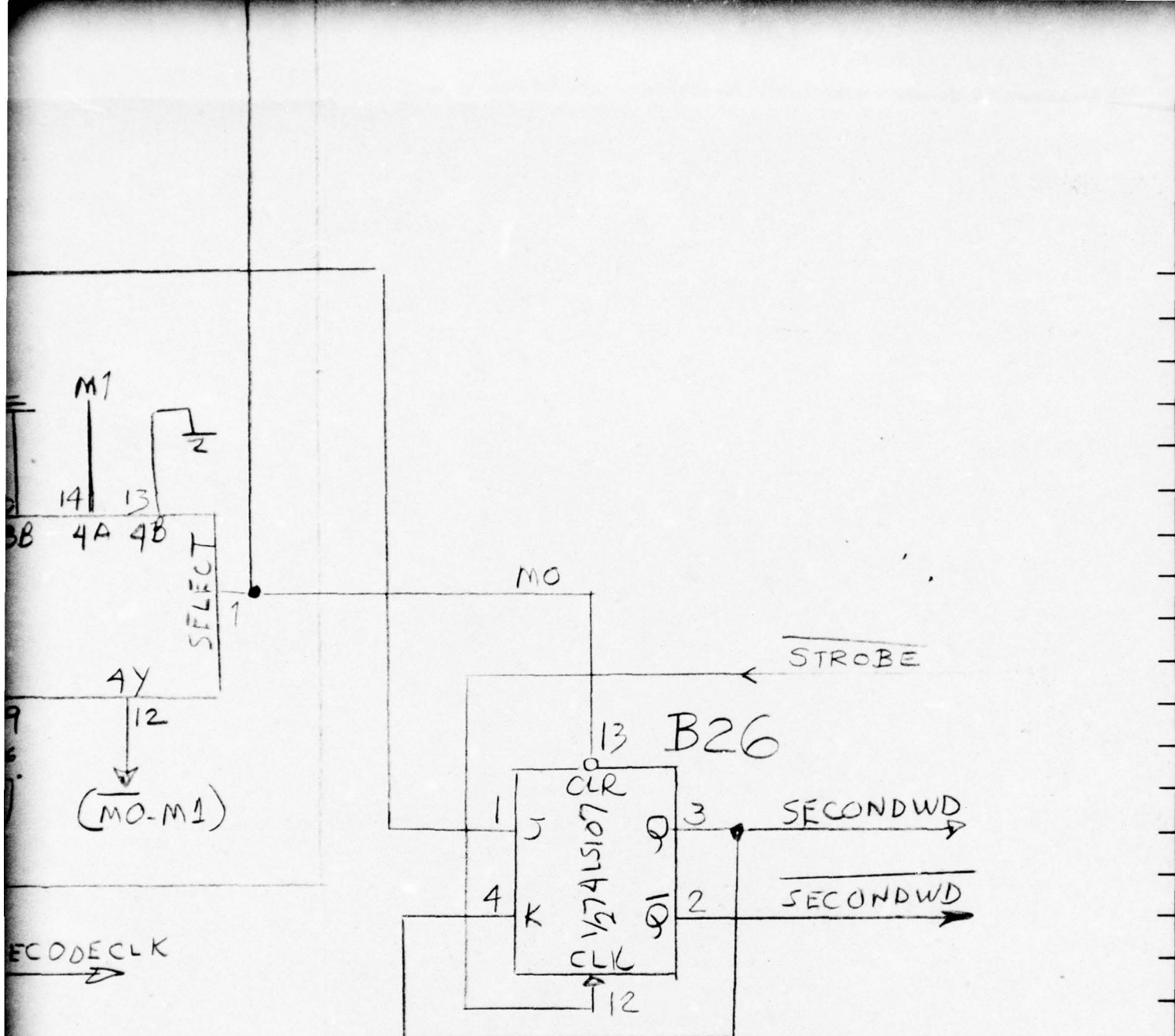
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED
 DIMENSIONS ARE IN INCHES
 TOLERANCES: ANGLES ±
 FRACTIONS ±
 3 PLACE DECIMALS ±
 2 PLACE DECIMALS ±

MATERIAL:

DR	
CHK	
A	
P	
D	
RELEASED	
CONTRACT NO.	

DRAWN
SIZE C
SCALE



C
B
A

QTY REQD	CODE IDENT	PART NO. OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	FIND NO.
----------	------------	-----------------------------	-----------------------------	----------

PARTS LIST

SPECIFIED CHECKS	DR	
	CHK	
	A	
	P	
	D	
RELEASED		

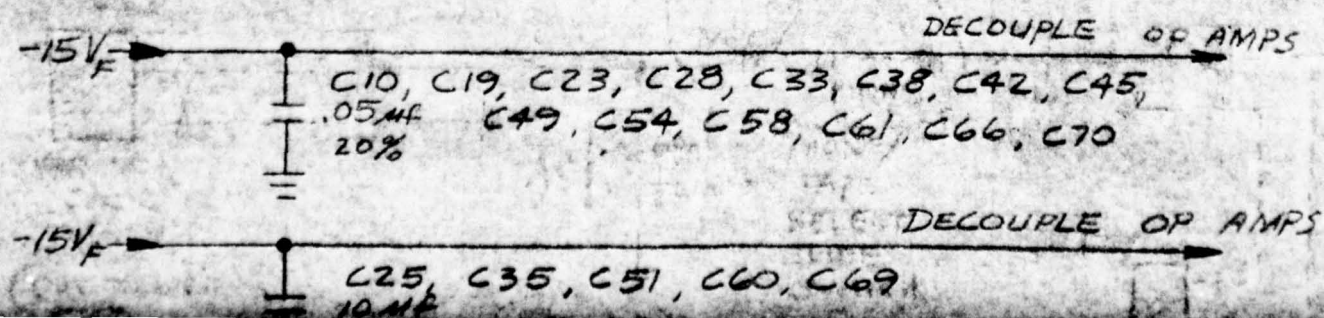
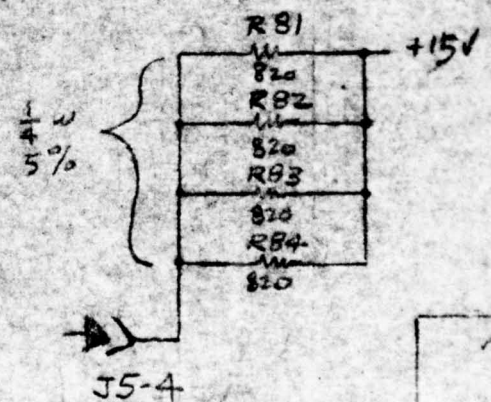
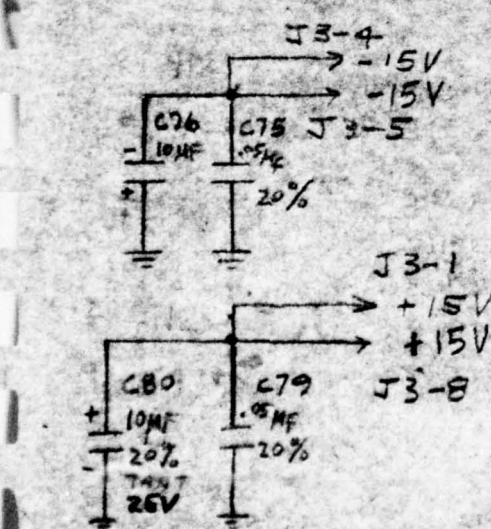
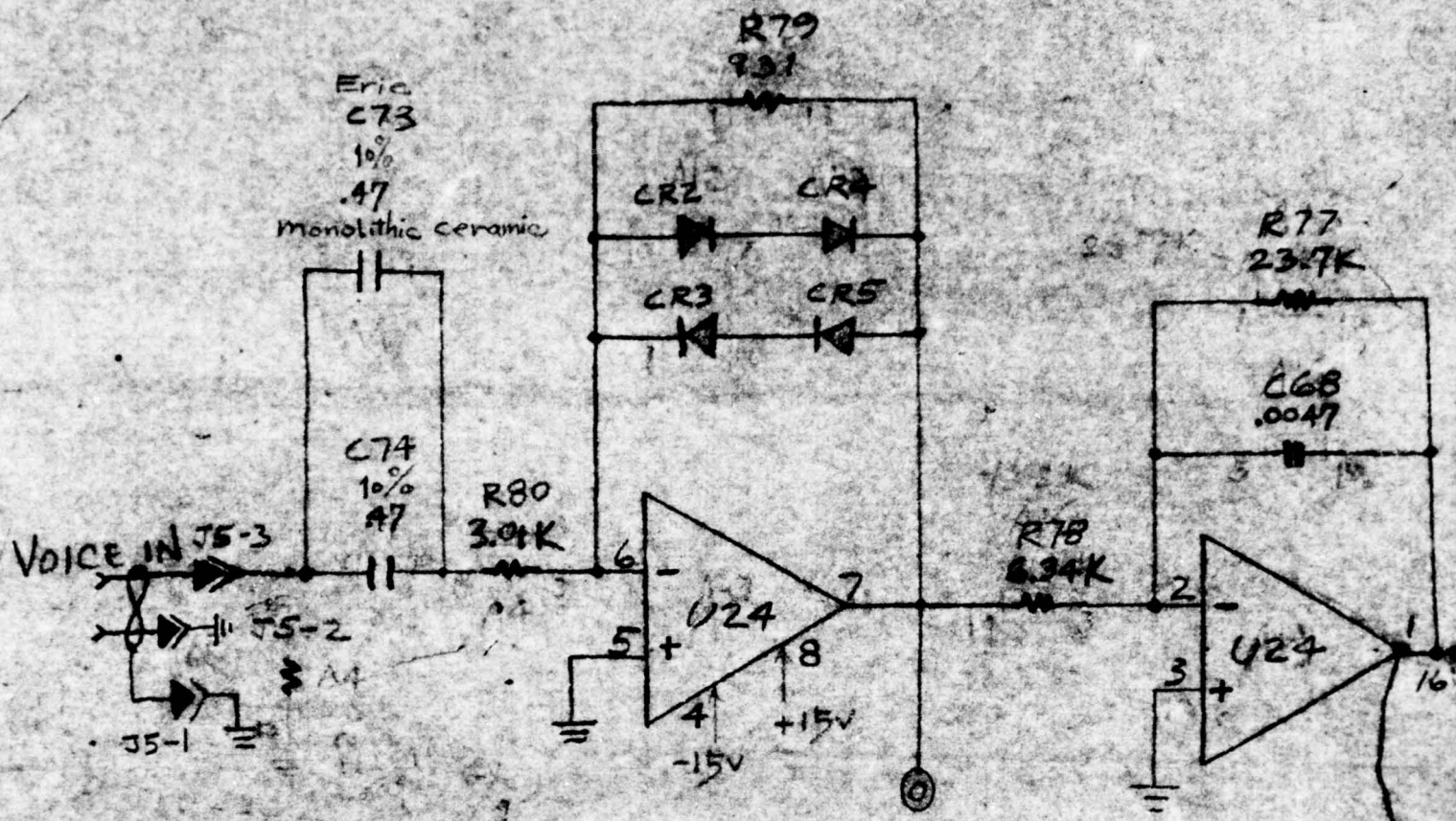
codex corporation
 NEWTON, MASSACHUSETTS 02195

