R. L. Best

Memorandum 6M-4874 B077

Sheet 1 of 9 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: SPEED OF FLUX REVERSAL WITH SLOW RISE-TIME DRIVE

To: Distribution List

From: Sydney Bradspies

Date: January 21, 1957

Approved: R. L. Best

Richard L. Best

Abstract: A great deal of difficulty has been encountered in attempts to plot the characteristics (switching time as a function of driving currents) of thin magnetic films. If the driving current rise time is too fast, the film output is hidden by noise due to inductive pickup. If the rise time is very slow, the film switches noiselessly, but the switching time is slowed down, and the results obtained may not be too meaningful. In an effort to overcome this latter difficulty, an equation has been developed that relates the effective driving current rise time, T_r, the observed switching time, T_s, and the switching time, T, that would occur if a current step were applied. The equation is valid only if $T_s > T_r$. It is:

$$\cos \frac{\pi(T_s - T_r)}{T} = - \frac{\sin \frac{\pi T_r}{T}}{\frac{\pi T_r}{T}}$$

When a test is run on a magnetic sample, T_s and T_r can be readily measured. It is then a simple task to compute T, with the aid of the accompanying graph of this equation. The above equation was devived with the aid of only one assumption-- that when a current step is applied, the voltage output

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

waveshape is a half sine wave. This assumption is shown to be reasonable.

Data is taken for two types of ferrite cores, one metallic tape core, and one thin magnetic film. In all cases, the results seem to uphold the equation stated previously.

Distribution List:

Taylor, N.H.
Brown, D.R.
Papian, W.N.
Mitchell, J.L.
Best, R.L.
Ellis, D.H.
Guditz, E.A.
Davidson, G.A.
Raffel, J.
Bradspies, S.
Rising, H.K.
Smith, D
Goodenough, J.
Zopatti, R.
Menyuk, N.
Canty, W.J.

Abstract only to Group 63, Staff, not listed.

I. INTRODUCTION:

In his thesis, H.K. Rising,[1] has demonstrated that "the instantaneous voltage output of a (magnetic) core (with a rectangular hysteresis loop) is a function only of the instantaneous net drive and the instantaneous flux state of the core". Rising took photographs of core outputs for several methods of excitation: he applied a smooth current step to the core and obtained a given voltage waveshape; he obtained the same voltage waveshape even when he applied a serrated current step of the same amplitude. No matter how wide the pulses were spaced when the serrated current was applied, the output voltage waveshape was unchanged.

From these results, Rising concluded that for any core, the plot of normalized voltage as a function of normalized flux is fixed. He proceeded to draw this normalized curve, and he found that it could be approximated very well by any one of three very simple algebraic expressions. He then employed one of these approximations in order to obtain the output voltage of a core for any arbitrary input current. Unfortunately, his result is complicated and would be difficult to use in an attempt to compute the switching time of a core for a given current.

An attempt to plot the characteristics of thin magnetic films (switching time as a function of driving currents) has proven to be very difficult. When the rise time of the driver is rapid, the film output is disguised by noise. When the rise time is slowed down enough to allow undistorted viewing of the output, the rise time is a large portion of the switching time. This makes it hard to correlate current amplitude and switching time.

In an effort to overcome these difficulties, an equation has been devived that relates switching time to rise time and current amplitude. This scheme depends on Rising's work to some extent, but it differs in that a somewhat different approach is employed.

As a result, if the switching time of a core is known for a given rise time of current, it is possible to easily compute the

[1] Rising, H.K., Magnetic - Core Pulse Amplifiers for Digital Computer Applications, S.M. Thesis, M.I.T., Department of Electrical Engineering, June, 1953.

switching time of that core if the same current is applied as a square wave. This equation has been compared to the results of a number of tests on ferrite cores, a metallic tape core, and a thin film. The results obtained on the cores have proven to be correct. The results for the thin film have at least proven to be consistent. It is not possible to check these results in order to prove that they are correct.

II. THE EQUATION:

The derivation of the equation that follows is a tedious task, and the intricate details are of no general interest. I have, therefore, taken the liberty of showing only the major steps in the process, and of describing that which is not explicitly written. If the reader is interested in seeing the particulars, he is invited to consult the author.

The only assumption that has been made in this proof, is that when a step of current $H(t) = H_m u(t)$ (1)

is applied to a core, its output voltage may be expressed as:

$$v(t) = V_m \sin \frac{\pi t}{T} \qquad (0 \leq t \leq T) \tag{2}$$

where
$H(t)$ = driving current as a function of time;
H_m = amplitude of driving current;
$u(t)$ = unit step function;
$v(t)$ = instantaneous core output voltage;
V_m = peak core output voltage;
t = time, and
T = core switching time when a step of current is applied.

These are shown in Fiqure 1.

Assuming that equations in (1) and (2) are true, and that the core has the hysteresis loop shown in figure 2, then it may be shown that the expression for voltage as a function of flux is:

$$\frac{v(\phi)}{V_m} = \left[1 - \left(\frac{\phi}{\phi_m}\right)^2\right]^{1/2} \tag{3}$$

where
$\frac{v(\phi)}{V_m}$ = normalized core voltage as a function of flux;
ϕ = flux state of core; and
ϕ_m = saturation flux of core = $\int_0^{T/2} v(t)\,dt = \frac{V_m T}{\pi}$ (4)

It should be noted that equation (3), is one of the approximations to Rising's normalized voltage-flux curve.

Equations (1) and (2) may be transformed, their ratio taken, and defined as the transfer impedance of the core. After combination with equation (4), it is found that

$$Z(s) \triangleq \frac{V(s)}{H(s)} = \left(\frac{\pi}{T}\right)^2 \frac{\phi_m}{H_m} \frac{s}{s^2 + (\pi/T)^2} \qquad (5)$$

where Z(s) = transfer impedance of the core; and

s = complex frequency

Equation (5), is only valid while the core is switching; that is, during the time interval $0 \leq t \leq T$.

By the use of equation (5), it is possible to compute the output voltage, V(s), for any given driving current, H(s). One case of especial interest is that in which the driving current rises linearly until it reaches its final value and then remains fixed at this value for some time (at least until the switching is completed). This is shown in figure 3.

During the time interval $0 \leq t \leq T_r$, the core output voltage is found to be

$$v(t) = \frac{\phi_m}{T_r}\left[u(t) - \cos\frac{\pi t}{T}\right] \qquad (6)$$

where T_r = effective rise time of driving current (time for current to rise from coercive force to final value); and

T = core switching time when a step of current is applied.

The flux switched during the rise time of the current is,

$$\phi = \int_0^{T_r} v(t)\,dt = \phi_m\left[1 - \frac{\sin\frac{\pi T_r}{T}}{\frac{\pi T_r}{T}}\right] \qquad (7)$$

The total flux to be switched is $2\phi_m$, and so the remaining flux is,

$$\phi_r = 2\phi_m - \phi = \phi_m\left[1 + \frac{\sin\frac{\pi T_r}{T}}{\frac{\pi T_r}{T}}\right] \qquad (8)$$

But the flux that remains is switched by the constant driving field, H_m. As discussed in the Introduction, the core will complete its switching in a sinusoidal manner, thus the remainimg flux is also

$$\phi_r = \int_{T-Y}^{T} V_m \sin \frac{\pi t}{T} dt = \phi_m \left[1 - \cos \frac{\pi Y}{T} \right] \qquad (9)$$

where $Y = T_s - T_r$ = time between end of rise of current and completion of core switching;and

T = core switching time when a step of current is applied.

The limits in equation (9), may be justified by noting that the nature of the core wave form has changed when the current is constant. As far as the core is concerned, it has always had a fixed driving force applied, and so the tail end of the pulse behaves as if it were to switch in time T. This constant current has started at a time Y earlier, and so the time interval during which the step is applied is from T-Y until T.

Equations (8) and (9) may be combined so as to eliminate the unwanted variables. The result is that

$$\cos \frac{\pi Y}{T} = - \frac{\sin \frac{\pi T_r}{T}}{\frac{\pi T_r}{T}} \qquad (10)$$

or

$$\cos \frac{\pi (T_s - T_r)}{T} = - \frac{\sin \frac{\pi T_r}{T}}{\frac{\pi T_r}{T}} \qquad (11)$$

where, $Y = T_s - T_r$;

T_r = effective rise time of driving current;

T_s = core switching time when current with rise time T_r is applied; and

T = core switching time when a step of current is applied.

T_s and T_r can be measured, and equation (11) can be solved for T.

Figure 4, used equation (11) so as to plot T_r/T as a function of T_r/T_s. From this curve, the step-current switching time, T, of any core can be determined, if the effective rise time, T_r, (the rise time from the coercive force to H_m) and the actual switching time, T_s, are both known. For example, in a memory plane, suppose that $T_s = 1.0$ µsec and that $T_r = 0.265$ µsec (effective rise time).

Then, $T_r/T_s = 0.265$. From figure 4, it is observed that $T_r/T = 0.3$. Therefore, $T = .265/.3 = 0.885$ µsec.

But the core would start to switch about 0.25 µsec earlier if a step were applied. Thus, switching could be completed in 0.885 µsec instead of 1.25 µsec. The saving would be about 0.36 µsec. However, the additional noise introduced into the system prevents the use of such a scheme.

From figure 4, it is noted that the actual switching time, T_s, and the possible switching time, T, do not differ by very much until the ratio T_r/T_s reaches about 0.3. When thr ratio exceeds 0.4 the difference between T_s and T begins to grow very large.

III. EXPERIMENTAL RESULTS:

In order to demonstrate the validity of equation (11), a large quantity of data was taken for several different magnetic materials: two different types of ferrite cores, one metallic tape core, and one thin film. For each experiment a driving current was established, and the rise time, T_r, was varied over a wide range of values. The switching time, T_s, was measured for each value of T_r. The rise time and the switching time were recorded according to my definitions of them as shown in figure 5.

The expected value of switching time, T, upon application of a current step was computed (for the ferrite and metallic tape cores only) by the well known relation,

$$T = \frac{S_w}{H_t - H_c} \qquad (12)$$

where, T = core switching time when a step of current is applied;

S_w = switching coefficient of the core;

H_t = total driving current; and

H_c = coercive force of core.

For each value of driving current, T was computed from equation (11) for the core materials. Then the ratio of T_r/T was plotted against T_r/T_s. On the same graphs, the theoretical curve (figure 4) is also shown.

Figure 6 shows the results obtained for a standard TX-O ferrite memory core--DCL-2-854 HU-1, size F397.

Figure 7 shows the results obtained for a slow switching, high coercive force ferrite memory core--DCL-3-75-1, size F394.

Figure 8 shows the results obtained for a metallic tape core made of 10 wraps of Mo Perm--1/8 mil x 3/4 inch diameter x 1/8 inch wide.

In each of these three cases, the correspondence between the experimental points, and the theoretical curve is very close, as long as the ratio $T_r/T_s < 0.8$. When the current rise time is made slower than this, the results are poor. There are several possible explanations for this phenomenon. In the first place, when $T_r/T_s > 0.8$, the theoretical curve rises very steeply, and a small error in T_r/T_s causes a great error in T_r/T. In the second place, as the switching time is increased, its measurement becomes less accurate because of the definition of figure 5. This is due to the fact that more of the flux reversal is not included in the measurement of T_s (because the output voltage is not really sinusoidal--the trailing edge is smooth as shown in figure 5, and not sharp as in the idealized case). This means that the recorded T_s is smaller than the actual value of T_s. The result is that for large values of T_r/T_s, the calculated value of T_r/T_s is made larger than it should be. It is noted that in figure 6, 7, and 8, T_r/T_s is larger than the theoretical value. This is to be expected.

Figure 9, shows the results obtained for a thin magnetic film -- number 1B. In this case, it was not possible to compute the theoretical value of T . The value used was the average value computed from each run of data. Once again, it is noted that the experimental and theoretical results correspond closely. The reason is that in each run, the calculated T was almost constant for all readings taken with fixed current.

IV. CONCLUSION:

In conclusion, it seems as though the problem of measuring switching time has been made somewhat simpler by the use of some rather simple computations. This could prove to be valuable in the testing of thin magnetic films.