DRAWING TITLE
 DECODER

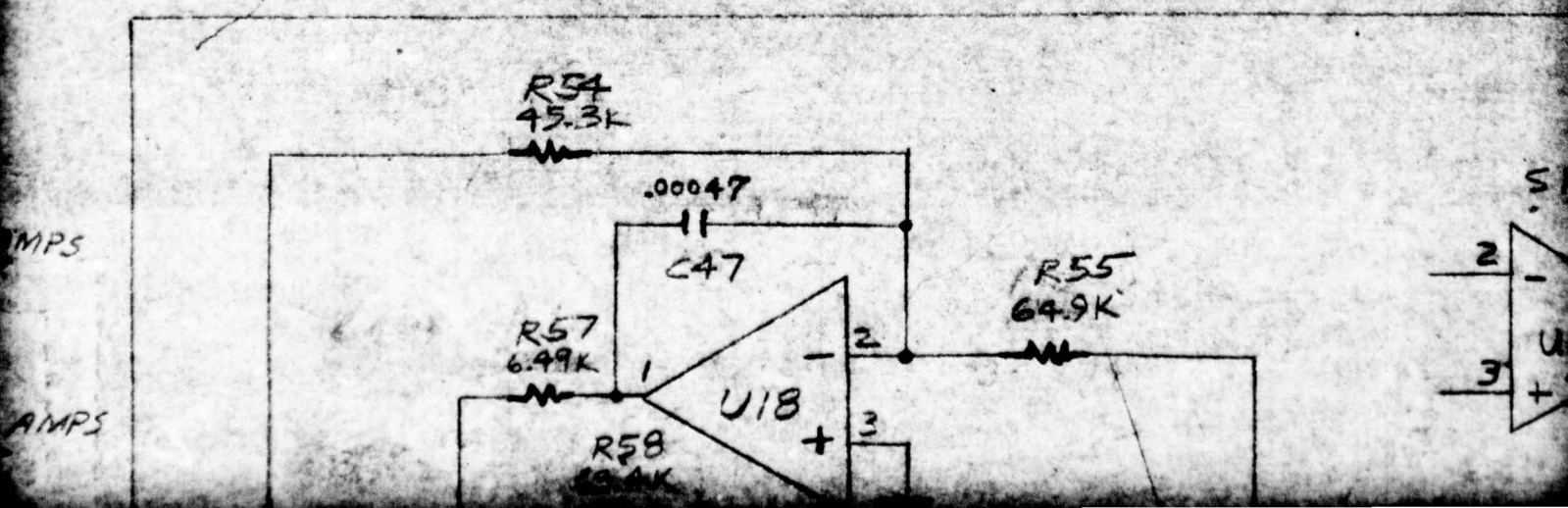
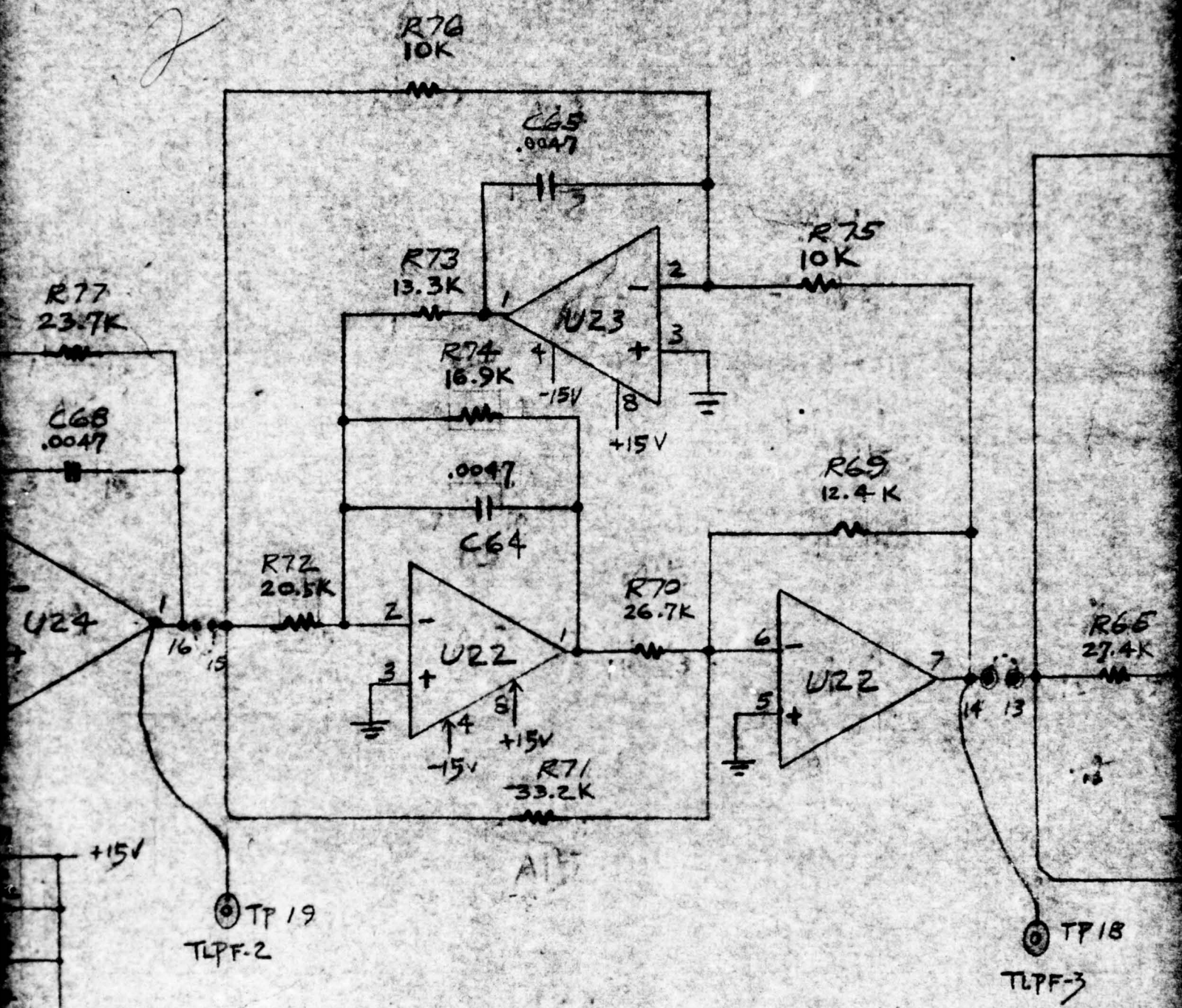
A

CONTRACT NO.	SIZE C	CODE IDENT NO. 25420	DRAWING NO. FIG. 3-18
	SCALE		SHEET 10 OF 10

6

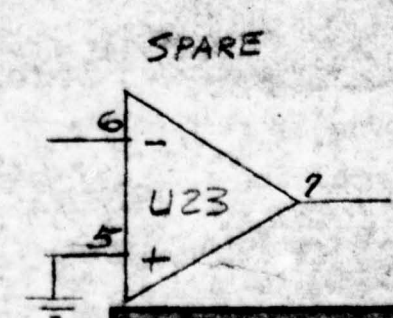
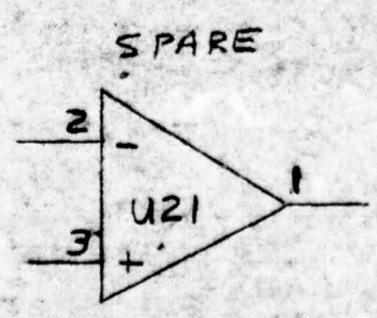
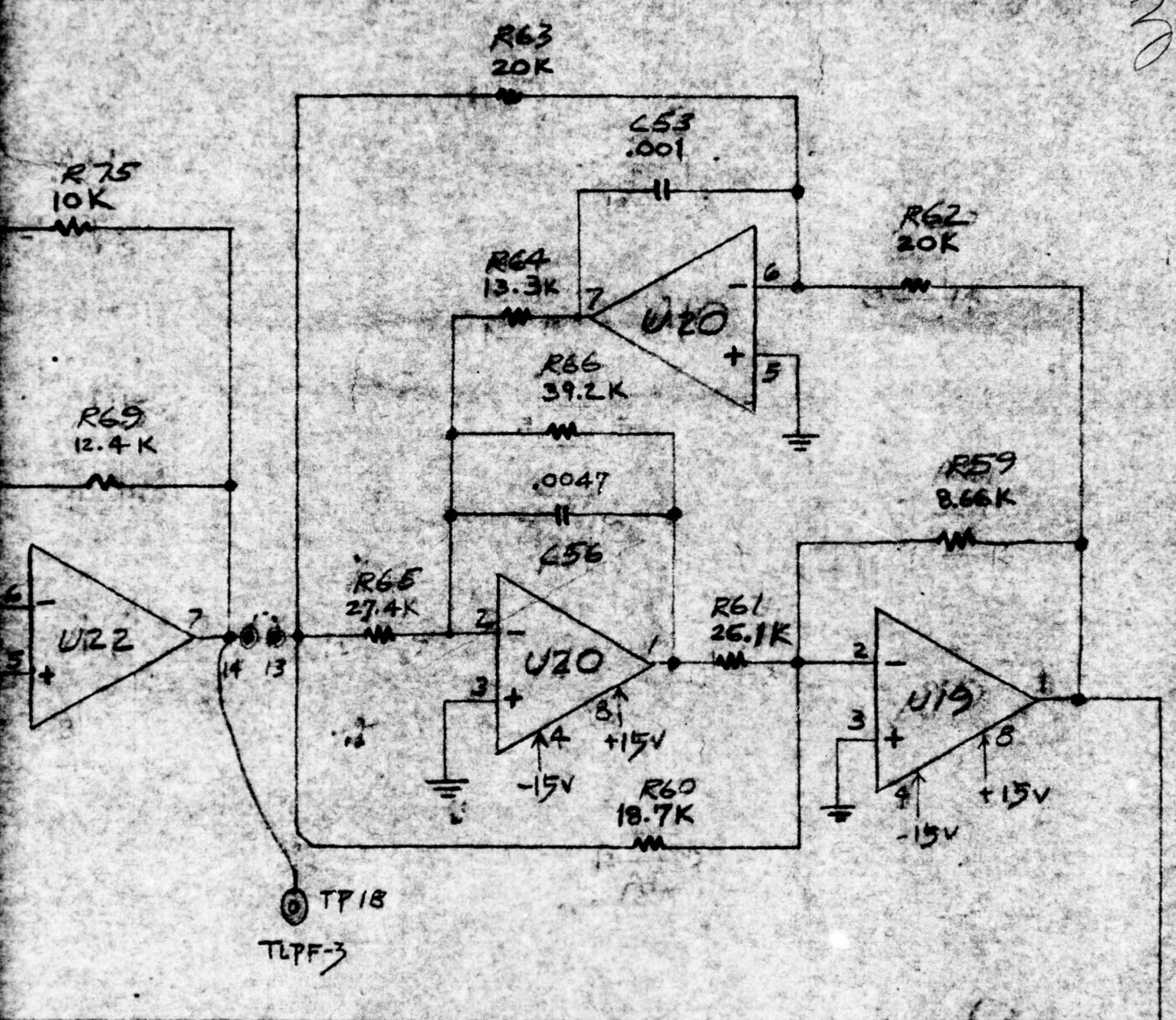


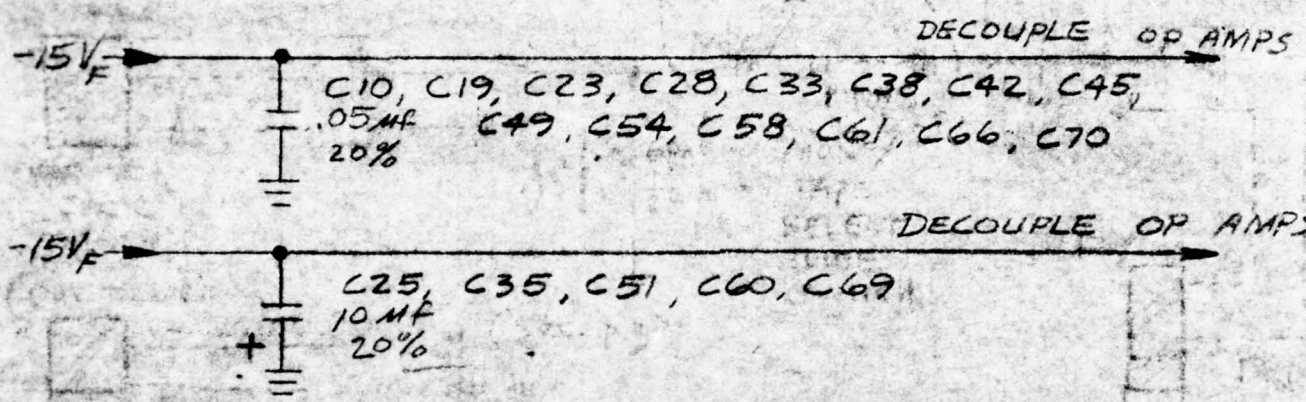
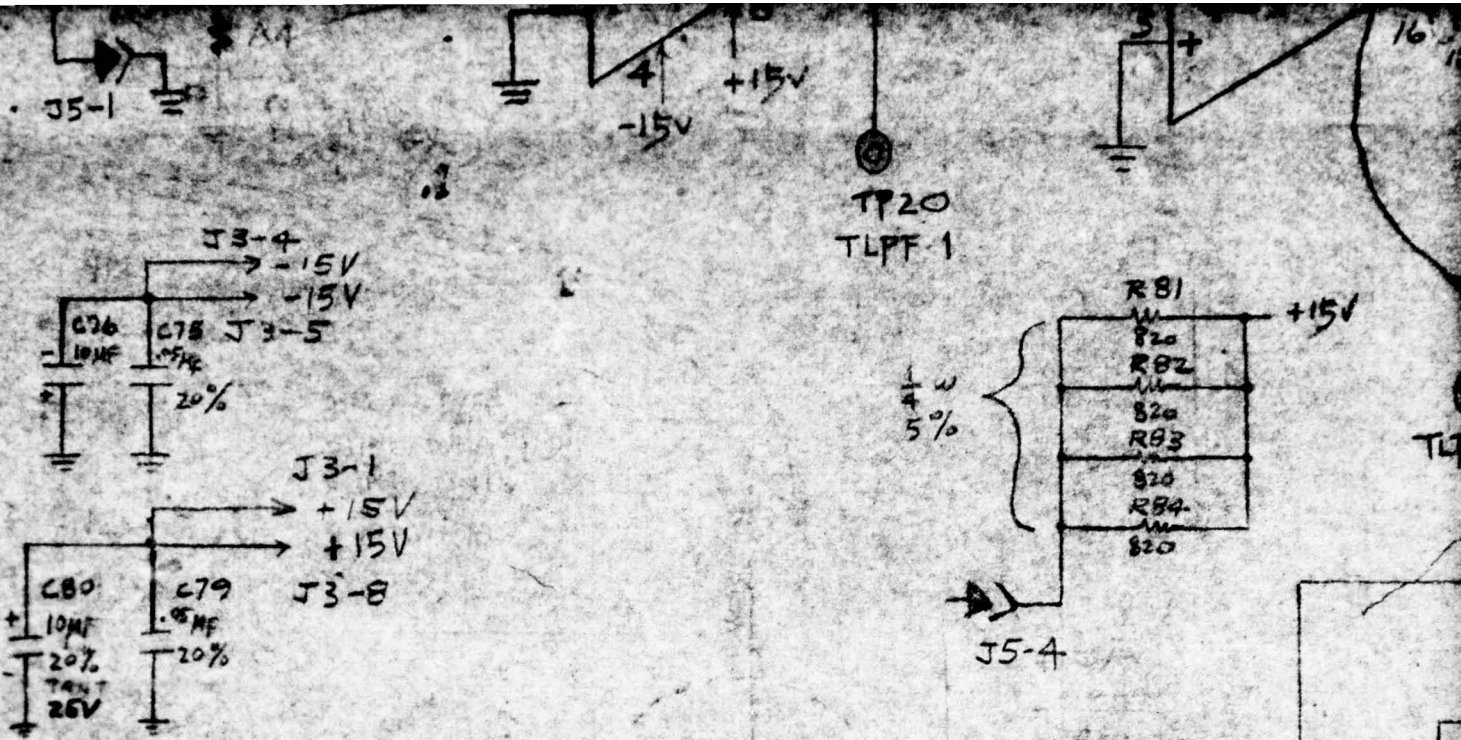
2



Jan 21 76

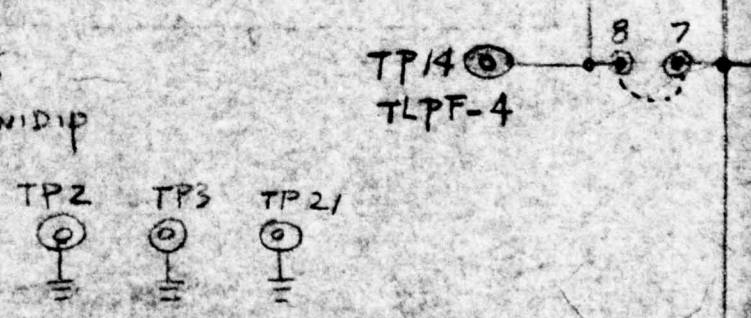
3

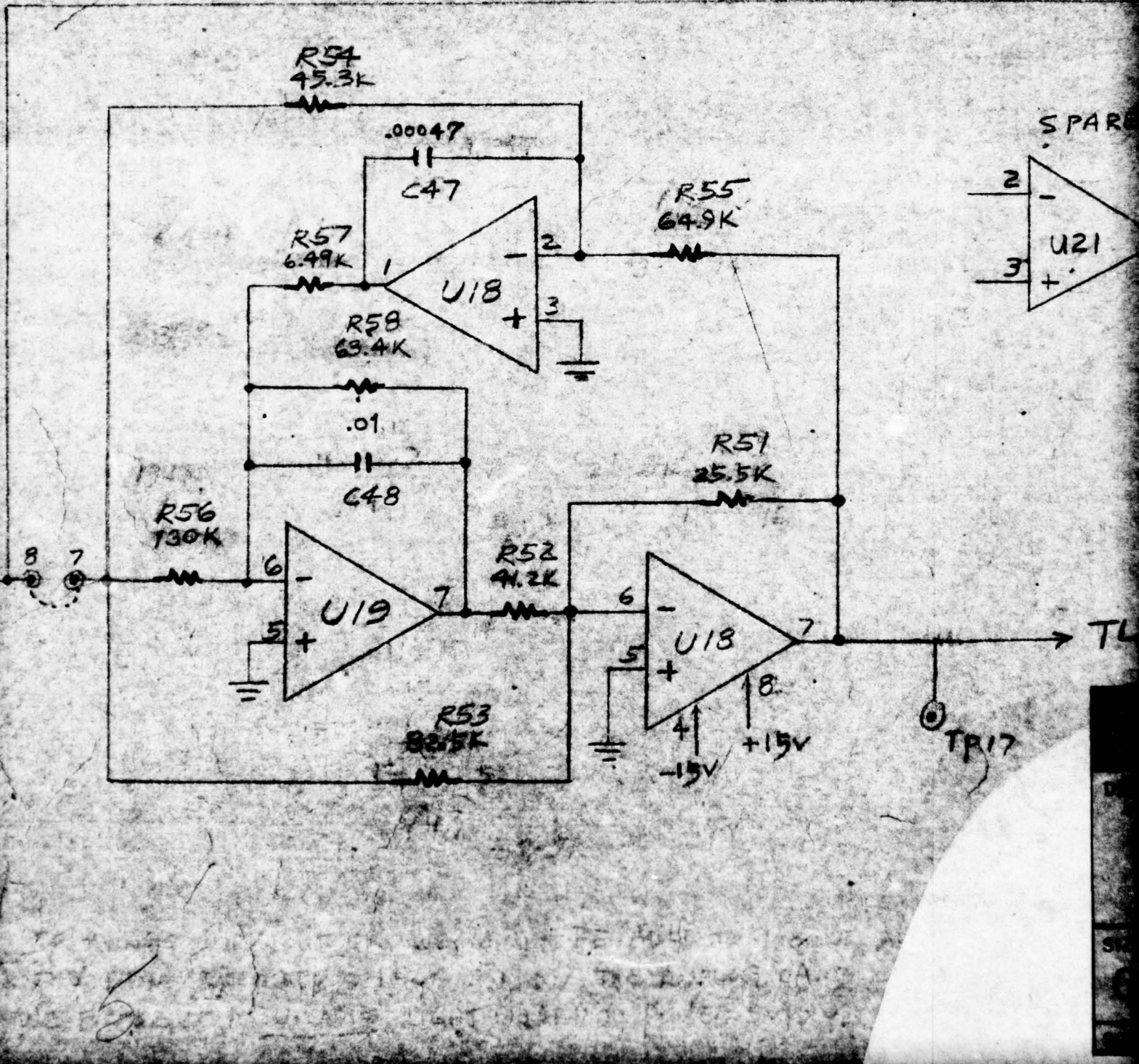
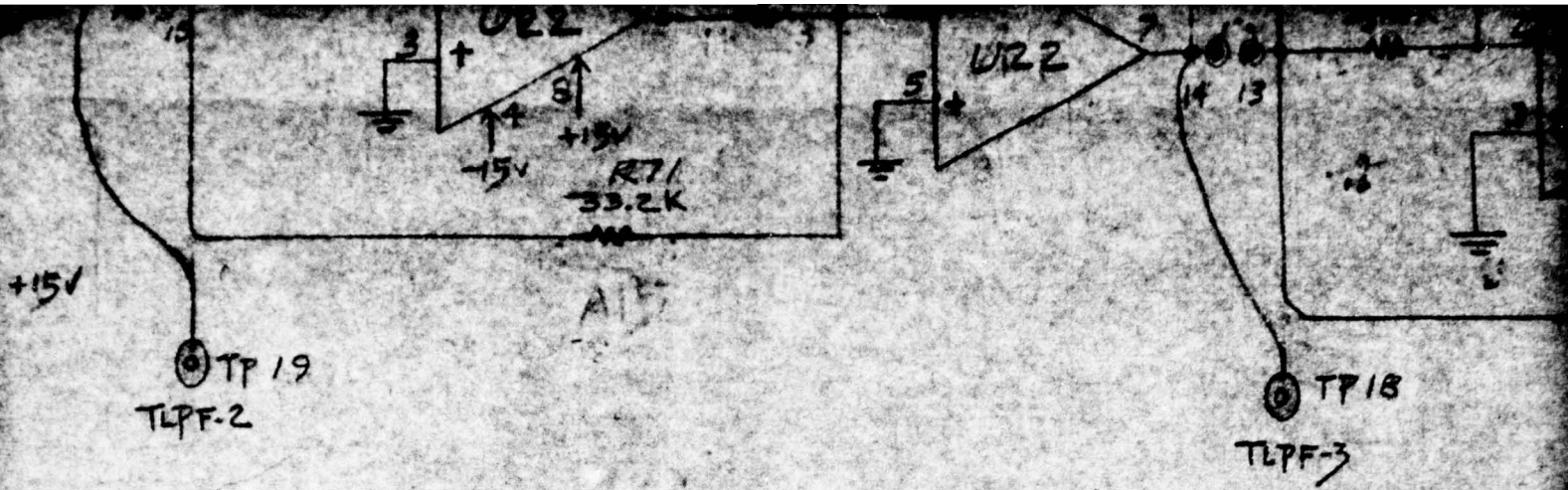


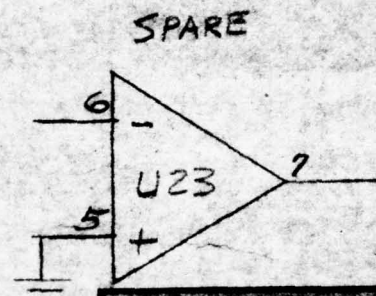
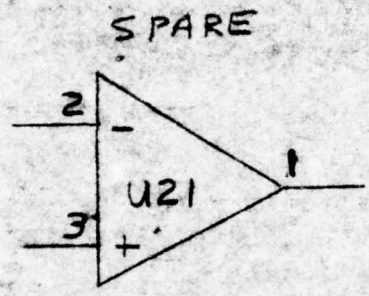
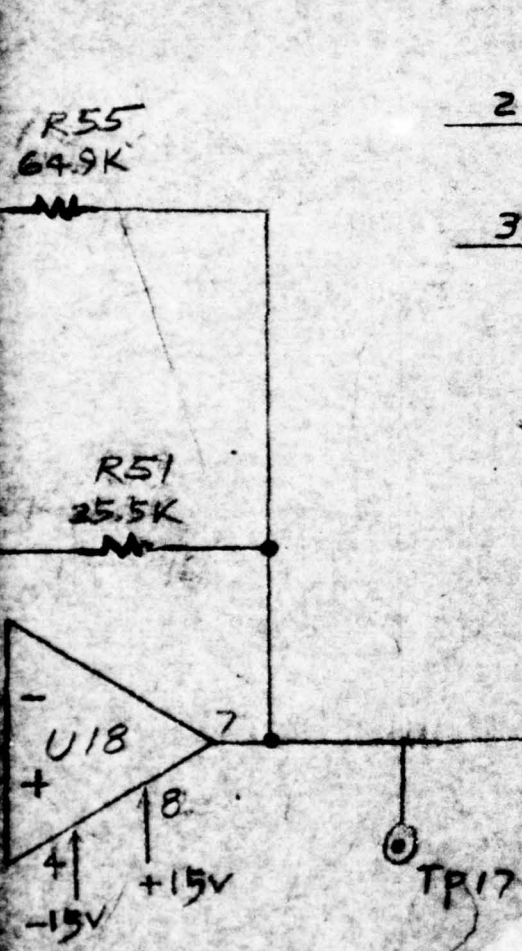
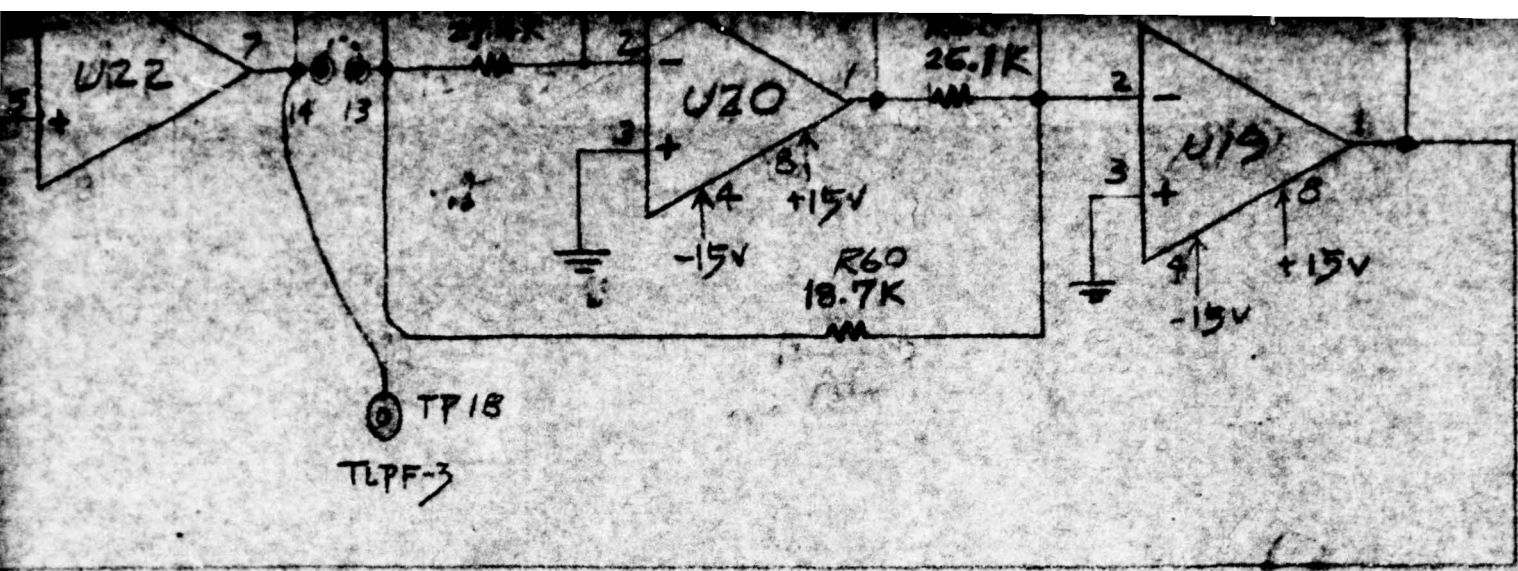