SB/md

Sydney Bradspies

Sydney Bradspies
Group 63, Staff

DRAWINGS:

Fig. 1,2,3	B-69627
Fig. 4	A-69789
Fig. 5	A-69790
Fig. 6	B-69791
Fig. 7	A-69792
Fig. 8	A-69793
Fig. 9	A-69794

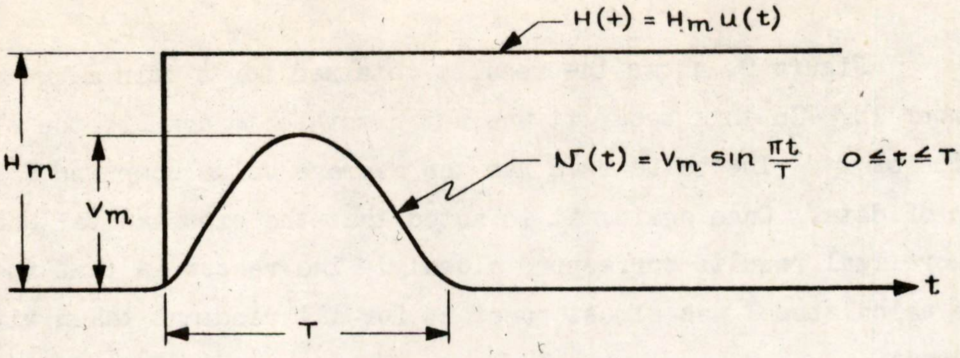


FIG. 1

CORE OUTPUT VOLTAGE WHEN STEP OF CURRENT IS APPLIED

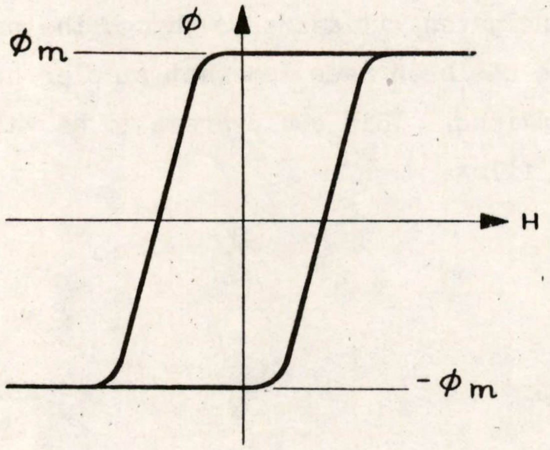


FIG. 2

HYSTERESIS LOOP OF CORE

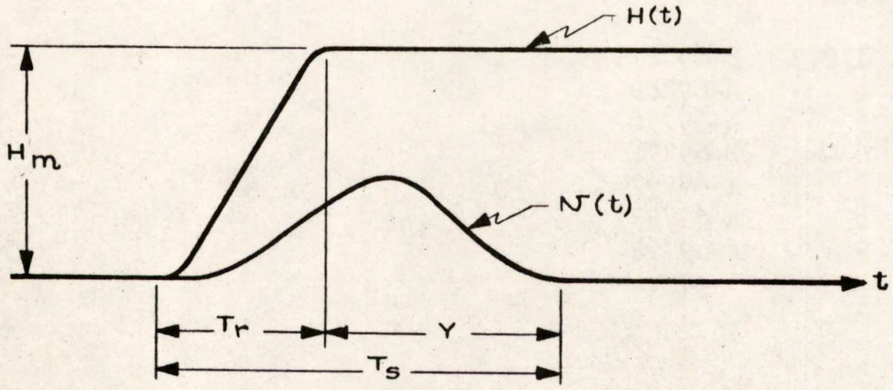


FIG. 3

CORE OUTPUT VOLTAGE WHEN RAMP OF CURRENT IS APPLIED. CORE IS BIASED SO IT STARTS SWITCHING AS SOON AS CURRENT IS APPLIED.

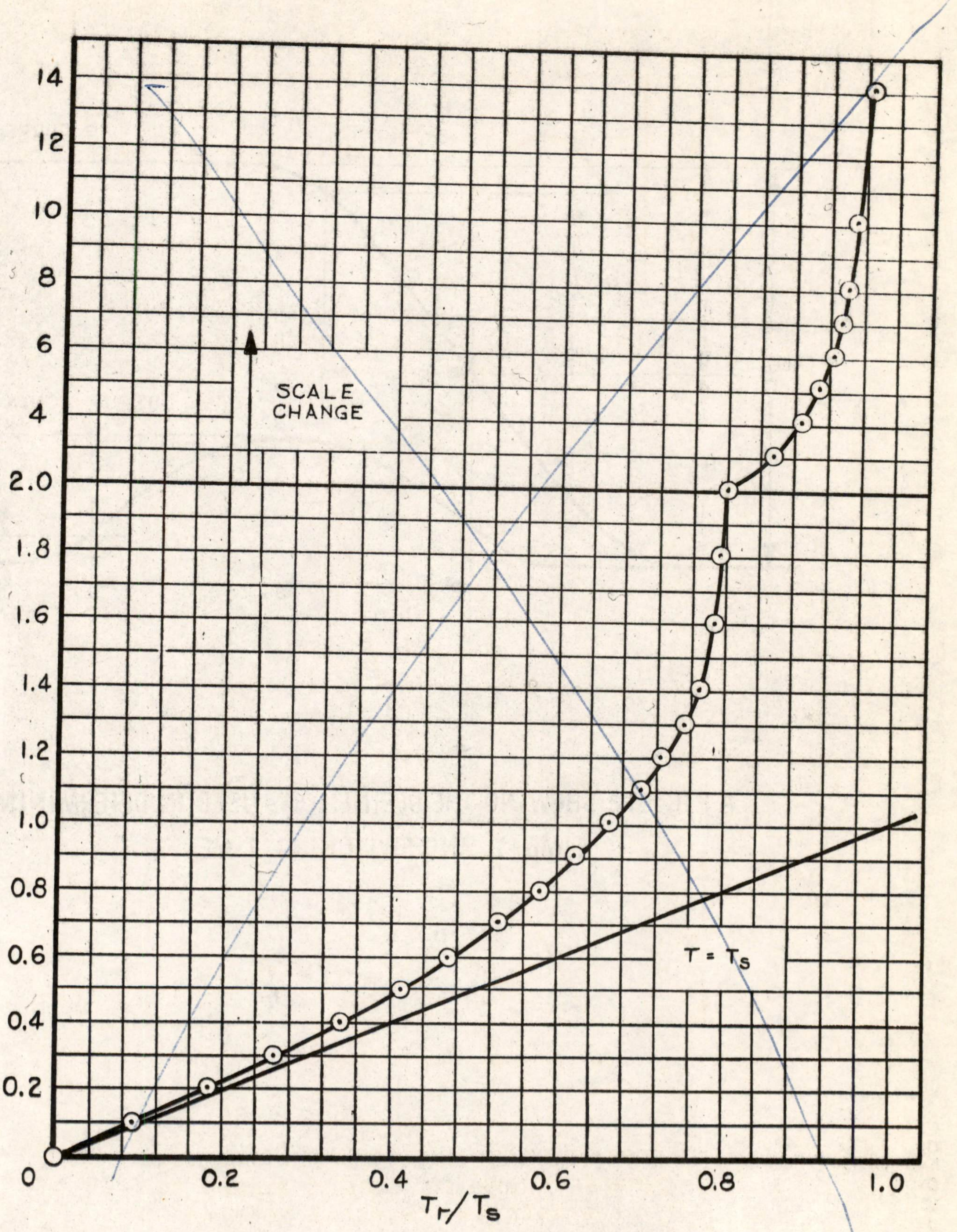


FIG. 4

PLOT OF T_r/T vs T_r/T_S AS DERIVED FROM THE EQUATION:

$$\cos\frac{\pi(T_s - T_r)}{T} = -\frac{\sin \pi T_r}{\frac{\pi T_r}{T}}$$

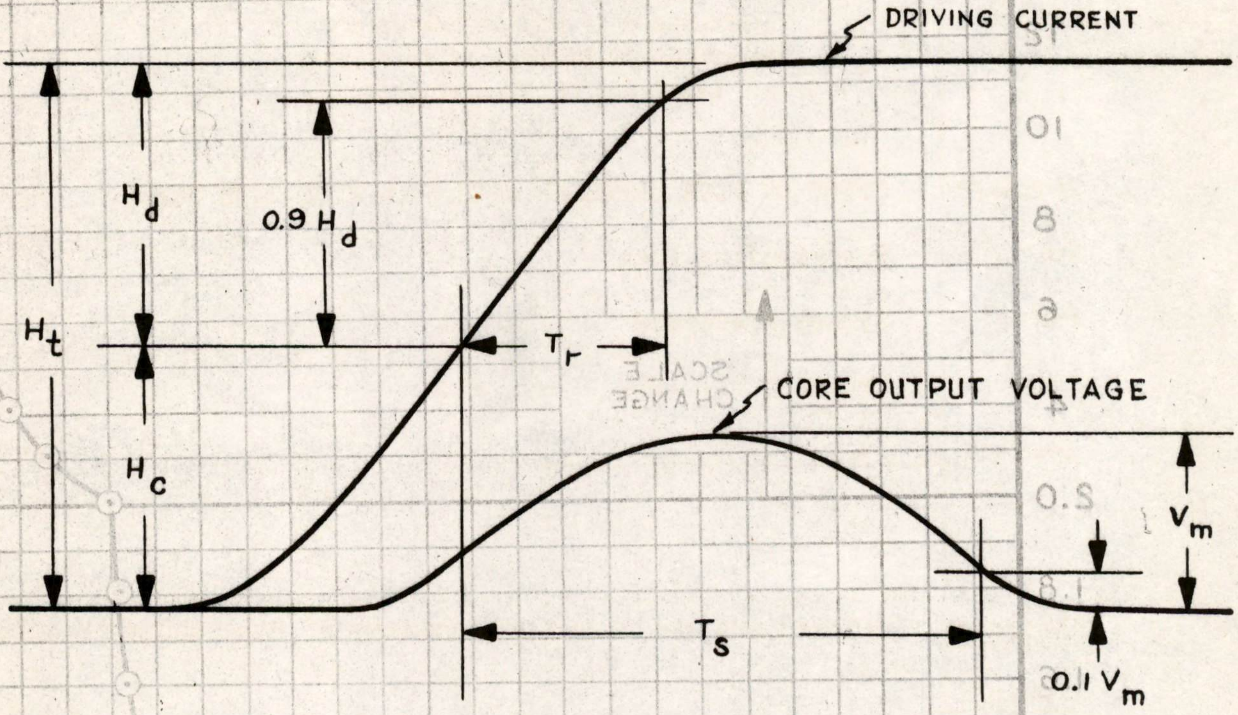


FIG. 5

A PICTURE SHOWING THE DEFINITIONS USED IN DETERMINING RISE TIME, T_r AND SWITCHING TIME, T_s

R Best B 077

Memorandum 6M-4874, Cl.

Sheet 1 of 4 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: SPEED OF FLUX REVERSAL WITH SLOW RISE-TIME DRIVE

To: Distribution List

From: Sydney Bradspies

Date: February 27, 1957

Approved: ______________________

Abstract: Enclosed are the corrected drawings of Memorandum 6M-4874.

Drawings:

Fig.4	A-69789-1
Fig.6	B-69791-1
Fig.7	A-69792-1
Fig.8	A-69793-1
Fig.9	A-69794-1

Sydney Bradspies

Sydney Bradspies
Group 63, Staff

Distribution List:

Taylor, N.H.
Brown, D.R.
Papian, W.N.
Mitchell, J.L.
Best, R.L.
Ellis, D.H.
Guditz, E.A.
Davidson, G.A.
Raffel, J
Bradspies, S.
Rising, H.K.
Smith, D.
Goodenough, J.
Zopatti, R.
Menyuk, N.
Canty, W.J.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

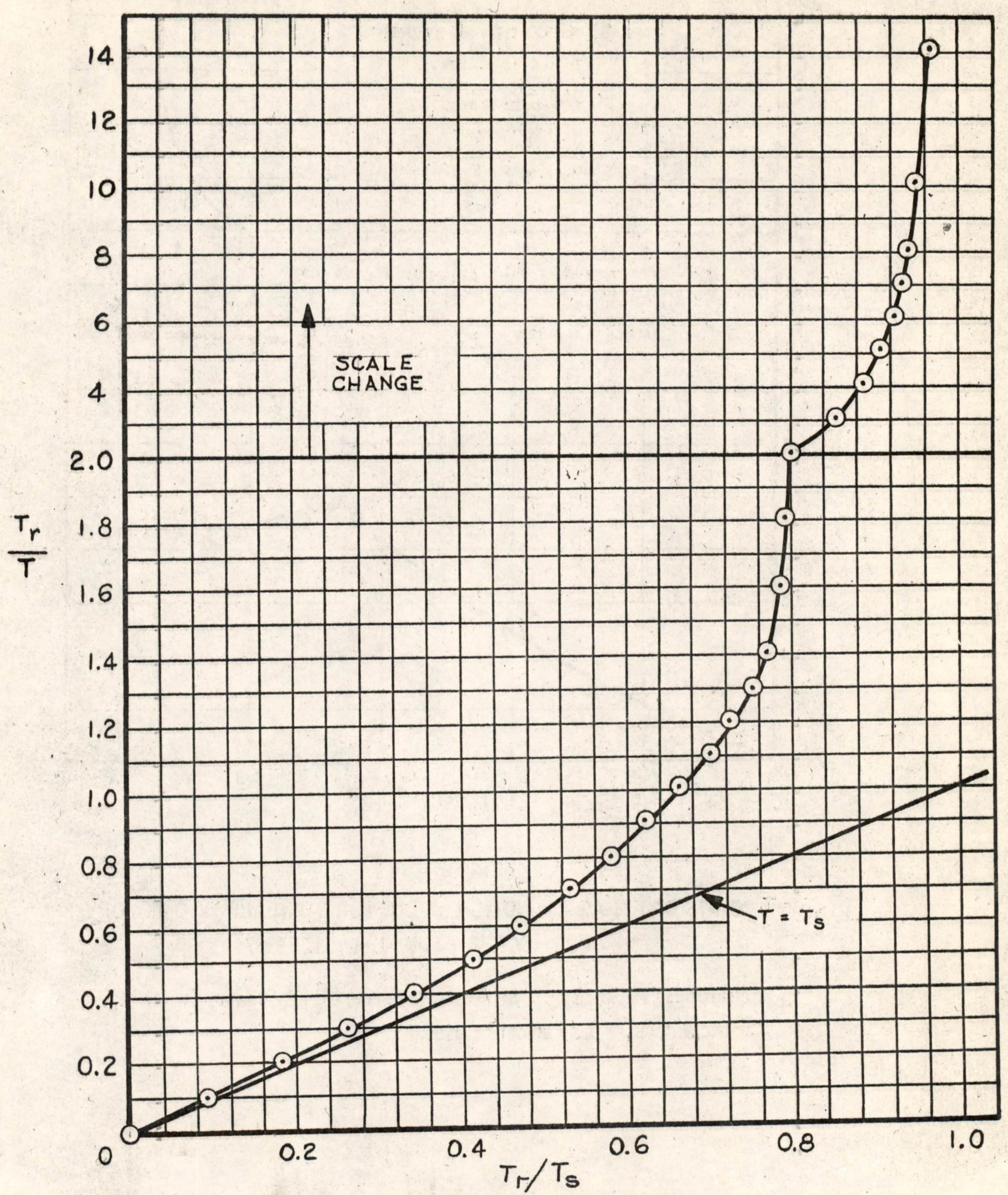


FIG. 4

PLOT OF T_r/T vs T_r/T_s AS DERIVED FROM THE EQUATION:

$$\cos \frac{\pi (T_s - T_r)}{T} = - \frac{\sin \frac{\pi T_r}{T}}{\frac{\pi T_r}{T}}$$