- UNLESS OTHERWISE SPECIFIED:
- ALL RESISTORS ARE $\pm 1\%$, $\frac{1}{10}W$
 - ALL CAPACITORS IN μF , $\pm 1\%$
 - ALL OP AMPS MC 1458, MINIDIP
 - ALL DIODES 1N914
 - POWER SUPPLIES $\pm 15V$

Decouple op amps
 POWER-SUPPLY PINS WITH







TLPFout
TO P/2

NEWTON, MASSACHUSETTS 02195

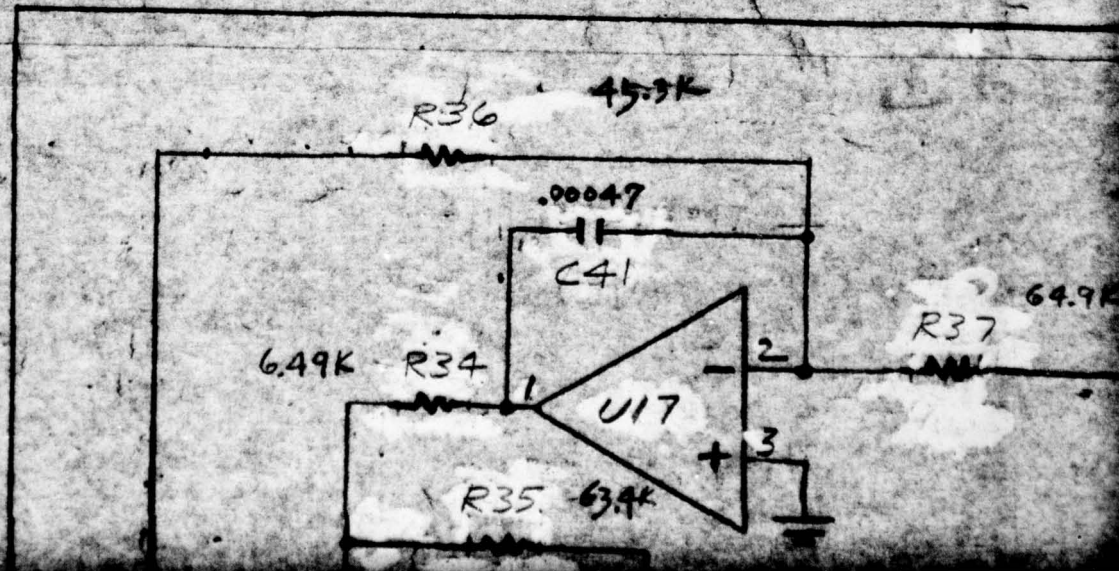
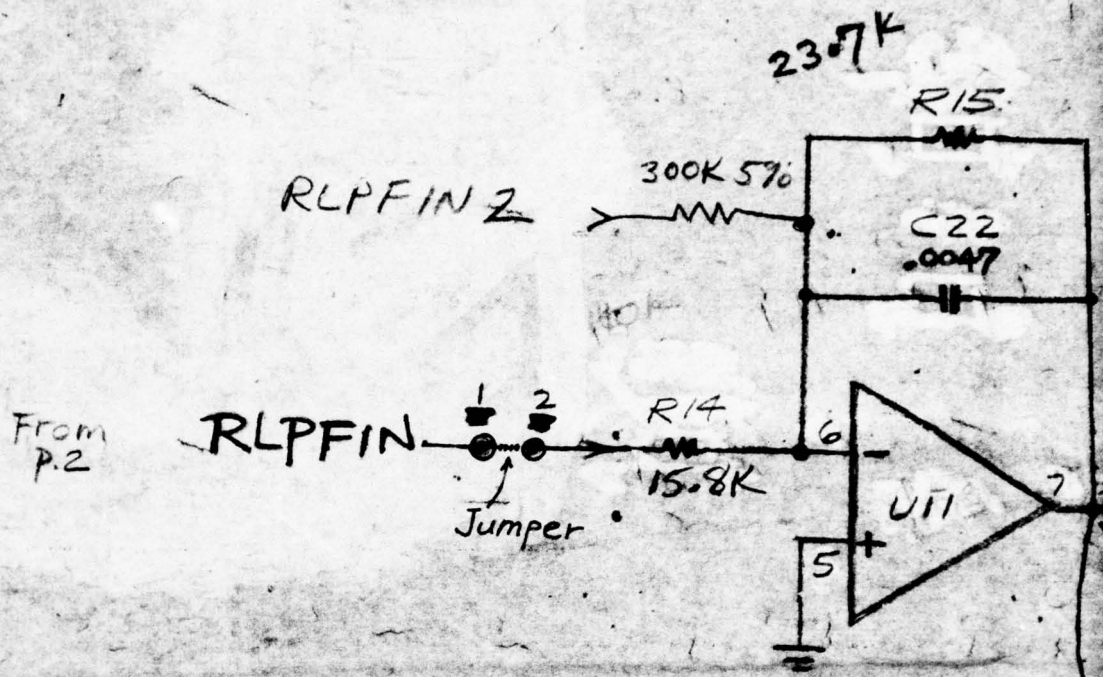
codex corporation

DRAWING TITLE

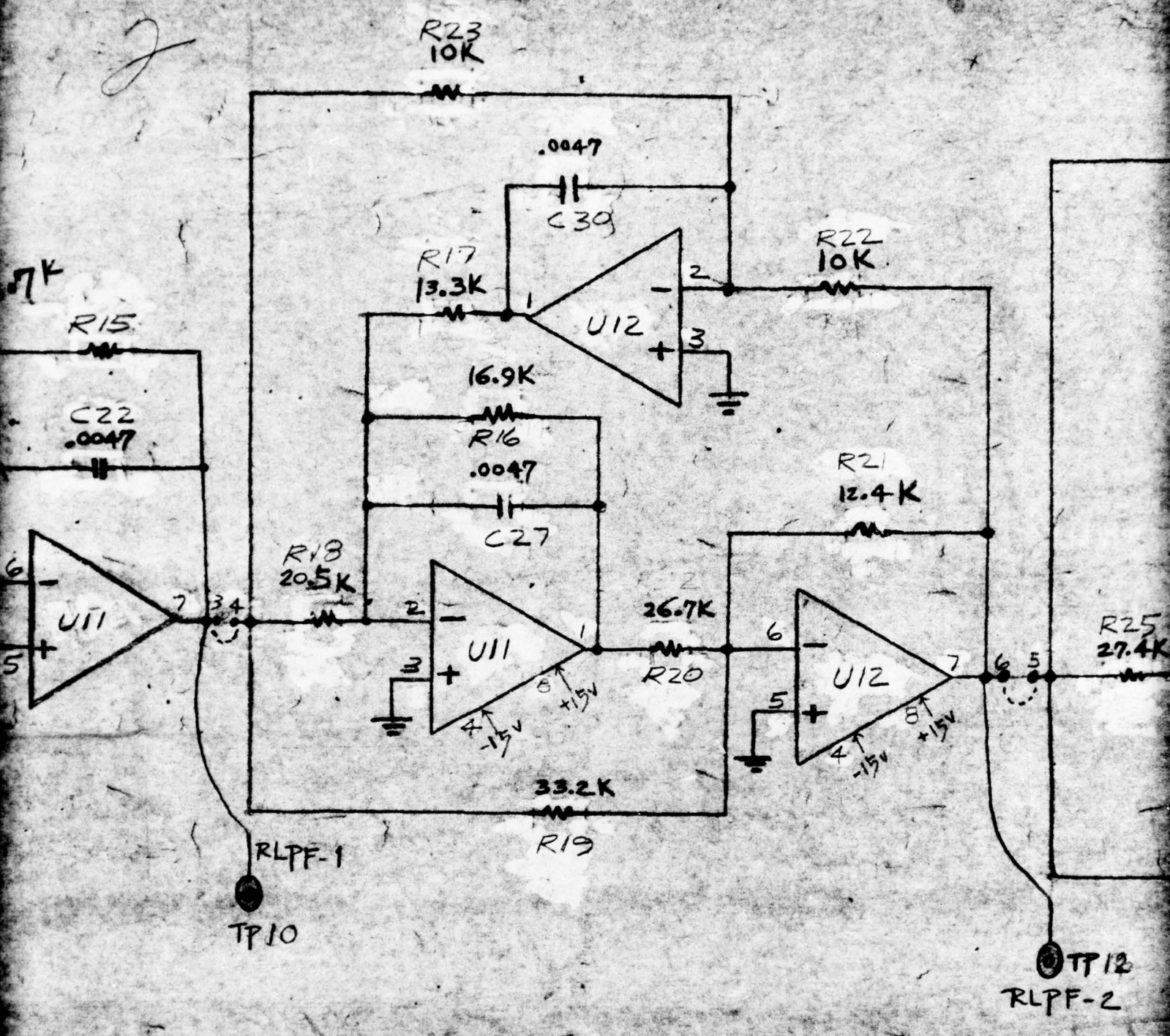
TRANSMIT FILTER

SIZE	CODE IDENT NO.	DRAWING NO.
C	25420	FIG. 3-19
SCALE	SHEET OF	

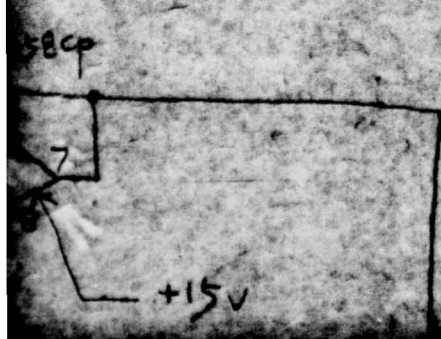
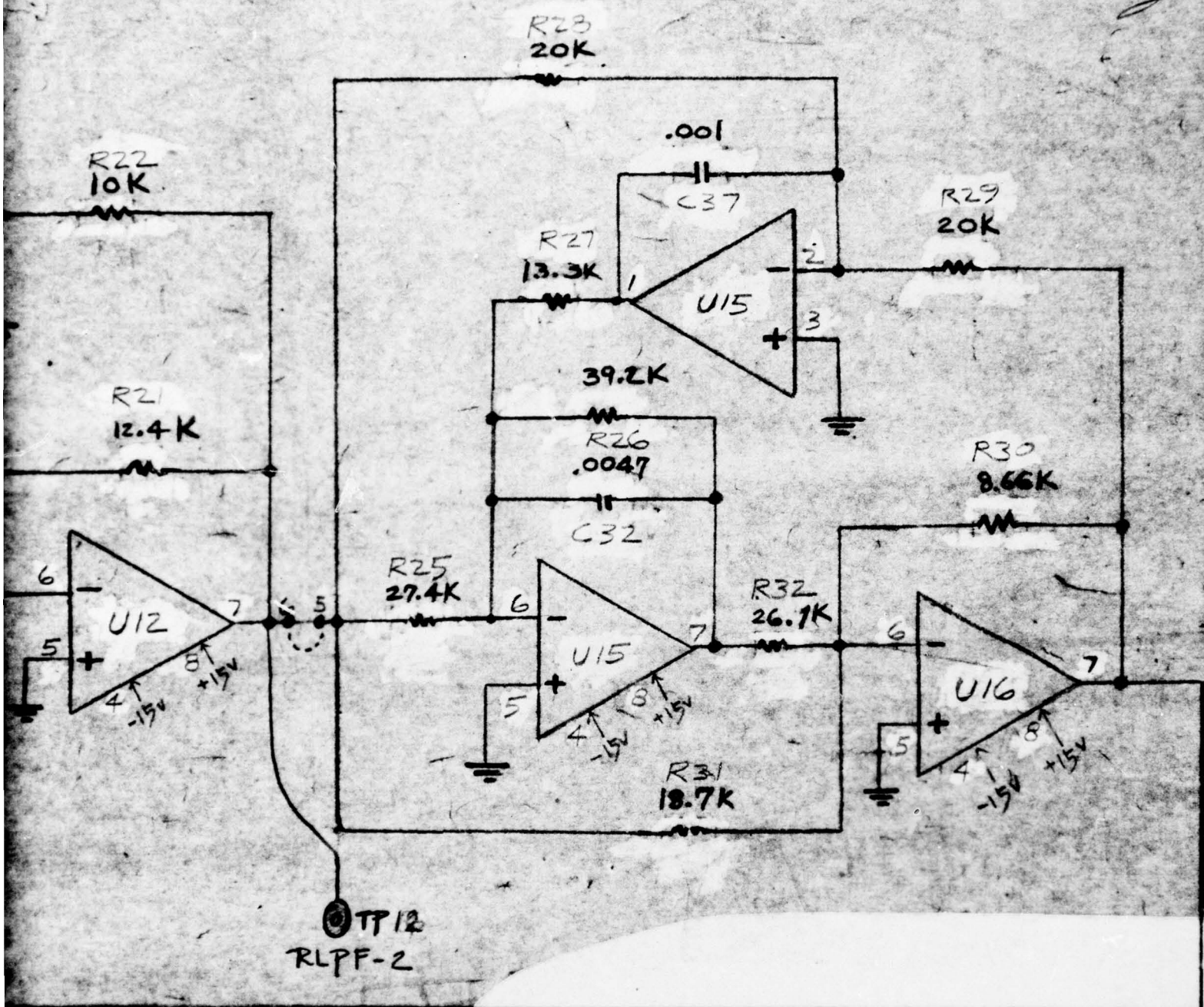
6