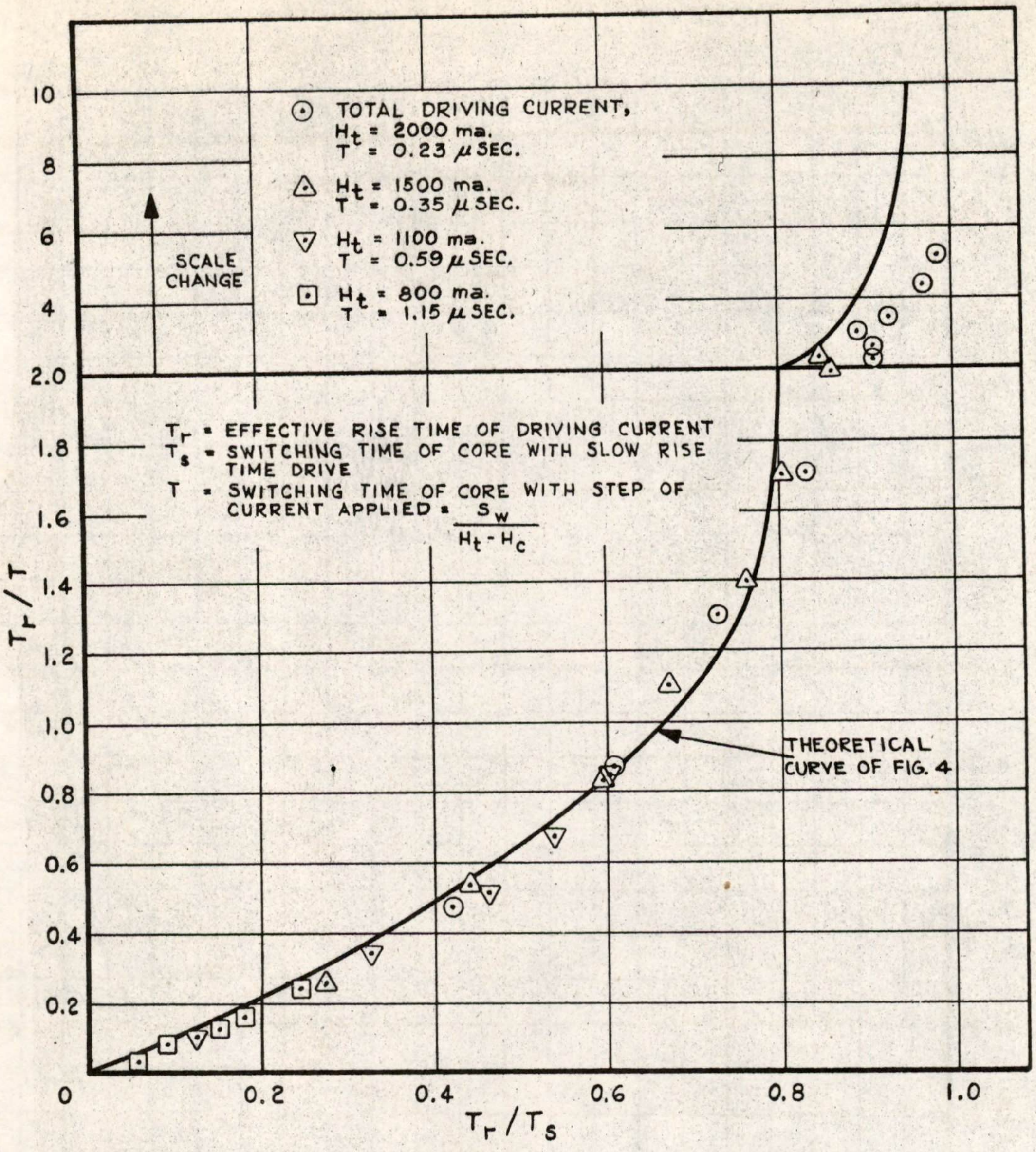


FIG. 6

POINTS OF T_r/T AS A FUNCTION OF T_r/T_s FOR STANDARD TX-0 MEMORY CORE - DCL -2 - 854 HU-1, SIZE F397

Squareness Ratio, R_s = 0.84; Coercive Force, $H_c \cong$ 500 ma; Switching Coefficient, S_w = 0.35 ampere - μsec.

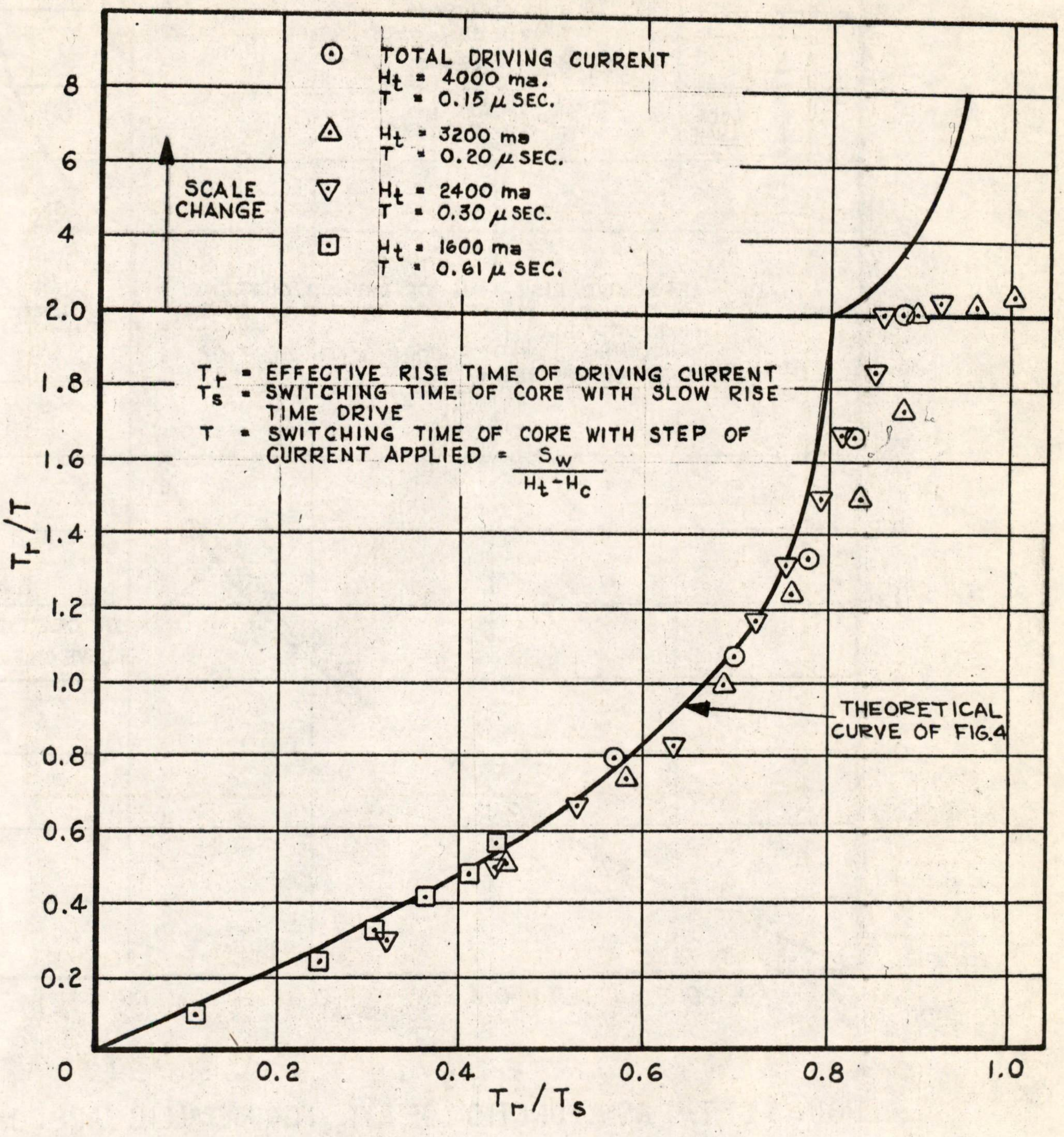


FIG. 7

POINTS OF T_r/T AS A FUNCTION OF T_r/T_s FOR FERRITE CORE NUMBER DCL-3-75-1, SIZE F-394

Squareness Ratio, R_s=0.80; Coercive Force, $H_c \cong$ 800 ma; Switching Coefficient, Sw = 0.49 ampere - μsec.

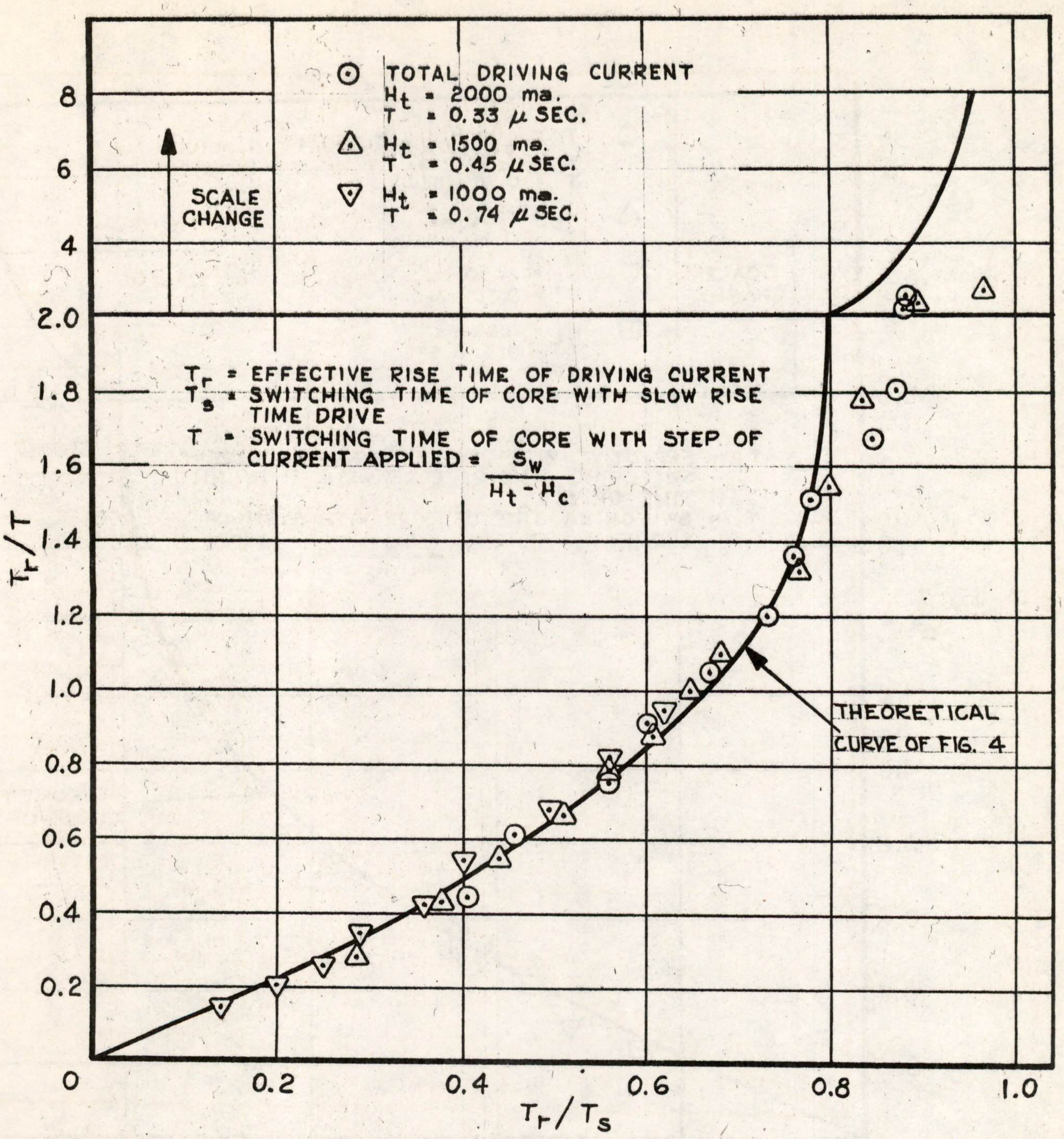


FIG. 8

POINTS OF T_r/T AS A FUNCTION OF T_r/T_s FOR METALLIC TAPE CORE MADE OF 10 WRAPS OF MO. PERM - 1/8 mil. x 3/4 " diameter x 1/8 " WIDE

Squareness Ratio, R_s = 0.86; Coercive Force, H_c ≅ 200 ma; Switching Coefficient, Sw = 0.588 ampere - μsec.

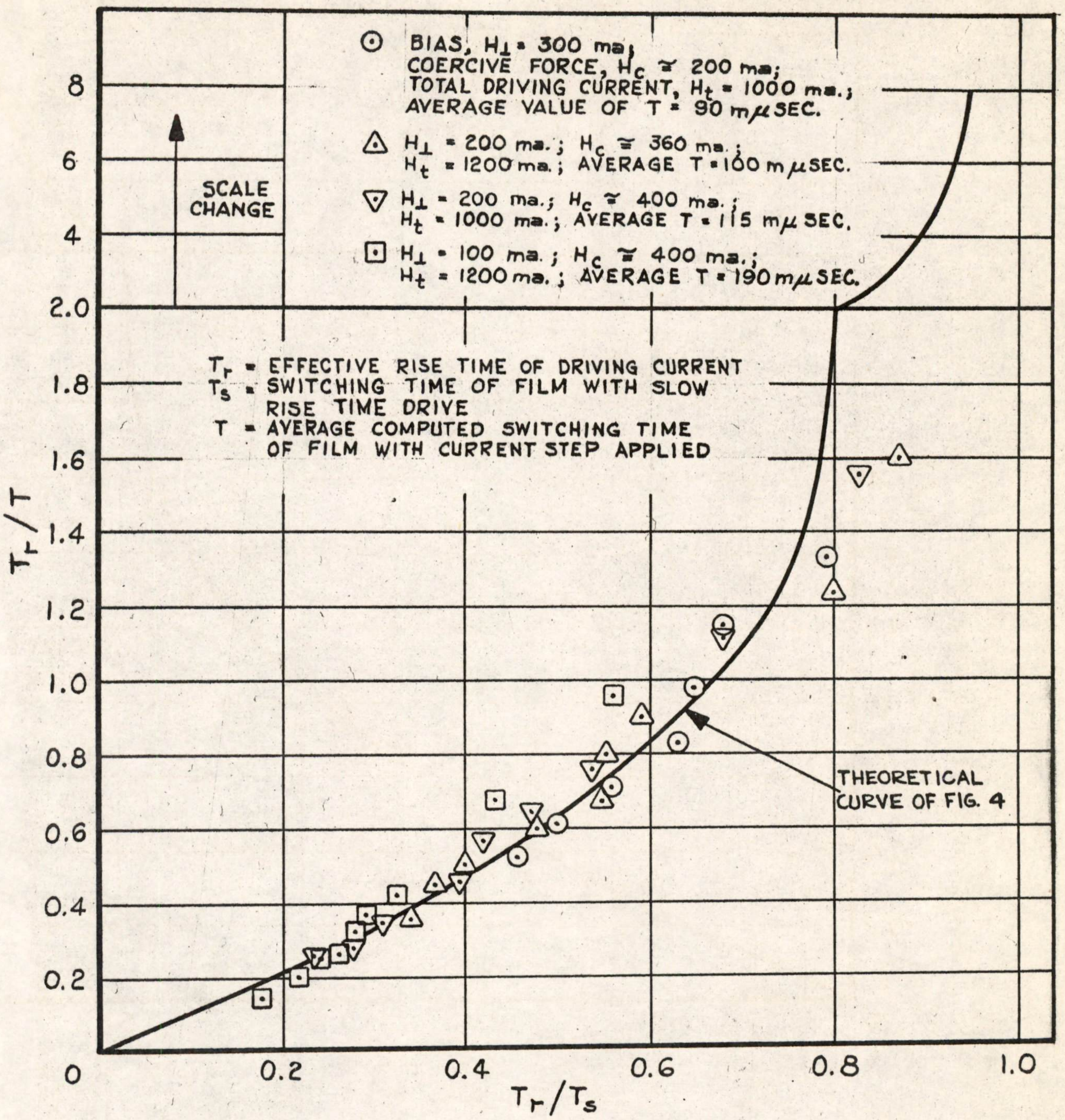


FIG. 9

POINTS OF T_r/T AS A FUNCTION OF T_r/T_s FOR THIN MAGNETIC FILM NUMBER 1B.

Composition: 80% Nickle, 20% Iron; Thickness: 1-2000Å; Substrate Temperature: 200°C; Depositing Field: 25 Gauss.

B 063 K. H. Olsen

Memorandum 6M-4878

Sheet 1 of 7 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: SURVEY OF PHOTO ELECTRIC ELEMENTS

To: Distribution List

From: J.L. Downing

Date: January 23, 1957

Approved: K Olsen
K.H. Olsen

Abstract: Currently available photoelectric elements (gas and vacuum photo tubes, multiplier phototubes, photoconductors, photodiodes, phototransistors, and photo voltaic cells) are listed with their ratings and spectral response. A chart showing their spectrum ranges and a list of manufacturers are included. Supplements listing sensitivity, frequency response, and dark currents are to be issued later.

Distribution List:

Group 63, Staff
Roger Tancrell, Group 24
Ted Clough, Group 65

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

INTRODUCTION

The purpose of this survey is to list all currently available commercial photoelectric elements and a few basic characteristics to aid the design engineer in selecting suitable units for photoelectric devices.

The elements are classified as follows:

A. Photo Emissive Elements: Those whose output depending entirely upon either the primary or secondary emission of electrons from a photo cathode. Including:

1. Gas Photo tubes: Gas filled tubes, with high sensitivity, slight non-linearity.
2. Vacuum photo tubes: linear with excellent dynamic response at high frequency.
3. Multiplier Phototubes: Secondary emission devices with large current amplification.

B. Photo Voltaic Cells: Units having a potential difference between their terminals when excited by light.

C. Photo Conductive Elements: Units whose conductance changes under illumimation. Including:

1. Photo Conductors.
2. Photo Diodes and photo transistors.

LIST OF PHOTOELECTRIC ELEMENTS

A. Photoemissive Elements
1. Phototubes - gas

Type	Spectral Response (1)	MAXIMUM RATINGS anode volts	 cathode µa (2)	 amb. t °C	Manufacturer	Remarks
1P29	S-3	100	5	100	RCA, GE	
1P37	S-4	100	5	75	RCA, GE	
1P40	S-1	90	3	100	RCA, GE	Good in high humidity
1P41	S-1	90	1.5	100	RCA, GE	End on illumination
20CG	Blue(3)	90	5		Mullard	
30CG	Blue	90	3		Mullard	
52CG	Blue	90	3		Mullard	
55CG	Blue	90	2		Mullard	
58CG	Blue	90	1.5		Mullard	End on Illumination
90AC	Blue	90	2.0		Mullard	
90CG	Blue	90	2.0		Mullard	
868	S-1	100	5	100	RCA, GE	
918	S-1	90	5	100	RCA, GE	
920	S-1	90	2	100	RCA, GE	Twin unit
921	S-1	90	3	100	RCA, GE	Cartridge type
923	S-1	90	3	100	RCA, GE	Renewal use
924	S-1	90	1.5	100	RCA	renewal use
927	S-1	90	2	100	RCA, GE	
928	S-1	90	3	100	RCA	non-directional
930	S-1	90	3	100	RCA, GE	
5581	S-4	100	3	75	RCA, GE	
5582	S-4	100	2	75	RCA	
5583	S-4	100	2	75	RCA	
5584	S-4	100	2	75	RCA	
6405/1640	S-1	90	5	100	RCA	Low microphonics

(1) See spectrum chart, page 10.
(2) Average cathode current may be doubled when anode supply voltage is limited to 80% maximum rated voltage.
(3) Where peak response falls.

2. Phototubes - vacuum

Type	Spectral Response (1)	MAXIMUM RATINGS anode volts	cathode μa	amb °C	Manufacturer	Remarks
1P39	S-4	250	5	75	RCA, GE	High humidity
1P42	S-9	180	0.4	75	RCA	End on Illumination
22	S-1	500		100	GE	
441	S-4	250		50	GE	
917	S-1	500	10	100	RCA, GE	Low leakage
919	S-1	500	10	100	RCA, GE	low leakage
922	S-1	500	5	100	RCA, GE	cartridge type
925	S-1	250	5	100	RCA	Short bulb
926	S-3	500	5	100	RCA	cartridge
929	S-4	250	5	75	RCA, GE	
934	S-4	250	4	75	RCA	
935	S-5	250	10	75	RCA, GE	
20CV	Blue(2)	150	20		Mullard	
51CV	Blue	100	.5		Mullard	
58CV	Blue	100	3		Mullard	
90AV	Red	100	5		Mullard	
90CV	Blue	100	10		Mullard	
5652	S-4	250	4	75	RCA	
5653	S-4	250	5	75	RCA	
6570	S-α	500	100	RCA	low microphonics	

1. See spectrum chart, page 10.

2. Where peak response falls.

3. Multiplier Phototubes

	MAXIMUM RATINGS					
Type	Spectral response	anode volts	anode ma	Current amplif $x10^6$	Manufacturer	Remarks
1P21	S-4	1250	0.1	2	RCA, GE	9 stage - low levels
1P22	S-8	1250	1.0	.2	RCA	9 stage -
1P28	S-5	1250	0.5	1.25	RCA	9 stage - ultra v.
931-A	S-4	1250	1.0	.8	RCA, GE	9 stage
2020	S-11	1500	2.0	.5	RCA	10 stage - end on ill.
5819	S-11	1250	.75	.5	RCA	10 stage - end on
6199	S-11	1250	.75	.6	RCA	10 stage - end on
6217	S-10	1250	.75	.6	RCA	10 stage
6291	S-11	1800	5.0	2	DuM	10 stage - end on
6292	S-11	1800	5.0	2	DuM	10 stage - end on
6323	S-4	1250	0.1		RCA	9 stage
6328	S-4	1250	0.1		RCA	9 stage short
6342	S-11	1500	2.0	.6	RCA	10 stage end on
6363	S-11	1800	5.0	2	DuM	10 stage end on
6364	S-11	1800	5	2	DuM	10 stage end on
6365	S-11	1300	.5	.003	DuM	6 stage end on small
6372	S-11	1200	.75	.6	RCA	10 stage
6467	S-11	1800	5	2	DuM	10 stage end on
6472	S-4	1250	.10		RCA	9 stage short
6655	S-11	1250	12.5	.5	RCA	10 stage end on
6810	S-11	2300	.4	12.5	RCA	14 stage end on
6903	S-13	1250		.4	RCA	10 stage end on

B. Photo Voltaic Cells

1. Barrier Layer Selenium Photocells

	Output µa	Remarks
PV1	75 (A)	General Electric
PV2		Available in various sizes, mounted and unmounted
PV3		and hermetically sealed.
A15	750 (B)	International Rectifier
A10	500 (B)	Open circuit voltage approaches 0.4 volts in
A5	250 (B)	full sunlight.
B15	750 (B)	
B10	350 (B)	
B5	220 (B)	Some available mounted and hermetically
B2	75 (B)	sealed.
RR	88 (C)	Weston
YR	80 (C)	2 1/4" diameter cells. Available mounted and
GB	70 (C)	hermetically sealed.
BB	58 (C)	
YY	52 (C)	
YG	10 (C)	Units may be matched for linearity, output and
R	90 (C)	spectral response.
B	75 (C)	
Y	50 (C)	
R1	220 (D)	Vickers
R10	360 (D)	Available in various sized.
R100	750 (D)	R10, R100, and S100 units available mounted.
SS1	200 (D)	
S10	360 (D)	
S100		

A- at 20 foot candles illumination and 100 ohms external resistance.
B- at 100 foot candles illumination and 100 ohms external resistance.
C- at 20 foot candles illumination and 200 ohms external resistance.
D- at 100 foot candles illumination and 150 ohms external resistance.

B. Photo Voltaic cells (cont.)

SILICON SOLAR CELLS

	output volts (1)	output ma (1)	Remarks
			<u>Hoffman</u>
S1-A	.30	130	1.25 inch diameter
S2	.39	120	2.86 inch diameter
P100	.30	32	1.57 inch diameter
52C	.42	20	.125 x .5 inch.
120C	.40	50	.39 x .78 inch.

Selenium Solar Cells

			<u>International Rectifier</u>
1B2	.26	3.5	.72 x .44 inches
1B5	.26	10	1.44 x .64 inches
1B10	.26	17	1.69 x .88 inches
1B15	.26	30	1.69 x 1.69 inches
1B20	.26	37	2.0 x 2.0 inches
1B30	.26	100	3.25 x 3.25 inches
SB-16B10	4.0	18	4.8 x 8.1 solar battery
SB-8B15	4.0	30	4.8 x 8.1 solar battery

(1) At maximum power output - full sunlight.