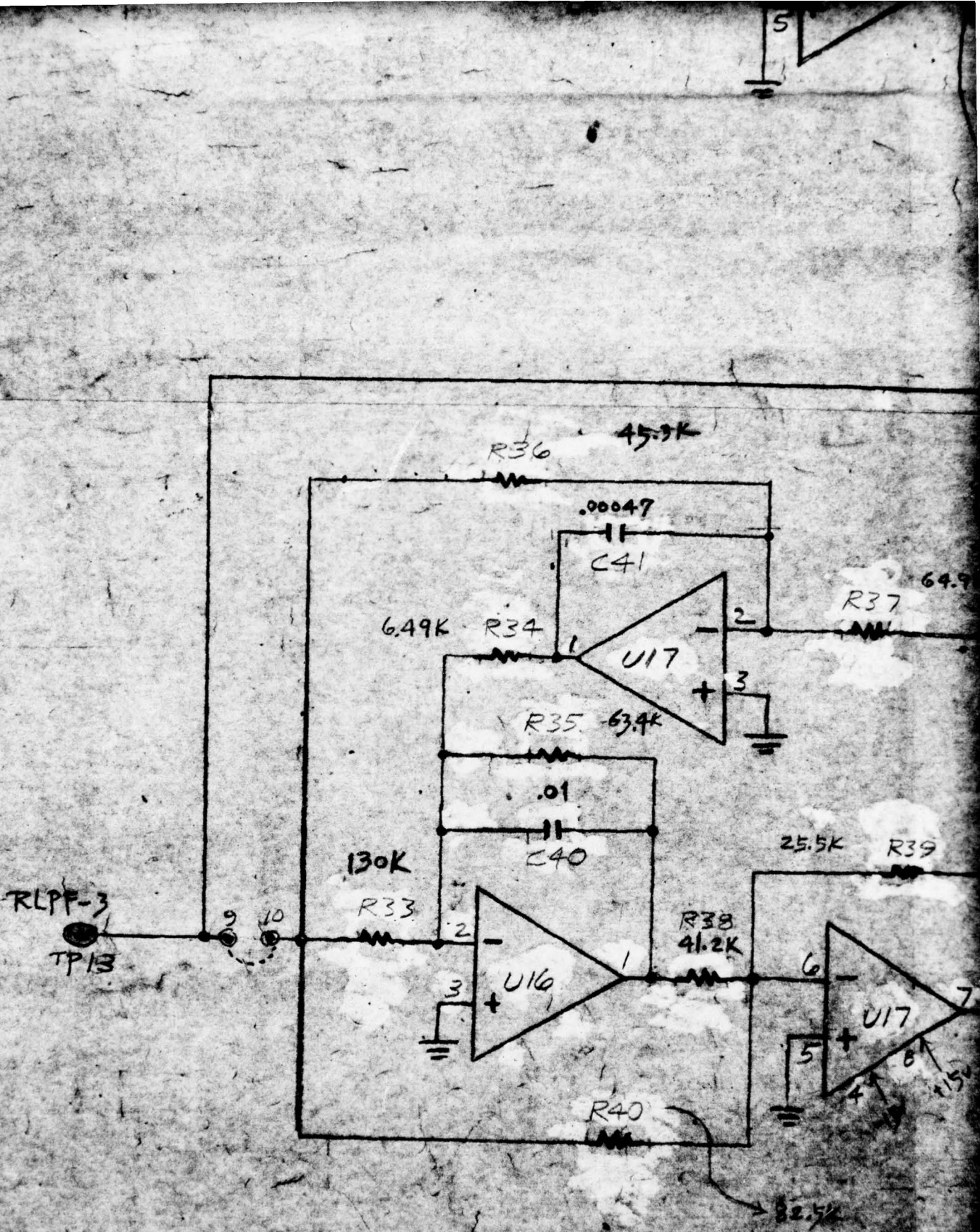
2

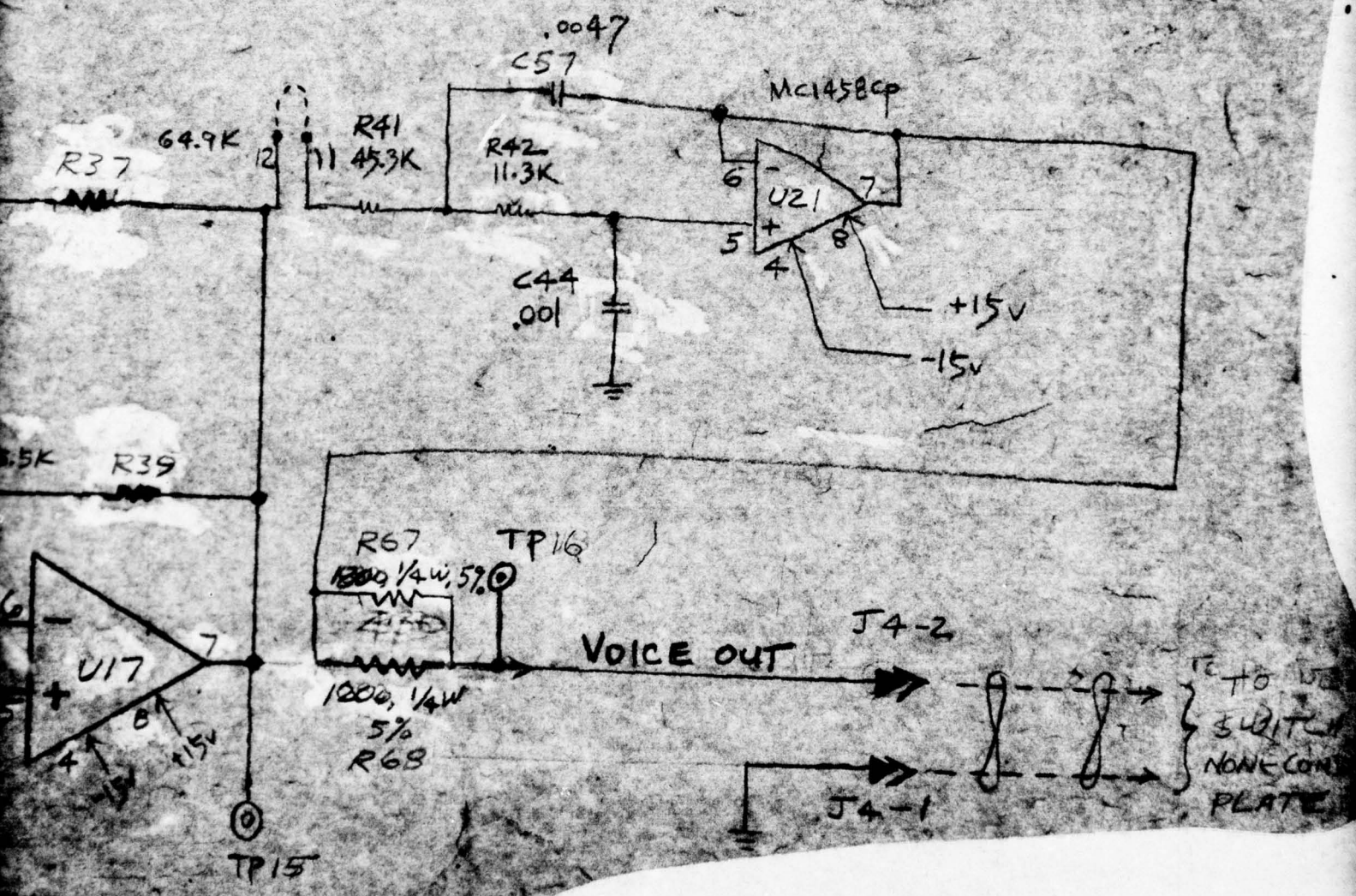
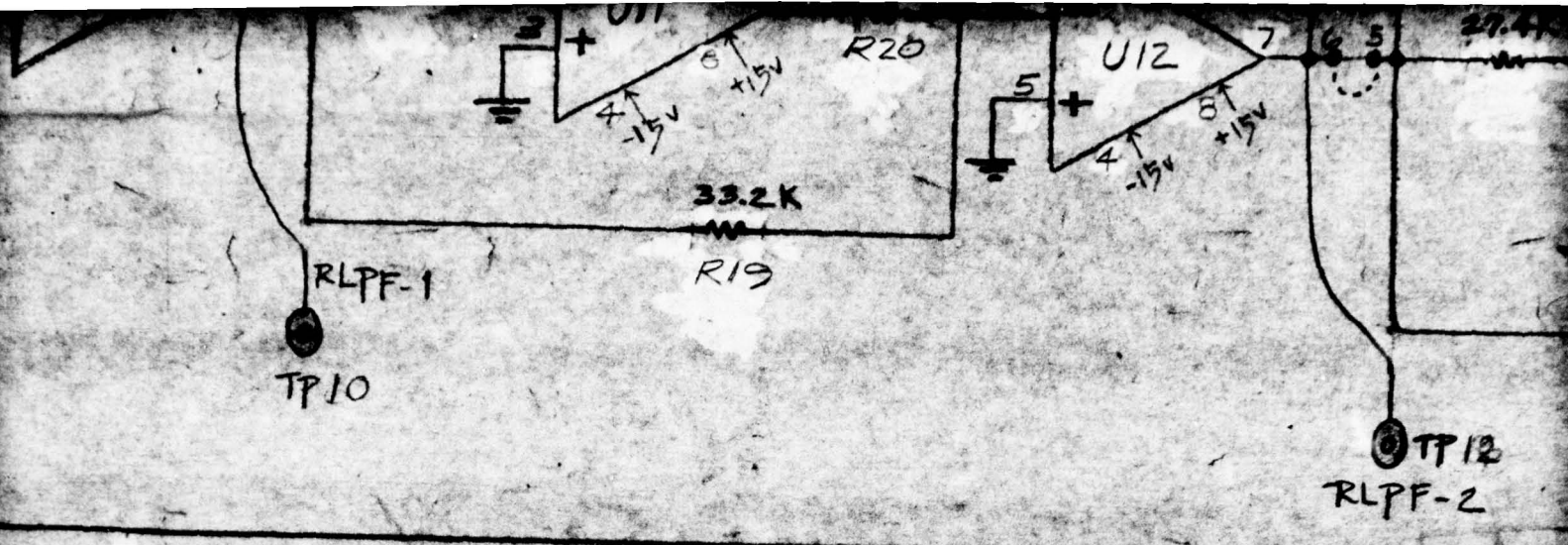