C. Photoconductive Elements

1. Photo Conductors

MAXIMUM RATINGS

No.	Type	Peak Response A°(1)	Applied volts	Power mw	Temp. °C	Remarks
PC1	CdS	5200	300	50	100	Photo Crystal
PC2	CdSe	7200	300	50	60	Photo Crystal
PC3	CdS	5200	300	50	100	Photo Crystal
CL2	CdS	5200	300	50	70	Clairex
CL3	CdSe	7500	300	50	70	Clairex
FT400*		7000	50			(high sensitivity - fatigues)Schwarz
FT401*		7000	50			(lower sensitivity - no fatique)Schwarz
61SV	PbS	25000	250		60	Mullard
EXTRON*	PbS	22000				Eastman Kodak
6694-A	CdS	5000	150	30	70	RCA
1N189	Ge	15000		40	50	Transistor Prods.
11A	Ge	15000	15	50	50	Transistor Prods.
11B	Ge	15000	15	50	50	Transistor Prods.
	CdSe	7600				General Electric
CE-700's	PbS	10000			100	Continental Electric

*Available in various sizes, sensitivities, and resistances.)

2. Photo diodes and photo transistors

			MAXIMUM RATINGS			
	Type	Peak Response	App.	Power	Temp.	Remarks
		°A	volts	mw	°C	
DIODES						
1N188	GePN	15000	40	30	70	Transistor Products
5B	GePN	15000	50	100	50	Transistor Products
5C	GePN	15000	50	100	50	Transistor Products
1N77A	GePN	15000	50	20	50	Sylvania
TRANSISTORS						
10A	NPN	15000	15	100	50	(2Lead)Transistor Prod.
10B	NPN	15000	15	100	50	(2lead)Transistor Prod.
GT66	PNP	16000	-12	50		(3 lead)General Transistor
T1800	NPN	15000	20	65		(2 lead) Texas Instrument

JLD/md

John L. Downing
John L. Downing

Attms:

Appendix A - List of Photoelement Manufacturers

Appendix B - Drawing A-69103 Graph (Comparison of Special Characteristics)

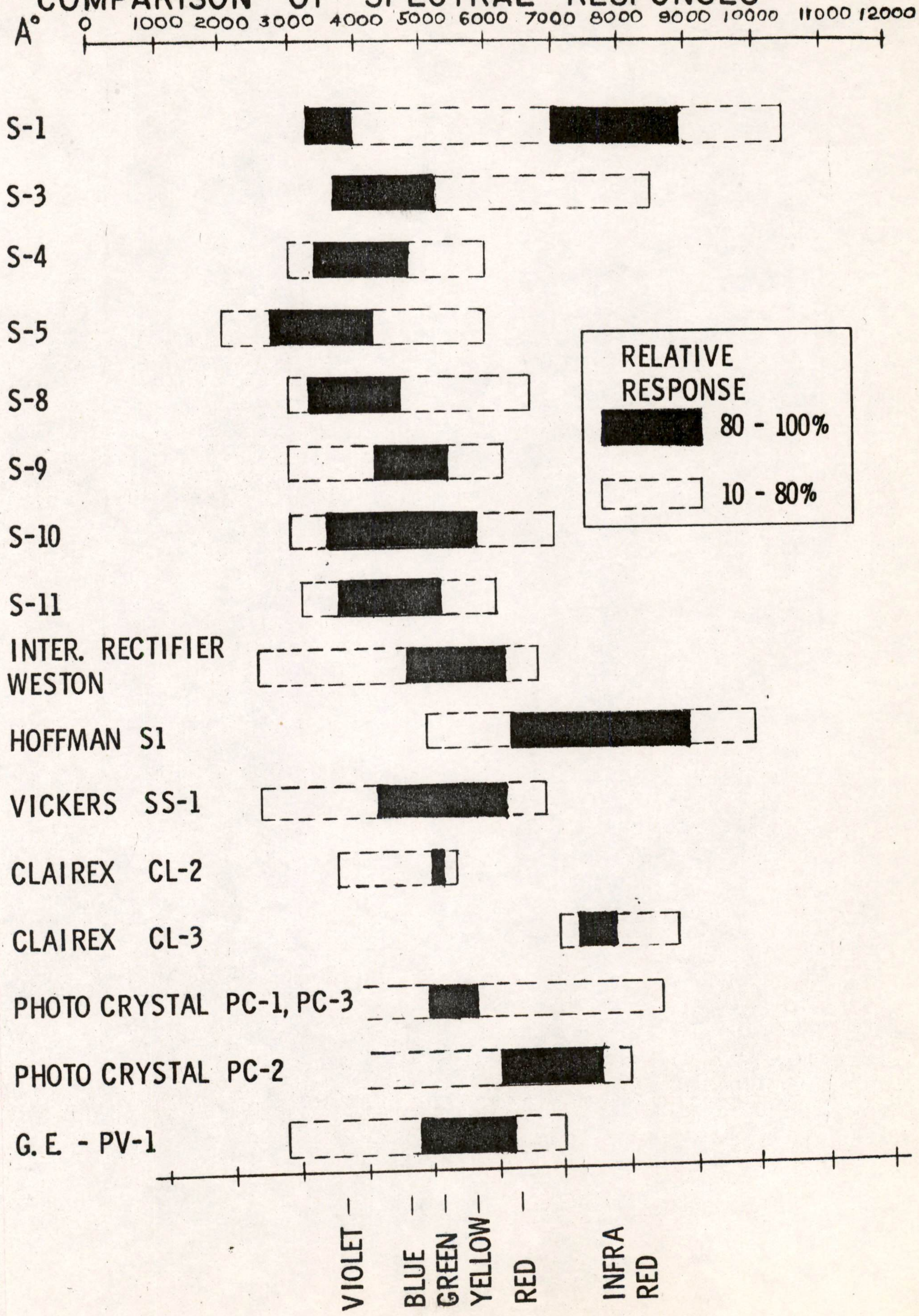
APPENDIX A

LIST OF PHOTOELEMENT MANUFACTURERS

1. Clairex Corporation,50 W., 26 St. N.Y. 10, N.Y.
2. Clerite Transistor Products, 241-257 Crescent St. Waltham 54, Mass.
3. Allen B. DuMont Laboratories Inc. 760 Bloomfield Ave. Clifton, N.J.
4. Eastman Kodak Co. 343 State St. Rochester 4, N.Y.
5. General Electric Co. 1 River Road, Schenectady 5, N.Y.
6. General Transistor Corp. 130-11 90th Ave. Richmand Hill 18, N.Y.
7. Hoffman Electronics Corp. 930 Pitner Ave. Evanston, Ill.
8. International Electronics Corp (Mullard Overseas Ltd.) 81 Spring St., N.Y.12, N.Y.
9. International Rectifier Corp. 1521 E. Grand Ave. El Sengundo, Calif.
10. Jarrell-Ash Co. (Hilger and Watts Ltd. "Schwarz") 26 Farwell St. Newtonville, Mass.
11. National Fabricated Products 2650 W. Belden St. Chicago 47, Ill.
12. Photo Crystals Inc. 15 S. First St., Geneva, Ill.
13. Radio Corporation of America, Camden, New Jersey
14. Texas Instruments Inc. 6000 Lemmon Ave. Dallas 9, Texas
15. Vickers Electric Division 1825 Locust St. St. Louis 3, Mo.
16. Weston Electrical Instrument Corp. 614 Frelinghuysen Ave., Newark 5, N.J.
17. Continental Electric Co., Geneva, Ill.
18. Sylvania Electric Products, Inc., Electronics Division, Woburn, Mass.

APPENDIX A

COMPARISON OF SPECTRAL RESPONSES



B 077 R. L. Best

Memorandum 6M-4913
Sheet 1 of 12 Sheets

Division 6 – Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: TRANSIENT RESPONSE OF JUNCTION TRANSISTORS--I

To: Donald J. Eckl

From: Ralph C. Johnston

Date: February 21, 1957

Approved: Donald J. Eckl
Donald J. Eckl

Abstract: Laplace transforms are used to solve the diffusion equation for hole flow in the base of a junction transistor. Current transfer functions are derived for a planar, homogeneous base transistor. In normalized form:

Common Base

$$I_c(\lambda) = I_e(\lambda) \frac{\gamma_N}{\operatorname{Cosh}\sqrt{\lambda - \frac{w^2}{L_p^2}}}$$

Common Emitter

$$I_c(\lambda) = I_b(\lambda) \frac{\gamma_N}{\operatorname{Cosh}\sqrt{\lambda - \frac{w^2}{L_p^2}} - \gamma_N}$$

Expressions are obtained for the inverses of these functions for steps of input current. In agreement with experimental results, a delay is found in the common base response. This leads to a modification of the rise and fall time equations of Ebers and Moll. The common emitter switching times are found to agree quite well with Ebers and Moll.

The storage times are found by breaking up the solution into its forward and reverse components and using the common base transfer function for each. This gives a storage coefficient involving ω_N, ω_I, β_N, β_I, γ_N, and γ_I which can be put approximately in the form of Ebers and Moll's result. The effect of γ_I on storage time is noted.

Distribution List:

Group 63 Staff

Baker, R.H. (Group 24)

Rediker, R.H. (Group 35)

Sawyer, D.E. (Group 35)

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

TRANSIENT RESPONSE OF JUNCTION TRANSISTORS--I

INTRODUCTION

The flow of minority carriers in a semiconductor is governed by the same diffusion equation that describes heat flow except that a minority carrier can recombine with a majority carrier. In this paper we shall follow the conventional practice of considering the minority carriers to be holes, and the transistor to be pnp. The equations for electrons can be obtained by replacing p by n and q by -q. The continuity equation[1]

$$-\frac{1}{q}\frac{dI_p(x,t)}{dx} - \frac{p(x,t)}{\tau_p} = \frac{dp(x,t)}{dt} \qquad \left[\frac{1}{cm^3 sec}\right] \tag{1}$$

states that in a slab of unit area and width dx, the number of holes entering minus those leaving per second, minus the number recombining per second equals the rate of growth of the hole density. The transport equation states that the number of holes passing through the slab in a second is D_p, the diffusion constant, times the slope of the hole density.

$$\frac{I_p(x,t)}{q} = -D_p\frac{dp(x,t)}{dx} \qquad \left[\frac{1}{cm^2 sec}\right] \tag{2}$$

We now differentiate (2), substitute this in (1) and obtain the diffusion equation.

$$D_p\frac{d^2p(x,t)}{dx^2} - \frac{p(x,t)}{\tau_p} = \frac{dp(x,t)}{dt} \tag{3}$$

This is a one-dimensional diffusion equation in which all hole flow is considered to be parallel and the emitter and collector to be planar and parallel. The assumption here is somewhat like neglecting fringing in a parallel plate capacitor. To solve this partial differential equation we take the Laplace transform with respect to time.

$$D_p \frac{d^2P(s,x)}{dx^2} - \frac{P(s,x)}{\tau_p} = s\,P(s,x) - p(x,0) \qquad (4)$$

To facilitate solution p(x,0) is set to zero, which means that the solution will start from rest and rise to the final value. Equation (4) may be put in the form:

$$\frac{d^2P(s,x)}{dx^2} - \Gamma^2 P(s,x) = 0 \qquad (5)$$

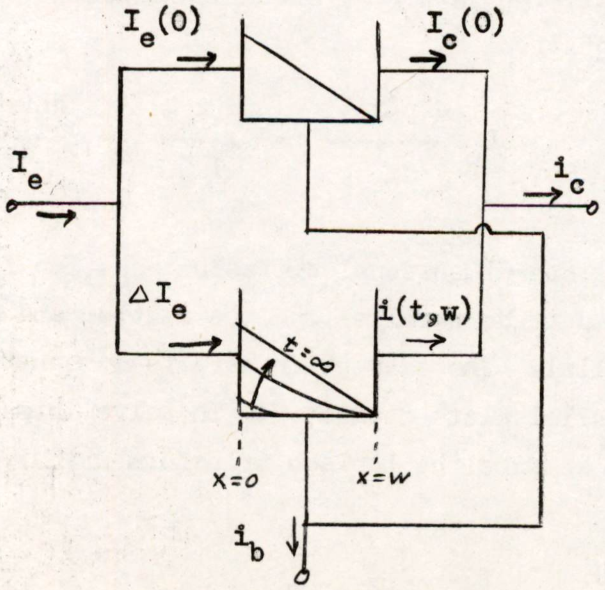
$$\text{where } \Gamma^2 = \frac{s + \frac{1}{\tau p}}{D_p}$$

with solution:

$$P(s,x) = A\,\text{Sinh}\,\Gamma x + B\,\text{Cosh}\,\Gamma x \qquad (6)$$

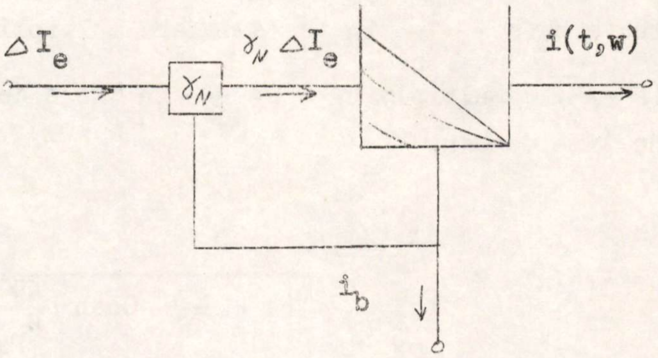
COMMON BASE ACTIVE SOLUTION

To make this transient solution fit physical problems we must add or subtract it from another solution which represents the initial conditions. This may be thought of as two transistors in parallel, one representing the initial conditions and the other the transient solution.