Jan 2-76
3



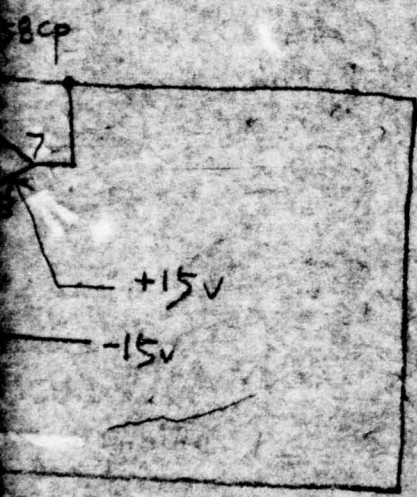
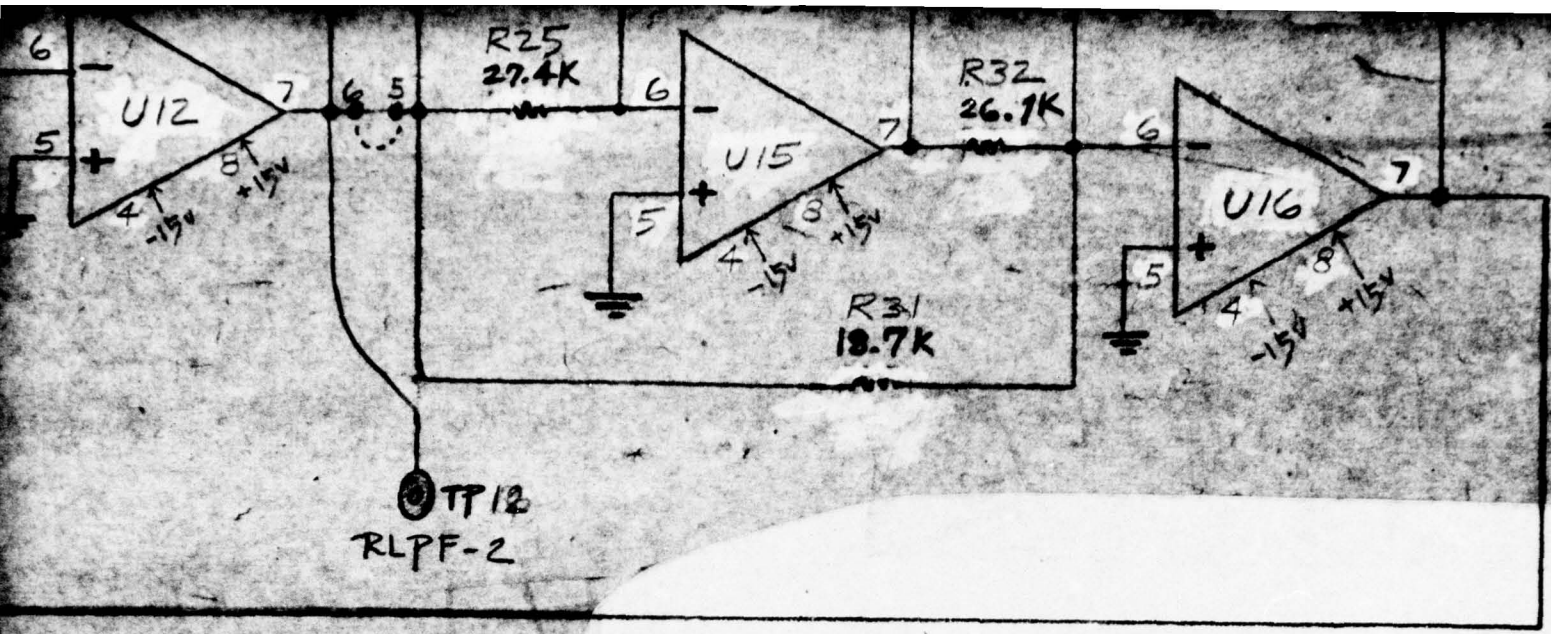
DR	
CHK	





5
Jan 2, 76

TE TO VO
3 UNIT
NON-CON
PLATE



DR	
CHK	
A	
P	
P	
D	
RELEASED	
CONTRACT NO.	

NEWTON, MASSACHUSETTS 02195

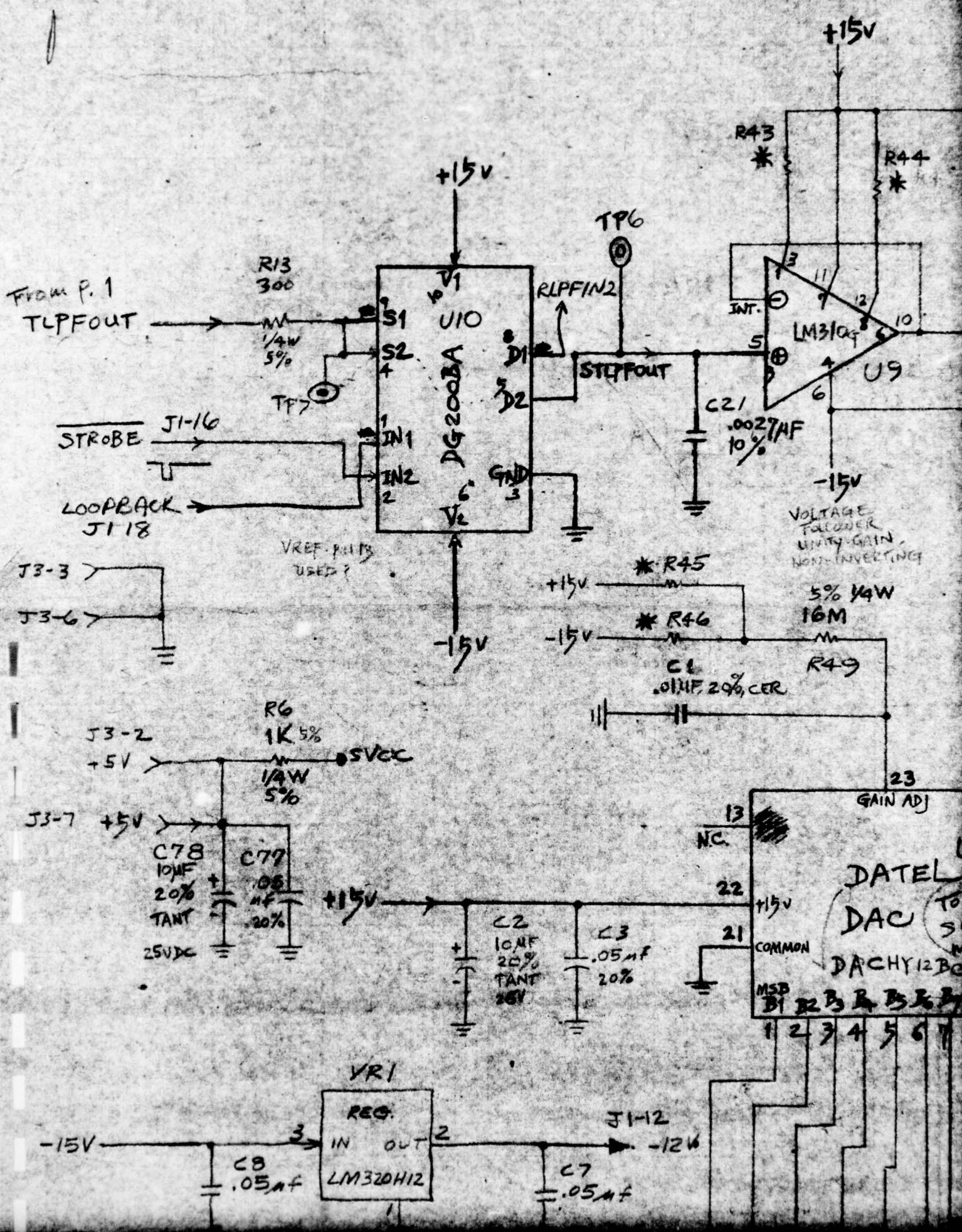
codex corporation

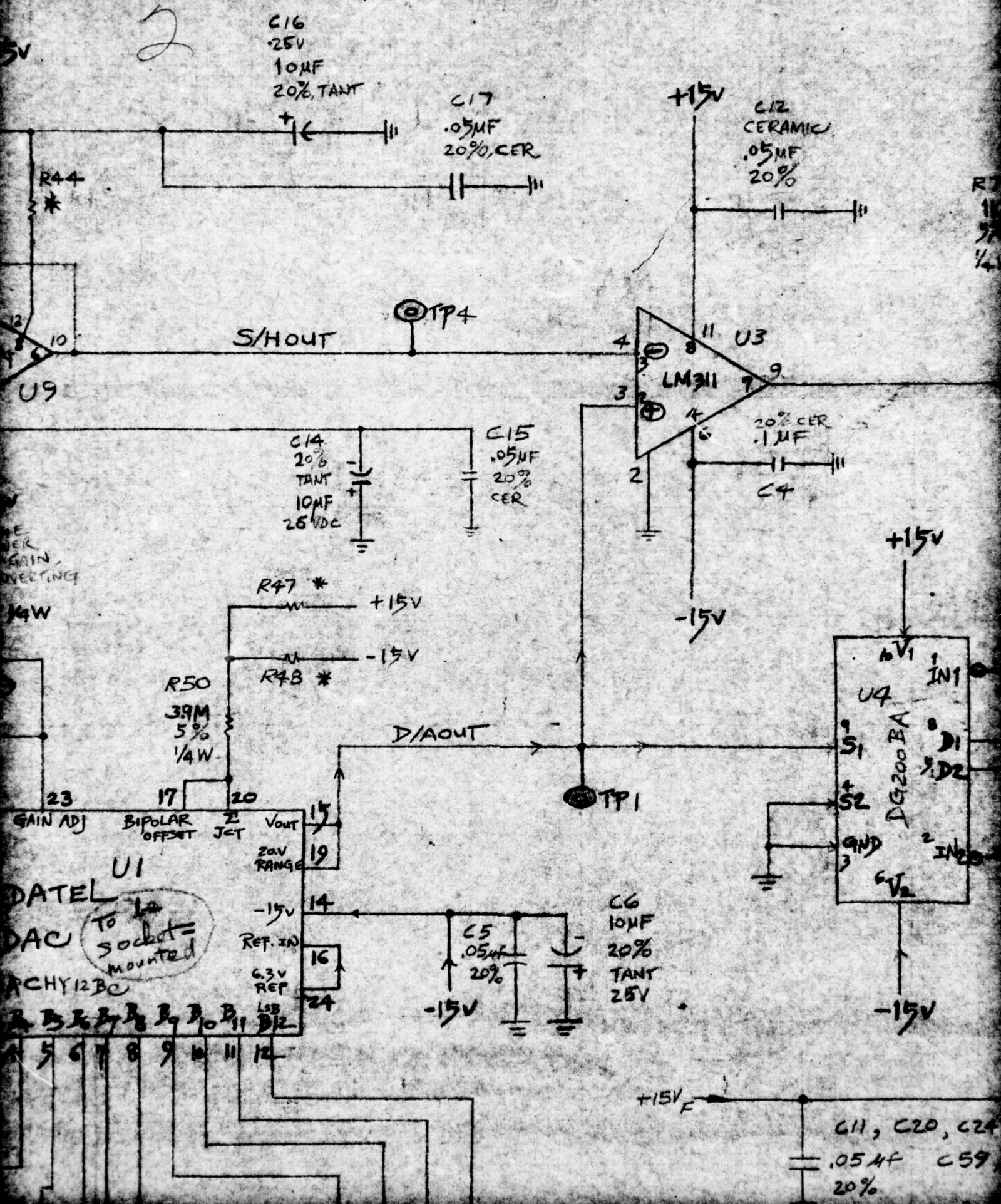
DRAWING TITLE

RECEIVE FILTER

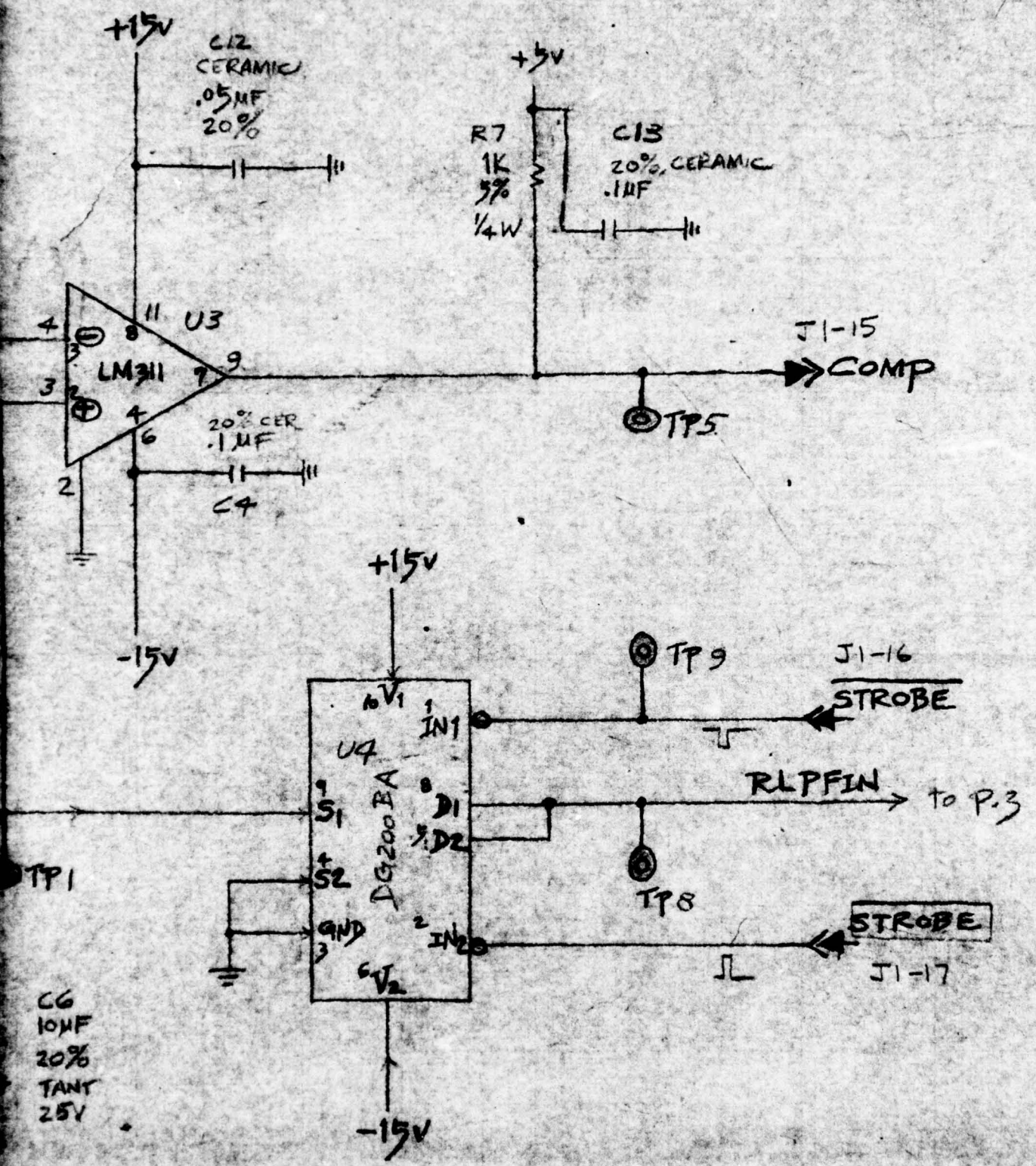
SIZE	CODE IDENT NO.	DRAWING NO.
C	25420	FIG. 3-20