The hole density plot is a useful device for visualizing the transient response.[2] By (2) the current is proportional to the slope of the hole density. Equation (3) shows that in the steady state, any curvature is due to recombination.

Emitter efficiency, γ_N, is the ratio of the hole current at x = 0 to the current in the emitter lead. It is a function of the emitter hole density but will be assumed to be constant in this analysis.



The above diagram represents the effect of the emitter efficiency. Only the fraction γ_N of the actual emitter current reaches the emitter of the ideal transistor. The vertical lines are electron current to make the device obey Kirchoff's current law.

In the active region the collector is reverse biased and the hole density there is zero. Thus the boundary conditions on (6) are

$$\begin{cases} P(s,w) = 0 \\ I(s,0) = \gamma_N \dfrac{\Delta I_e}{s} \end{cases} \qquad (7)$$

The constants A and B may be found giving

$$P(s,x) = \frac{\Delta I_e \, \gamma_N}{q \, D_p \, \Gamma \, s} \; \frac{\operatorname{Sinh} \Gamma (w-x)}{\operatorname{Cosh} \Gamma \, w} \qquad (8)$$

The collector current is obtained by applying (2) to (8) and setting x = w.

$$I(s,w) = \frac{\Delta I_e \, \gamma_N}{s} \; \frac{1}{\operatorname{Cosh} \Gamma w} \qquad (9)$$

The inversion of this Laplace transform is the solution for the collector current as a function of time. To do this we must expand (9) in a partial fraction expansion which is actually d one by finding the residue at each pole. To facilitate the solution two theorems from Laplace transform theory will be used; the shift of poles in the s plane, and the time scale change.[3]

Replacing s by $s - \frac{1}{\tau_p}$ in the s domain multiplies in the time domain by $\exp(-\frac{t}{\tau_p})$. Multiplying s by $\frac{D_p}{w^2}$ in the s domain multiplies t by $\frac{w^2}{D_p}$ in the time domain.

$$i(t,w) = e^{-\frac{t}{\tau_p}} \mathcal{L}_s^{-1} \frac{\Delta I_e \ \gamma_N}{(s - \frac{1}{\tau_p}) \ \text{Cosh}\sqrt{\frac{sw^2}{D_p}}}$$

$$\tilde{i}(T,w) = e^{-\frac{w^2}{L_p^2}T} \mathcal{L}_\lambda^{-1} \frac{\Delta I_e \ \gamma_N}{(\lambda - \frac{w^2}{L_p^2}) \ \text{Cosh}\sqrt{\lambda}} \qquad (10)$$

where $\lambda = \frac{w^2}{D_p} s$ $\qquad L_p^2 = D_p \tau_p$

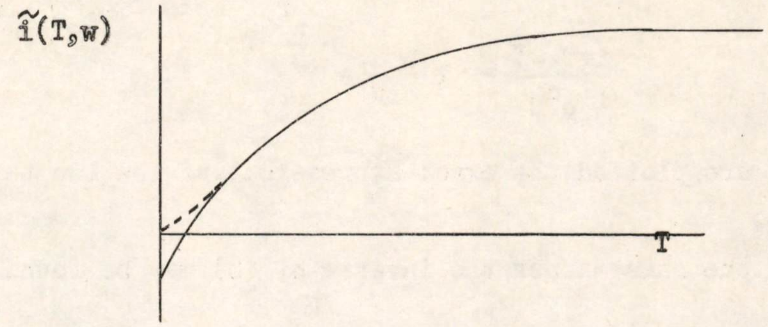
$T = \frac{D_p}{w^2} t$

This has poles at $\lambda = \frac{w^2}{L_p^2}$ and $\text{Cosh}\sqrt{\lambda} = \text{Cos}\sqrt{-\lambda} = 0$

or

$$\lambda = -(\frac{2n+1}{2})^2 \pi^2 \ ; \ \frac{w^2}{L_p^2} \qquad n = 0,1,\ldots \qquad (11)$$

Fig. 1b is a plot of the poles in the λ plane. To avoid appearing to have a pole in the right half plane, which would lead to a growing exponential, the poles have been shifted back by $s = \frac{1}{\tau_p}$ or $\lambda = \frac{w^2}{L_p^2}$. The residue at any pole is proportional to the product of the reciprocals of vectors drawn to that pole from all others. Because of the proximity of the pole at the origin to its neighbor, these two terms dominate all the rest. Note that the residue at the first pole from the origin is negative because one vector points in the negative direction, and is larger than the residue at the origin because the pole is closer to the outer poles. Neglecting all poles but the two, we might expect this:



Since (5) requires that the function start at zero the effect of the neglected poles is just to alter the response near zero as s hown by the dotted line.

The inversion of (10) is accomplished as follows:

$$\mathcal{L}_\lambda\, e^{+\frac{w^2}{L_p^2}T}\,\tilde{i}(T,w) = \frac{K_1}{\lambda - \frac{w^2}{L_p^2}} + \sum_{n=0}^{\infty}\frac{K_{2n}}{\lambda + (\frac{2n+1}{2})^2\pi^2} = \frac{\Delta I_e\,\gamma_N}{(\lambda - \frac{w^2}{L_p^2})\,\operatorname{Cosh}\sqrt{\lambda}} \tag{12}$$

Multiplying through by $\lambda - \frac{w^2}{L_p^2}$ and letting $\lambda \to \frac{w^2}{L_p^2}$ we obtain

$$K_1 = \frac{\Delta I_e\,\gamma_N}{\operatorname{Cosh}\frac{w}{L_p}} = \Delta I_e\,\gamma_N\,\beta_N = \Delta I_e\,\alpha_N \tag{13}$$

where β_N is the transport factor, not to be confused with the common emitter current gain. The N stands for normal as we later consider the inverted current gains. In a similar manner K_{2n} can be obtained. Thus

$$\frac{\tilde{i}(T,w)}{\Delta I_e\,\alpha_N} = 1 - \frac{1}{\beta}\frac{4}{\pi}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1) + \frac{4}{\pi^2}\frac{w^2}{L_p^2}\frac{1}{(2n+1)}}\; e^{-\left[(\frac{2n+1}{2})^2\pi^2 + \frac{w^2}{L_p^2}\right]T} \tag{14}$$

Let $\beta = 1$ and neglect all terms but two.

$$\frac{\tilde{i}(T,w)}{\Delta I_e \alpha_N} = 1 - \frac{4}{\pi} e^{-\frac{\pi^2}{4} T} \qquad (15)$$

In Fig. 2 are plotted the exact expression and the two term approximation for $\beta = 1$.

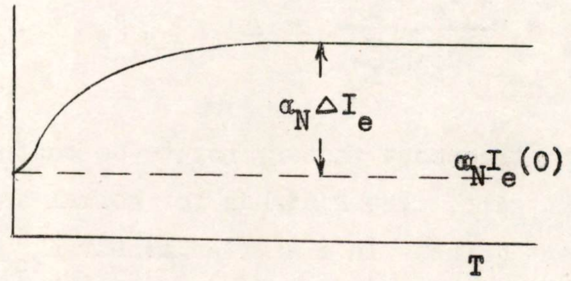
In the same manner the inverse of (8) may be found.

$$\frac{qD_p \tilde{P}(T,x)}{\alpha_N \Delta I_e} = \frac{\operatorname{Sinh} \frac{w}{L_p}(1 - \frac{x}{w})}{\frac{w}{L_p}}$$

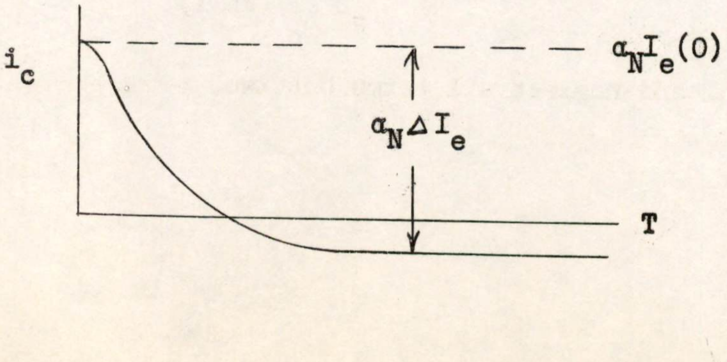
$$- \frac{1}{\beta} \frac{8}{\pi^2} \sum_{n=0}^{\infty} \frac{(-1)^n \operatorname{Sin} (\frac{2n+1}{2})\pi(1 - \frac{x}{w})}{(2n+1)^2 + \frac{4}{\pi^2} \frac{w^2}{L_p^2}} e^{-\left[(\frac{2n+1}{2})^2 \pi^2 + \frac{w^2}{L_p^2}\right] T} \qquad (16)$$

This is plotted in Fig. 3 for $\beta = 1$.

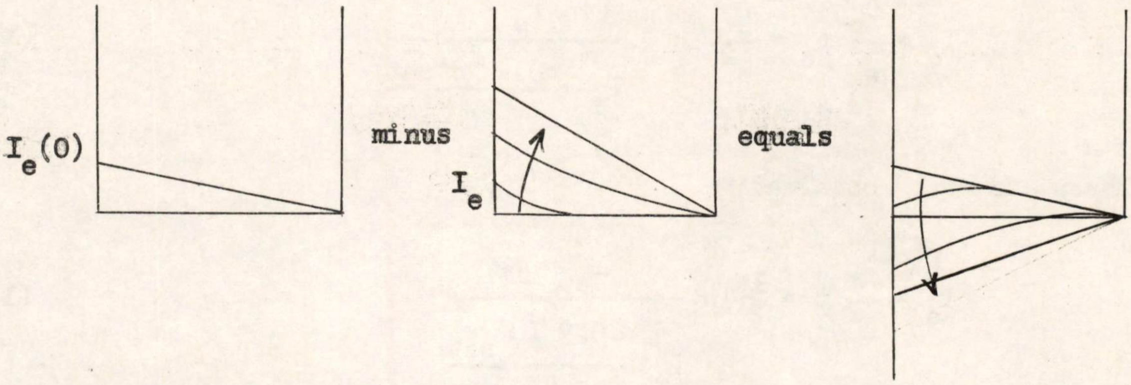
When the change of emitter current, ΔI_e, is a positive step, the transient collector current, $\tilde{i}(T,w)$, is added to $I_e(0)$ times the steady-state current gain α_N.



When ΔI_e is a negative step, $\tilde{i}(T,w)$ is subtracted from $\alpha_N I_e(0)$. If $\alpha_N \Delta I_e$ is larger than $\alpha_N I_e(0)$, as is usually the case during turn-off, the solution is valid only for positive currents as the emitter cannot emit backwards.

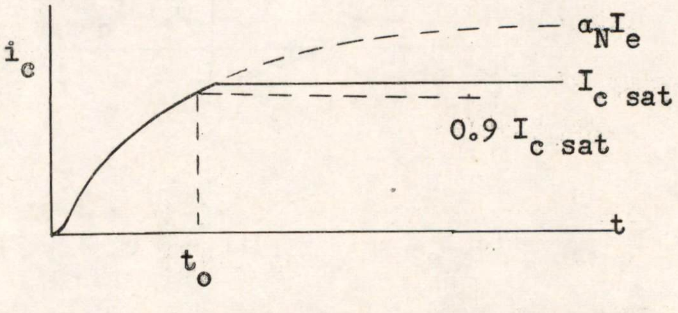


Actually the response near zero is an approximation because negative hole densities are assumed, an impossibility. Below are shown the hole density plots from which the above curve is obtained by using (2) at the collector.



While the slope at the collector is still negative, and collector current still flows, the hole density at the emitter is negative. This difficulty also arises in the derivation of switching times by Ebers and Moll[4]. The actual effect may be similar to this, however. When the emitter hole density reaches zero, an avalanche or punch through effect can take place which tends to move the emitter junction boundary toward the collector.

The rise time, t_o, is defined as the time for the collector current to rise from zero to 0.9 of its final value, $I_{c\ sat}$, which equals the collector supply voltage divided by the collector load resistance. This assumes zero emitter-collector voltage, a good approximation for supply voltages over one volt.