SCALE _____ SHEET _____ OF _____





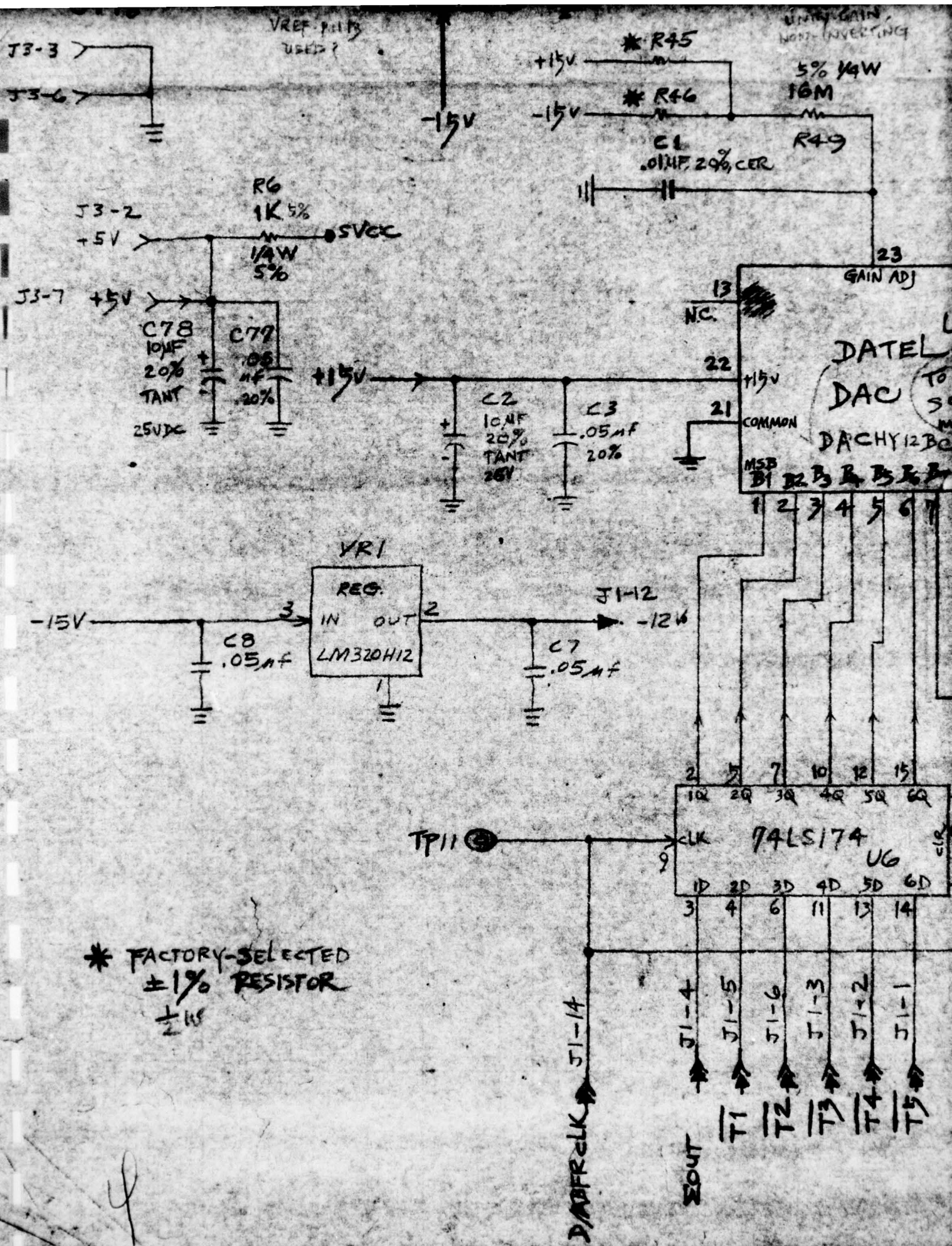
Jan 2, 76 2
3



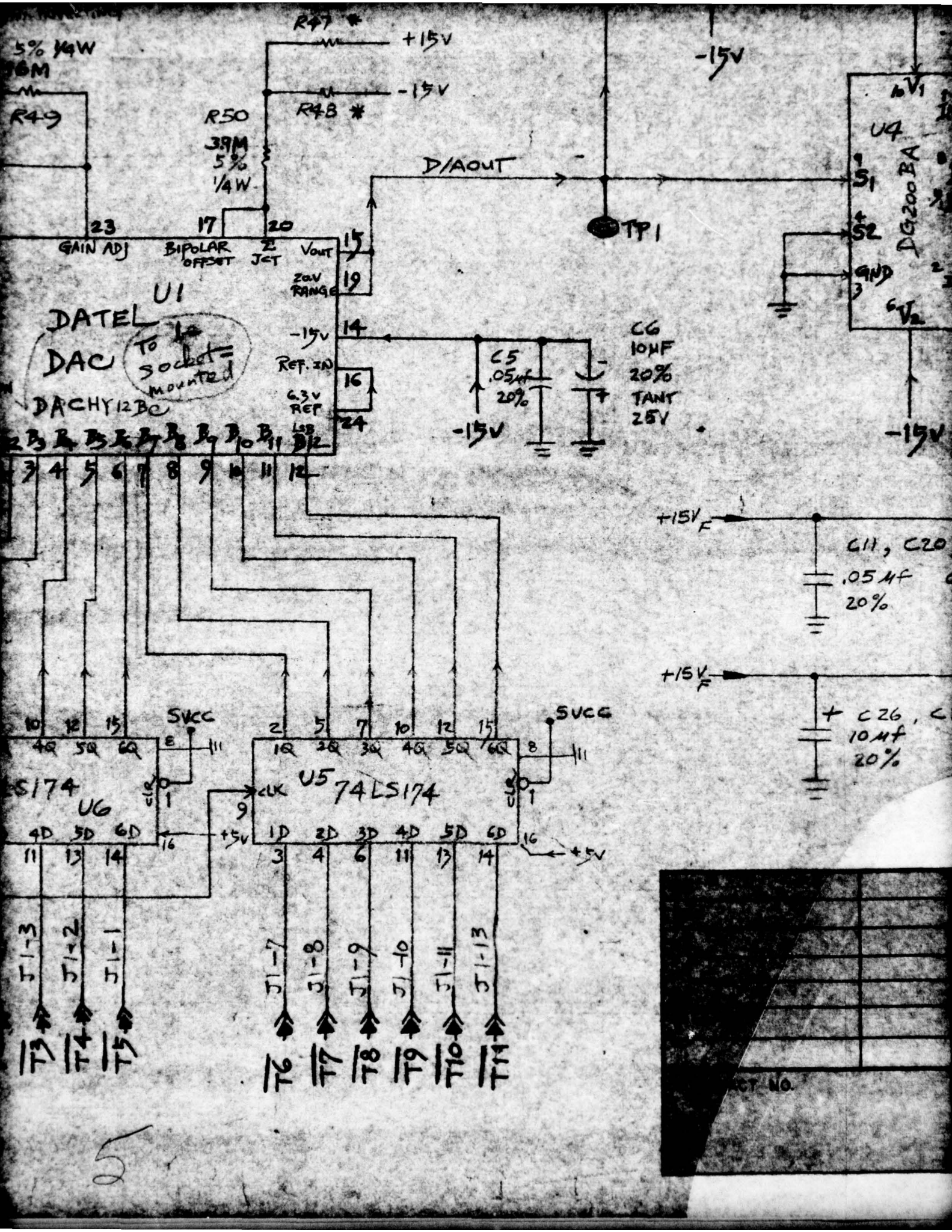
C6
10µF
20%
TANT
25V

DECOUPLE OP AMPS

C11, C20, C24, C29, C34, C39, C43, C46, C50, C55
= .05µF C59, C62, C67, C71
20%



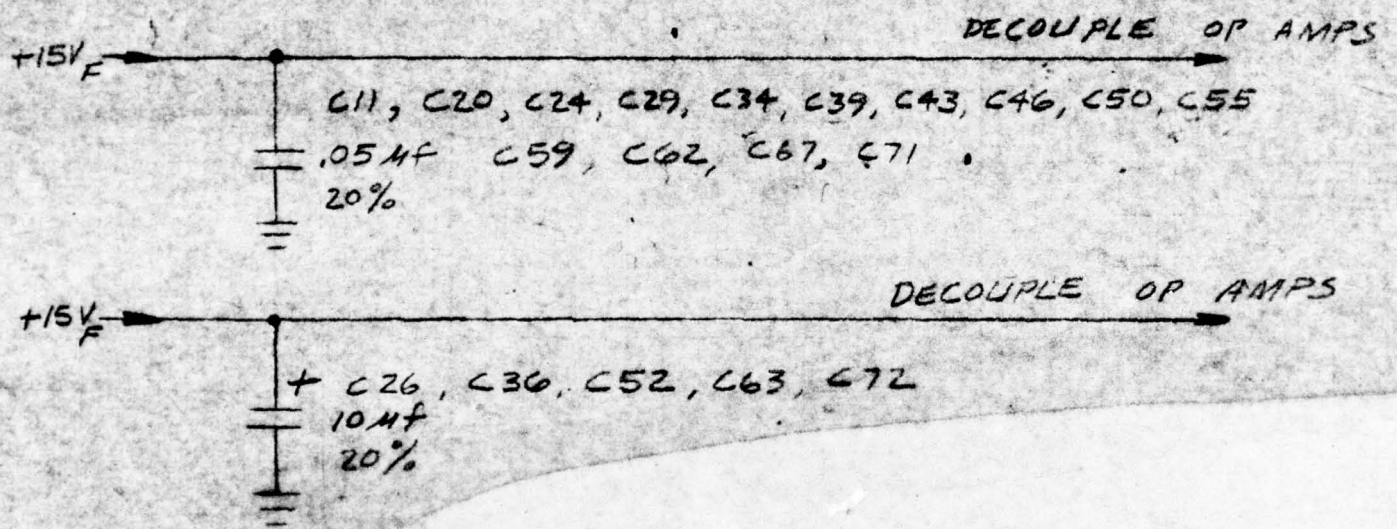
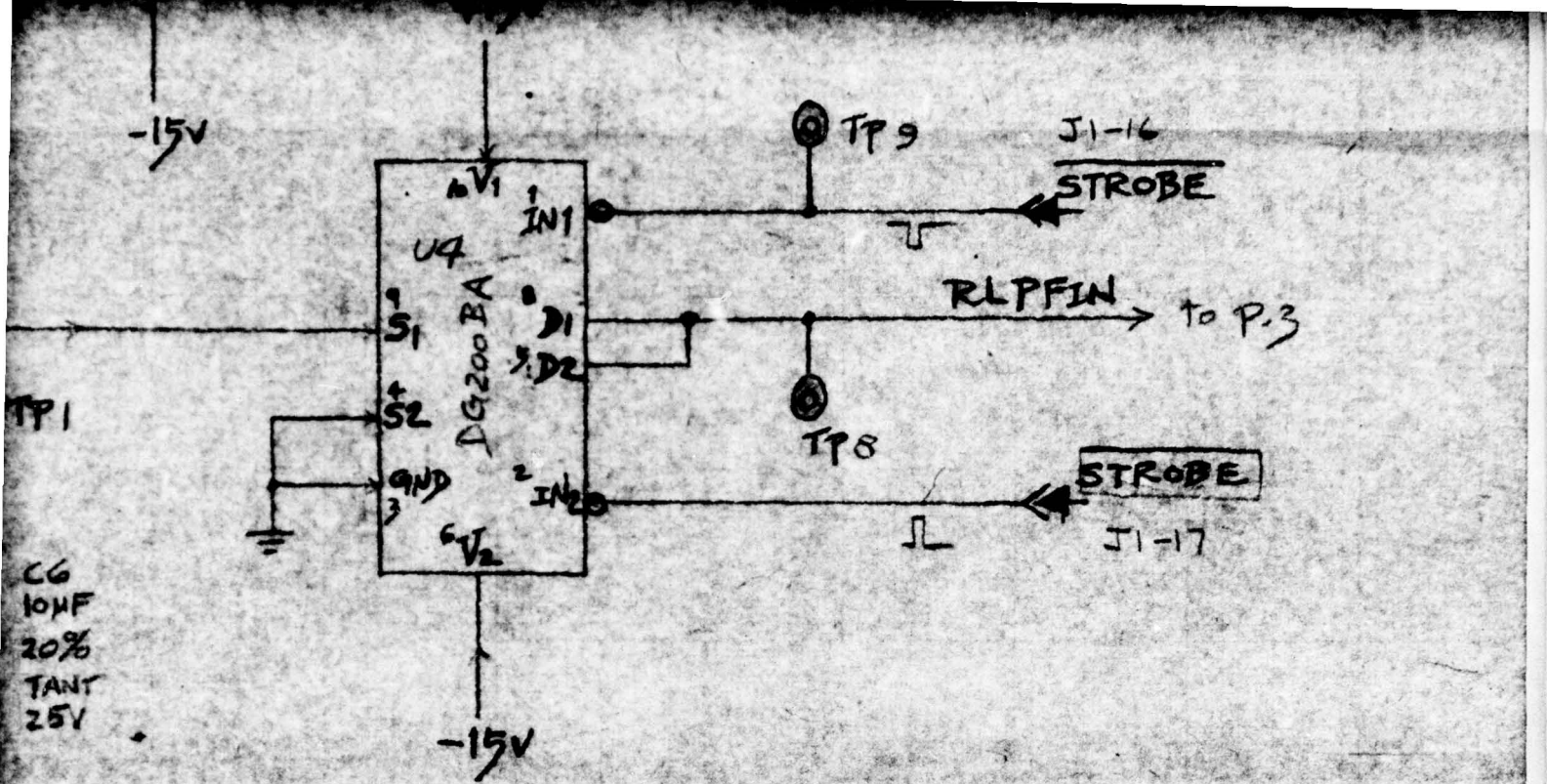
* FACTORY-SELECTED
 ±1% RESISTOR
 1/4W



5



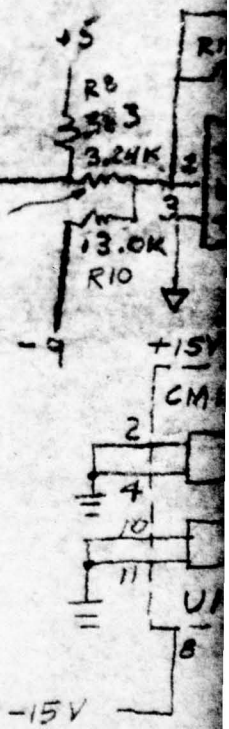
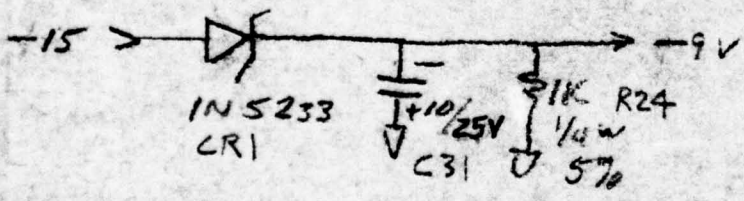
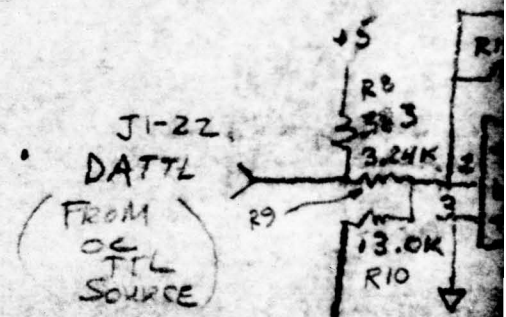
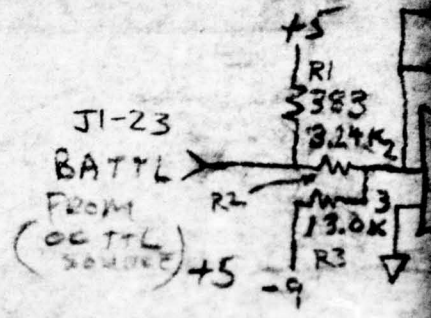
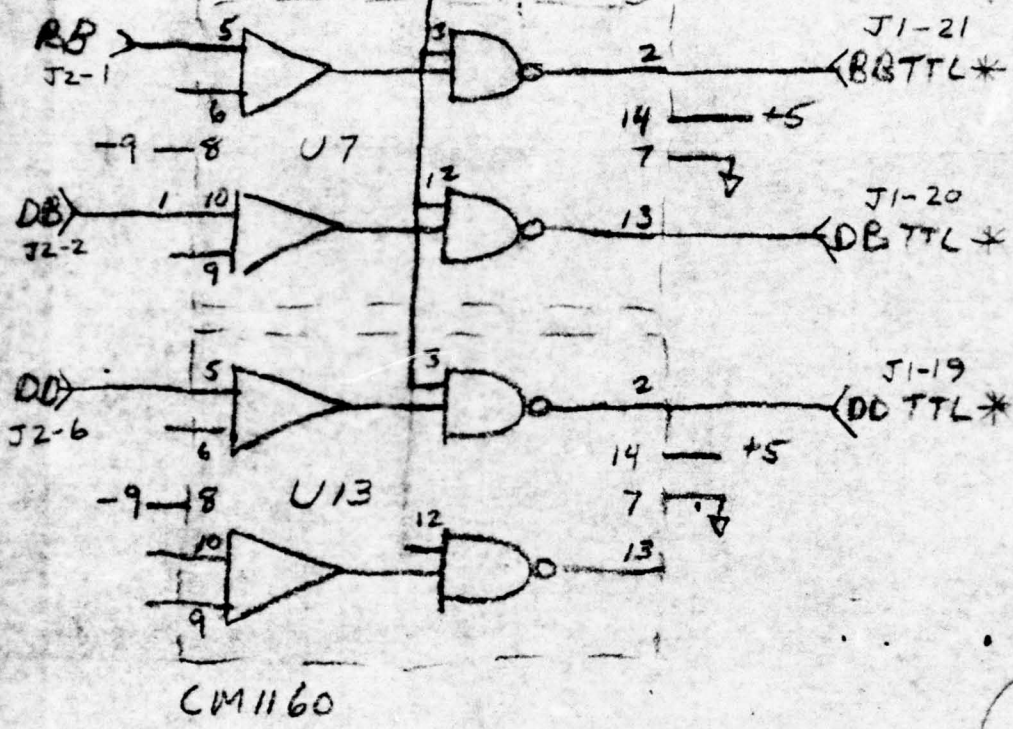
RT NO.



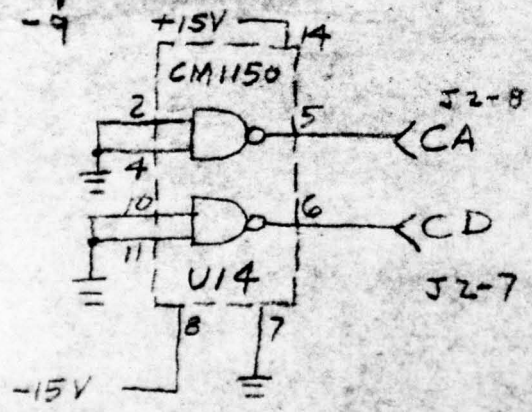
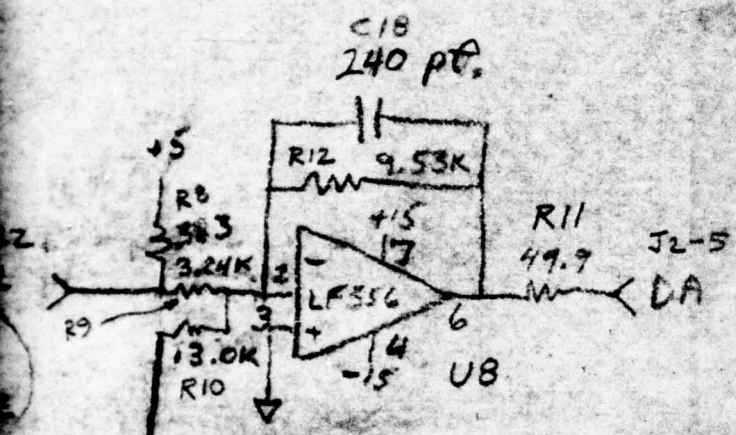
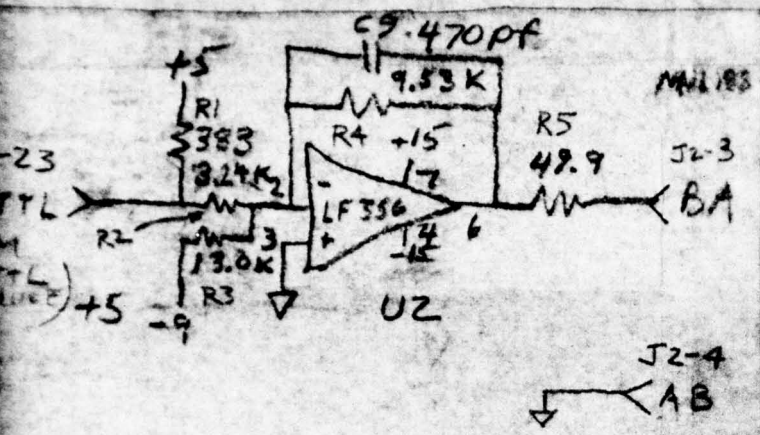
		NEWTON, MASSACHUSETTS 02195	
DRAWING TITLE			
<h1 style="margin: 0;">DIGITAL ANALOG</h1>			
FACT NO.	SIZE	CODE IDENT NO.	DRAWING NO.
	C	25420	FIG. 3-21
SCALE		SHEET OF	

6

MIL 138 CM1160 SVCC



3



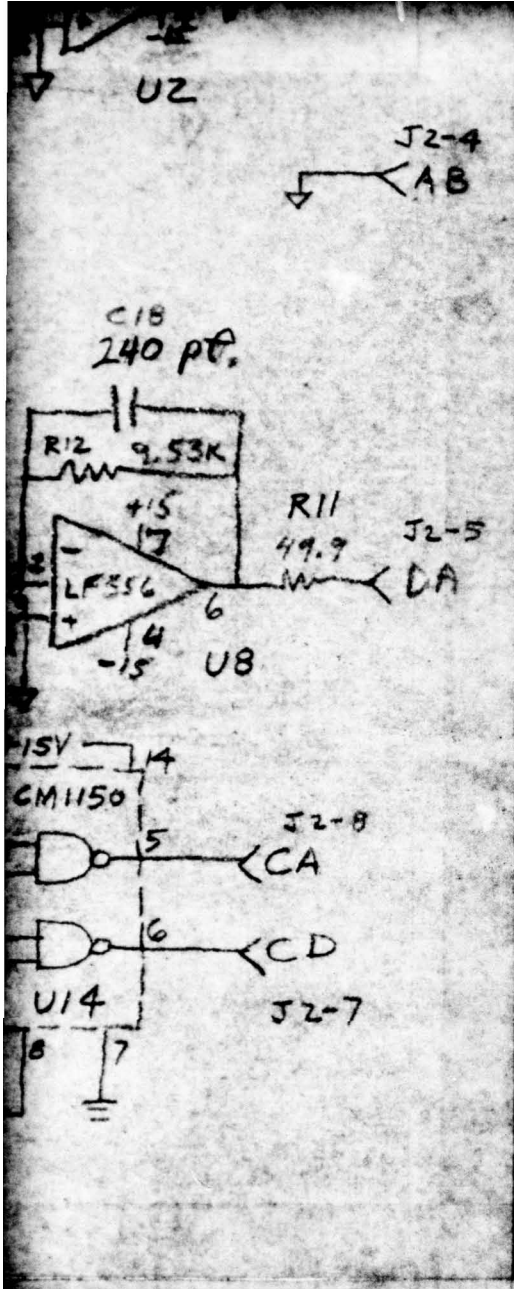
DR			
CR			

NEWTON, MASSACHUSETTS


CODEX CORPORATION

DRAWING TITLE

INTERFACE



3

DR		NEWTON, MASSACHUSETTS 02198	
CN			
		DRAWING TITLE	
		INTERFACE	
CONTRACT NO.		SIZE	DRAWING NO.
		C	FIG. 3-22
		SCALE	SHEET