The rise time may be f ound by using the above f igure and (15) the two terma pproximation.

$$\alpha_N I_e \left(1 - \frac{4}{\pi} e^{-\frac{\pi^2}{4} T}\right) = 0.9\ I_{c\ sat}$$

$$T_o = \frac{D_p}{w^2} t_o = \frac{4}{\pi^2} \ln \frac{4}{\pi} \frac{I_e}{I_e - \frac{0.9\ I_{c\ sat}}{\alpha_N}} \qquad (17)$$

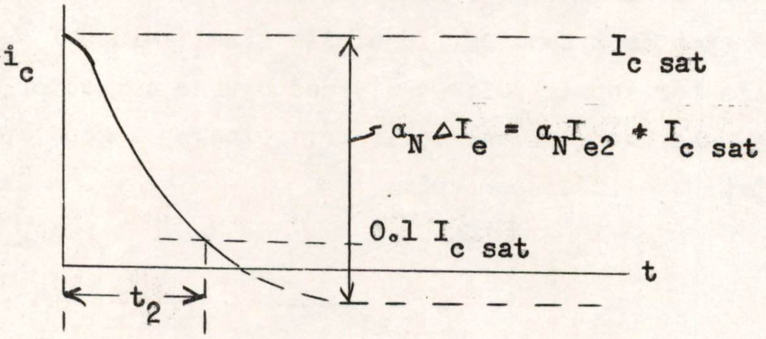
Ebers and Moll[4] obtained

$$T_o = \frac{D_p}{w^2} t = \frac{1}{2} \ln \frac{I_e}{I_e - \frac{0.9\ I_{c\ sat}}{\alpha_N}} \qquad (18)$$

with $\omega_N = \frac{2\ D_p}{w^2}$ (19)

by using $(1 - e^{-2T})$ instead of $(1 - \frac{4}{\pi} e^{-\frac{\pi^2}{4} T})$. This results from assuming that the hole density plot (Fig. 3) consists of straight lines.[5] These curves are compared in Fig. 2. In (17) the effect of the delay is incorporated in the $\frac{4}{\pi}$ while (18) permits zero rise time for infinite I_e.

For the fall time the following curve is obtained.



Thus

$$I_{c\ sat} - (\alpha_N I_{e2} + I_{c\ sat})\left(1 - \frac{4}{\pi} e^{-\frac{\pi^2}{4} T_2}\right) = 0.1\ I_{c\ sat}$$

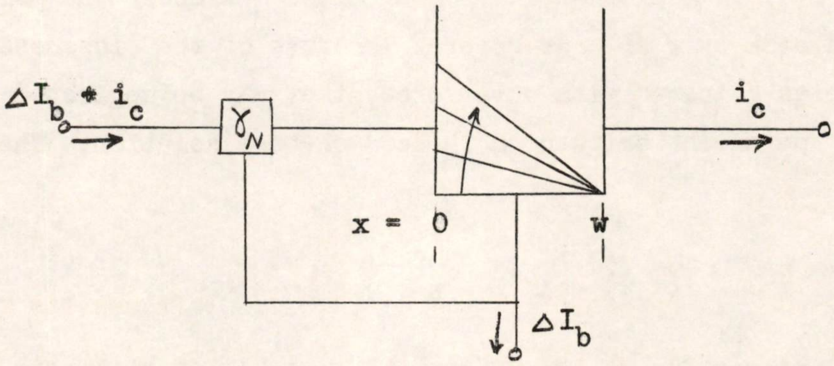
$$T_2 = \frac{D_p}{w^2} t_2 = \frac{4}{\pi^2} \ln \frac{4}{\pi} \frac{I_{c\ sat} + \alpha_N I_{e2}}{0.1\ I_{c\ sat} + \alpha_N I_{e2}} \qquad (20)$$

as compared with Ebers and Moll's relation,

$$T_2 = \frac{D_p}{w^2}\, t_2 = \frac{1}{2}\ln\frac{I_{c\ sat} + \alpha_N I_{e2}}{0.1\, I_{c\ sat} + \alpha_N I_{e2}} \tag{21}$$

COMMON EMITTER ACTIVE SOLUTION

The common emitter configuration is the usual one used because of its high current gain and phase inversion property. A step of base current, ΔI_b is applied.



The boundary consitions on (6) are now

$$\begin{cases} P(s,w) = 0 \\ I(s,0) = \gamma_N\left(\frac{\Delta I_b}{s} + I(s,w)\right) \end{cases} \tag{22}$$

with solution

$$P(s,x) = \frac{\Delta I_b \gamma_N}{q D_p \Gamma s}\, \frac{\text{Sinh}\,\Gamma(w-x)}{(\text{Cosh}\,\Gamma w - \gamma_N)} \tag{23}$$

and

$$I(s,w) = \frac{\Delta I_b \gamma_N}{s}\, \frac{1}{(\text{Cosh}\,\Gamma w - \gamma_N)} \tag{24}$$

After normalization and a shift of poles in the plane, (24) becomes

$$\tilde{i}(T,w) = e^{-\frac{w^2}{L_p^2}T}\, \mathcal{L}_\lambda^{-1}\, \frac{\Delta I_b\, \gamma_N}{(\lambda - \frac{w^2}{L_p^2})(\text{Cosh}\sqrt{\lambda} - \gamma_N)} \tag{25}$$

This has poles at $\lambda = \frac{w^2}{L_p^2}$ and at $\text{Cosh}\sqrt{\lambda} = \text{Cos}\sqrt{-\lambda} = \gamma_N$

$$\sqrt{-\lambda} = n2\pi \pm \text{Cos}^{-1}\gamma_N$$

Thus

$$\lambda = \frac{w^2}{L_p^2}, \quad -(n2\pi \pm \text{Cos}^{-1}\gamma_N)^2 \qquad n = 0,1,2,\cdots \tag{26}$$

Fig. 1(b) is a plot of the poles in the λ plane. The poles have been shifted back by w^2/L_p^2 as before. Because of the closeness of the first two poles compared with outer ones, they may be neglected to a much better approximation than in the common-base solution. The response is then

$$\tilde{i}(T,w) = K_1(1 - e^{\lambda_1 T}) \tag{27}$$

where K_1 is the residue at the origin and λ_1 is the location of the first pole from the origin.

$$\lambda_1 = -(\text{Cos}^{-1}\gamma_N)^2 - \frac{w^2}{L_p^2} \tag{28}$$

But by (13) and the power series for Cosh x, we have

$$\text{Cosh}\,\frac{w}{L_p} = \frac{1}{\beta} \cong 1 + \frac{w^2}{2L_p^2}$$

$$\frac{w^2}{L_p^2} \cong 2(\frac{1}{\beta} - 1) \tag{29}$$

Similarly

$$\text{Cos}\sqrt{-\lambda} = \gamma_N \cong 1 - \frac{(-\lambda)}{2}$$

$$-(\text{Cos}^{-1}\gamma_N)^2 \cong -2(1 - \gamma_N) \tag{30}$$

So (28) becomes

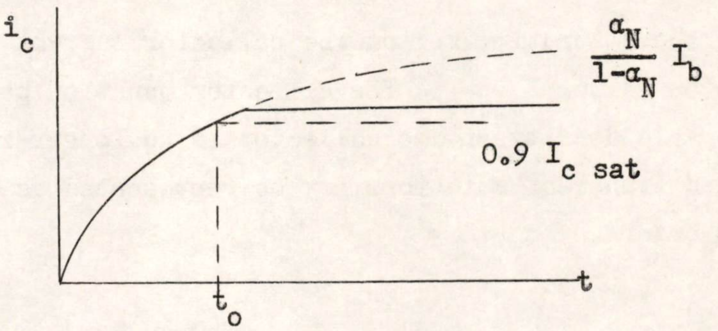
$$\lambda_1 \cong -2(1 - \gamma_N + \frac{1}{\beta} - 1) = -\frac{2}{\beta}(1 - \alpha_N) \tag{31}$$

and after neglecting the factor $\frac{1}{\beta}$ and using (19), the result of Ebers and Moll is obtained

$$\lambda_1 = -2(1-\alpha_N) = -\frac{w^2}{D_p}\,\omega(1-\alpha_N) \tag{32}$$

The validity of (27) may be checked by taking the inverse of (24) with $\gamma_N = 1$. The residues at the first two poles lead to a delay (T = .1 Fig. 2) of 1/12 which is very small with respect to the normalized time constant of the exponential of $\frac{1}{2(1-\alpha_N)}$.

The normalized rise and fall times may be obtained in a manner similar to the common base derivation using (27) instead of (15). Thus



$$T_o = \frac{D_p}{w^2}\,t_o = \frac{1}{2(1-\alpha_N)}\ \ln \frac{I_b}{I_b - 0.9\,\frac{1-\alpha_N}{\alpha_N}\,I_{c\ sat}} \tag{33}$$

and similarly

$$T_2 = \frac{D_p}{w^2}\,t_2 = \frac{1}{2(1-\alpha_N)}\ \ln \frac{I_{c\ sat} + \frac{\alpha_N}{1-\alpha_N}\,I_{b2}}{0.1\ I_{c\ sat} + \frac{\alpha_N}{1-\alpha_N}\,I_{b2}} \tag{34}$$

COMMON COLLECTOR ACTIVE SOLUTION

The transient emitter current is obtained by adding I_b to the transient collector current obtained in the common emitter solution.

$$\tilde{i}(T,w) = \Delta I_b\,\frac{\alpha_N}{1-\alpha_N}\,(1 - e^{-2(1-\alpha_N)T}) + \Delta I_b = \Delta I_b\,\frac{1}{1-\alpha_N}\,(1-\alpha_N e^{-2(1-\alpha_N)T}) \tag{35}$$

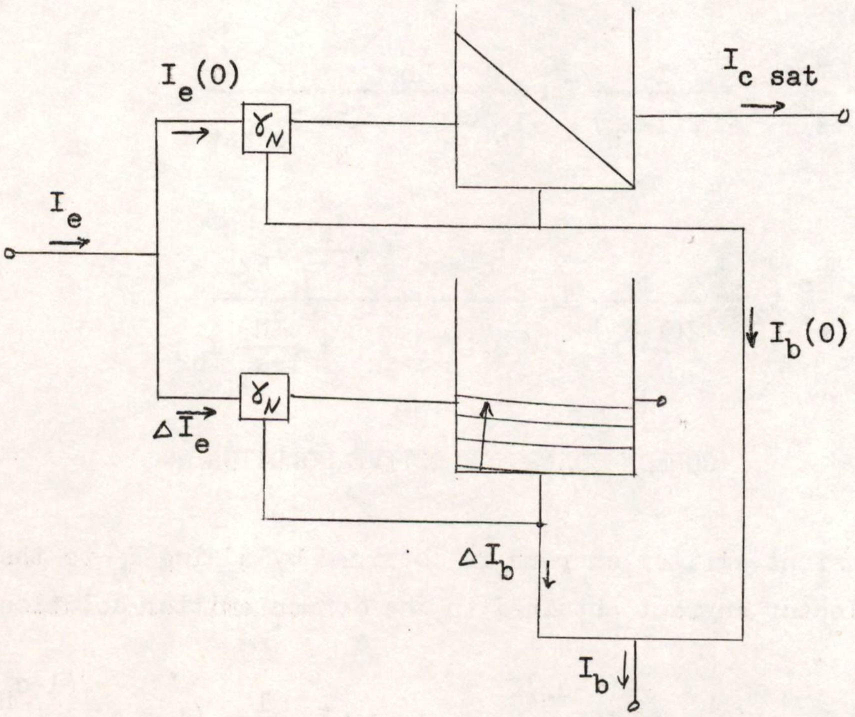
The rise and fall times may now be obtained in the same manner as before.

$$T_o = \frac{D_p}{w^2} t_o = \frac{1}{2(1-\alpha_N)} \ln \frac{\alpha_N I_b}{I_b - 0.9(1-\alpha_N) I_{e\ sat}} \qquad (36)$$

$$T_2 = \frac{D_p}{w^2} t_2 = \frac{1}{2(1-\alpha_N)} \ln \frac{\alpha_N (I_{e\ sat} + \frac{1}{1-\alpha_N} I_{b2})}{0.1\ I_{e\ sat} + \frac{1}{1-\alpha_N} I_{b2}} \qquad (37)$$

SATURATION SOLUTION

In the saturation region the collector current is limited by its saturation value, $I_{c\ sat}$. The collector junction becomes forward biased and the hole density at the collector is no longer zero. The steady-state and transient solutions may be represented as two parallel transistors as before.



It is seen from the above figure that $\Delta I_e = \Delta I_b$, therefore the solution is valid for all three circuit configurations. The boundary conditions on (6) are

$$\begin{cases} I(s,w) = 0 \\ I(s,0) = \dfrac{\Delta I_{b,e}\gamma_N}{s} \end{cases} \qquad (38)$$

with solution

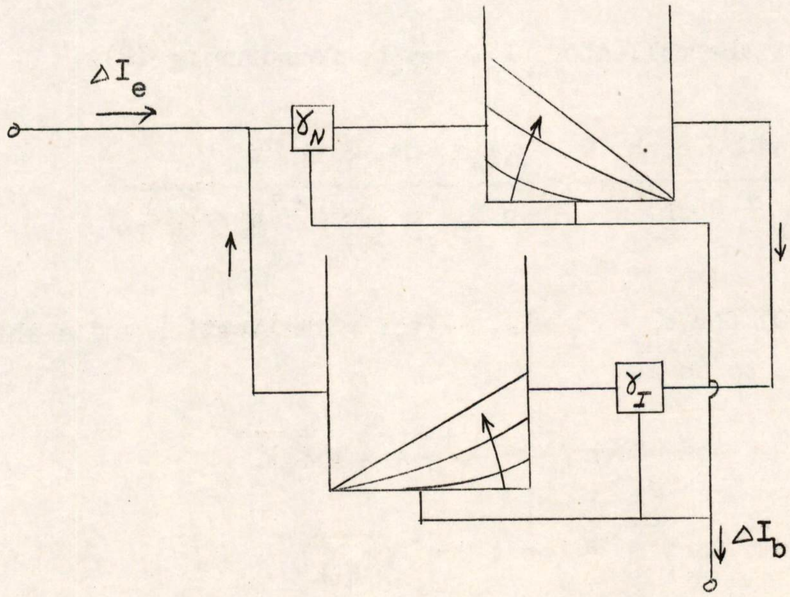
$$P(s,x) = \frac{\Delta I_{b,e}\gamma_N}{q\,D_p\,\Gamma\,s}\,\frac{\text{Cosh}\,\Gamma(w-x)}{\text{Sinh}\,\Gamma w} \qquad (39)$$

and

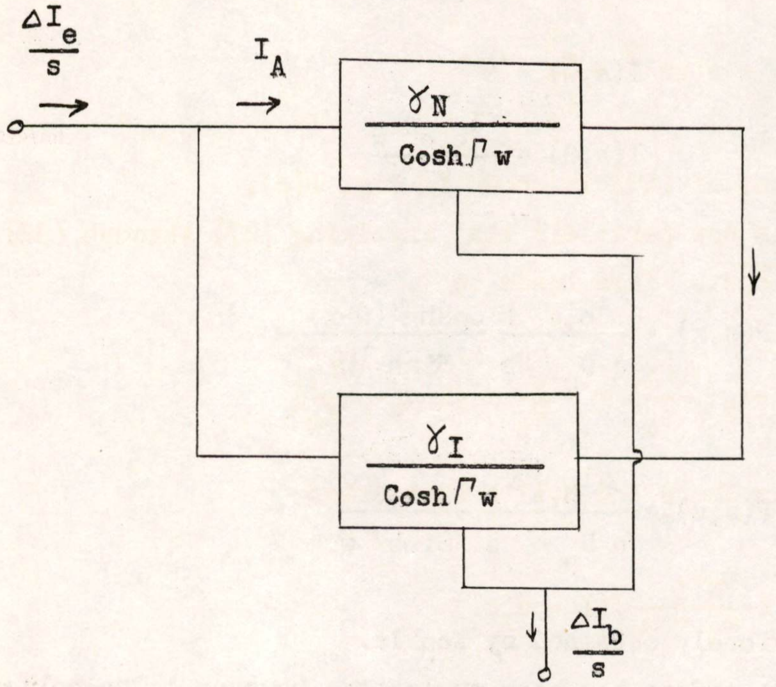
$$P(s,w) = \frac{\Delta I_{b,e}\gamma_N}{q\,D_p\,\Gamma\,s}\,\frac{1}{\text{Sinh}\,\Gamma w} \qquad (40)$$

as has been previously obtained by Konkle.[6]

An important effect has been neglected, however. The collector is forward biased and is emitting holes with efficiency γ_I. To take account of this, the transient solution is broken up into two active solutions in the manner of Ebers and Moll.



The active region hole density plots may be replaced by the transfer functions derived in the common base analysis.


$\frac{\Delta I_e}{s}$
I_A
$\frac{\gamma_N}{\text{Cosh}\,\Gamma w}$
I_B
$\frac{\gamma_I}{\text{Cosh}\,\Gamma w}$
$\frac{\Delta I_b}{s}$

$$I_A(s) = \frac{\Delta I_e}{s} + \frac{\gamma_N \gamma_I}{\text{Cosh}^2 \Gamma w} \, I_A(s)$$

$$I_B(s) = \frac{\gamma_N}{\text{Cosh}\,\Gamma w} \, I_A(s) = \frac{\Delta I_e \, \gamma_N \; \text{Cosh}\,\Gamma w}{s \, (\text{Cosh}^2 \Gamma w - \gamma_N \gamma_I)} \qquad (41)$$

The hole density at the collector, P_c, may be found using (8).

$$P_c(s) = \frac{I_B(s) \, \gamma_I \; \text{Sinh}\,\Gamma w}{q \, D_p \Gamma \; \text{Cosh}\,\Gamma w} = \frac{\Delta I_e \;\; \gamma_N \gamma_I \; \text{Sinh}\,\Gamma w}{q \, D_p \Gamma \, s \, (\text{Cosh}^2 \Gamma w - \gamma_N \gamma_I)} \qquad (42)$$

This reduces to (40) for $\gamma_N = \gamma_I = 1$. After normalization and a shift, the poles are found to be at

$$\lambda = \frac{w^2}{L_p^2} \qquad \text{and} \quad \text{Cosh}\sqrt{\lambda} = \text{Cos}\sqrt{-\lambda} = \pm\sqrt{\gamma_N \gamma_I} \,.$$

$$\text{or} \sqrt{-\lambda} = (n\pi \pm \text{Cos}^{-1}\sqrt{\gamma_N \gamma_I})$$

thus after a shift back, the poles are located at

$$\lambda = 0, \quad -(n\pi \pm \text{Cos}^{-1}\sqrt{\gamma_N \gamma_I})^2 - \frac{w^2}{L_p^2} \qquad n = 0,1,2,\cdots \qquad (43)$$

Similarly (42) has zeros at

$$\lambda = -(n\pi)^2 - \frac{w^2}{L_p^2} \qquad (44)$$

The pole-zero plot of (42) is shown in Fig. 1(c).

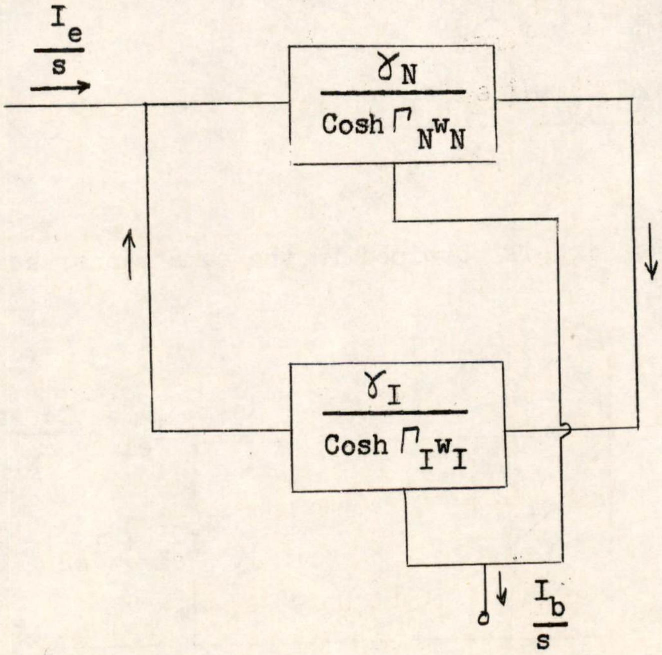
The analysis now parallels that involving (27) through (32) with γ_N replaced by $\sqrt{\gamma_N \gamma_I}$. This leads to

$$\lambda_1 = - \frac{2}{\beta}(1-\alpha_N\sqrt{\frac{\gamma_I}{\gamma_N}}) = - \frac{w^2}{D_p}\,\omega\,(1-\alpha_N\sqrt{\frac{\gamma_I}{\gamma_N}}) \qquad (45)$$

This differs from Ebers and Moll's expression

$$\lambda_1 = - \frac{w^2}{D_p}\,(\frac{\omega_N\omega_I}{\omega_N + \omega_I})(1 - \alpha_N\alpha_I) \qquad (46)$$

In order to take into account deviations from the one-dimensional planar transistor, they c onsidered the inverted transistor to have a different than the normal one. This can also be done by considering two one-dimensional transistors having different base widths.


$\frac{I_e}{s}$
$\frac{\gamma_N}{\text{Cosh}\,\Gamma_N w_N}$
$\frac{\gamma_I}{\text{Cosh}\,\Gamma_I w_I}$
$\frac{I_b}{s}$

Thus (42) becomes

$$P_c(s) = \frac{\Delta I_e\;\gamma_N\gamma_I\;\text{Sinh}\,\Gamma_I w_I}{q\,D_p\,\Gamma_I\,s\,(\text{Cosh}\,\Gamma_N w_N\;\text{Cosh}\,\Gamma_I w_I - \gamma_N\gamma_I)} \qquad (47)$$

It does not help here to shift the poles by $\frac{w^2}{L_p^2}$ because it is not equal in the inverted and normal directions. Equation (47) has poles at

$$\text{Cosh}\sqrt{(s_1 + \frac{1}{\tau_N})(\frac{w_N^2}{D_p})} \quad \text{Cosh}\sqrt{(s_1 + \frac{1}{\tau_I})(\frac{w_I^2}{D_p})} = \gamma_N \gamma_I \qquad (48)$$

Using (19), (29), and two terms of the power series for Cosh x, we obtain

$$s_1 = -(\frac{\omega_N \omega_I}{\omega_N + \omega_I})\left[1 - (1 - \frac{1-\beta_N}{\beta_N \gamma_N \gamma_I} - \frac{1-\beta_I}{\beta_I \gamma_N \gamma_I})\gamma_N \gamma_I\right]$$

$$\cong -(\frac{\omega_N \omega_I}{\omega_N + \omega_I})\left[1 - (1-[1-\beta_N])(1-[1-\beta_I])\gamma_N \gamma_I\right]$$

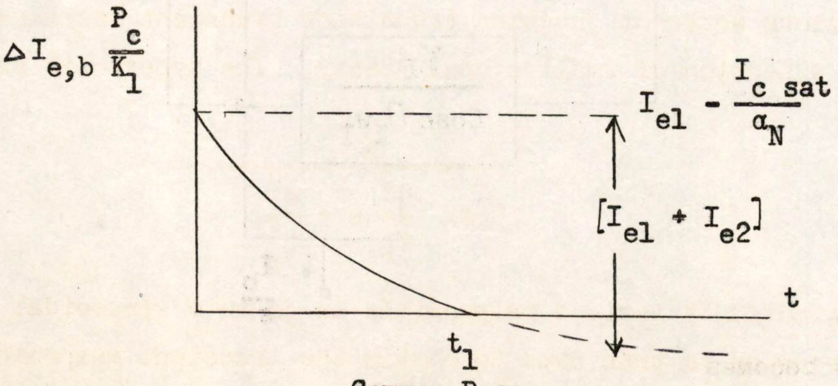
$$\cong -(\frac{\omega_N \omega_I}{\omega_N + \omega_I})(1 - \alpha_N \alpha_I) \qquad (49)$$

Although it does not enter into the expression for storage time, the value of collector hole density finally reached with input ΔI_e is obtained by finding the residue at the origin of (47).

$$P_c = K_1(1 - e^{s_1 t}) \qquad (50)$$

$$K_1 = \frac{\Delta I_{e,b}\, w}{q\, D_p} \quad \frac{\alpha_N \alpha_I}{1 - \alpha_N \alpha_I} \qquad (51)$$

The storage time, t_1, is obtained in the same manner as the fall times.



Common Base

$$t_1 = \frac{\omega_N + \omega_I}{\omega_N\omega_I(1-\alpha_N\alpha_I)} \ln \frac{I_{e2} + I_{e1}}{I_{e2} + \frac{I_{c\ sat}}{\alpha_N}} \qquad \text{(Common Base)} \qquad (52)$$

$$t_1 = \frac{\omega_N + \omega_I}{\omega_N\omega_I(1-\alpha_N\alpha_I)} \ln \frac{I_{b2} + I_{b1}}{I_{b2} + I_{c\ sat}(\frac{1-\alpha_N}{\alpha_N})} \qquad \text{(Common Emitter)} \qquad (53)$$

$$t_1 = \frac{\omega_N + \omega_I}{\omega_N\omega_I(1-\alpha_N\alpha_I)} \ln \frac{I_{b2} + I_{b1}}{I_{b2} + I_{e\ sat}(1-\alpha_N)} \qquad \text{(Common Collector)} \qquad (54)$$

If (45) were used instead of (47), the time constant term in the above equations would be

$$\frac{1}{\omega(1-\alpha_N\sqrt{\frac{\gamma_I}{\gamma_N}})} \qquad (55)$$

By making γ_I as small as possible, storage time is reduced without affecting the response in the active region. After assuming γ_N equals one, the factor $\frac{1}{1-\alpha_N\sqrt{\gamma_I}}$ is plotted vs γ_I in Fig. 4 with α_N as a parameter. The effect of γ_I on the storage time is quite apparent.

SUMMARY

Previous works on junction transistor transient response have treated it as an extension of small signal theory. The hyperbolic expression for β

$$\frac{1}{\text{Cosh}\ \frac{W^2}{D_p} s + \frac{W^2}{L_p^2}}$$

is approximated by various polynomials in $j\omega$ in a sinusoidal analysis. This expression is then used to obtain the transient response.

In this paper the hyperbolic functions are solved directly by Laplace transforms. This leads to a modification of Ebers and Moll's switching time equations for common base operation. The hyperbolic expressions with common emitter operation and in the storage region are a lso solved directly. The results agree quite well with Ebers and Moll. Other forms of the results include explicitly the emitter efficiencies.

Ralph C. Johnston

Ralph C. Johnston

RCJ

Attached Drawings: Fig. 1- A-48864-G
Fig. 2- A-48865-G
Fig. 3- A-48866-G
Fig. 4- A-48867-G

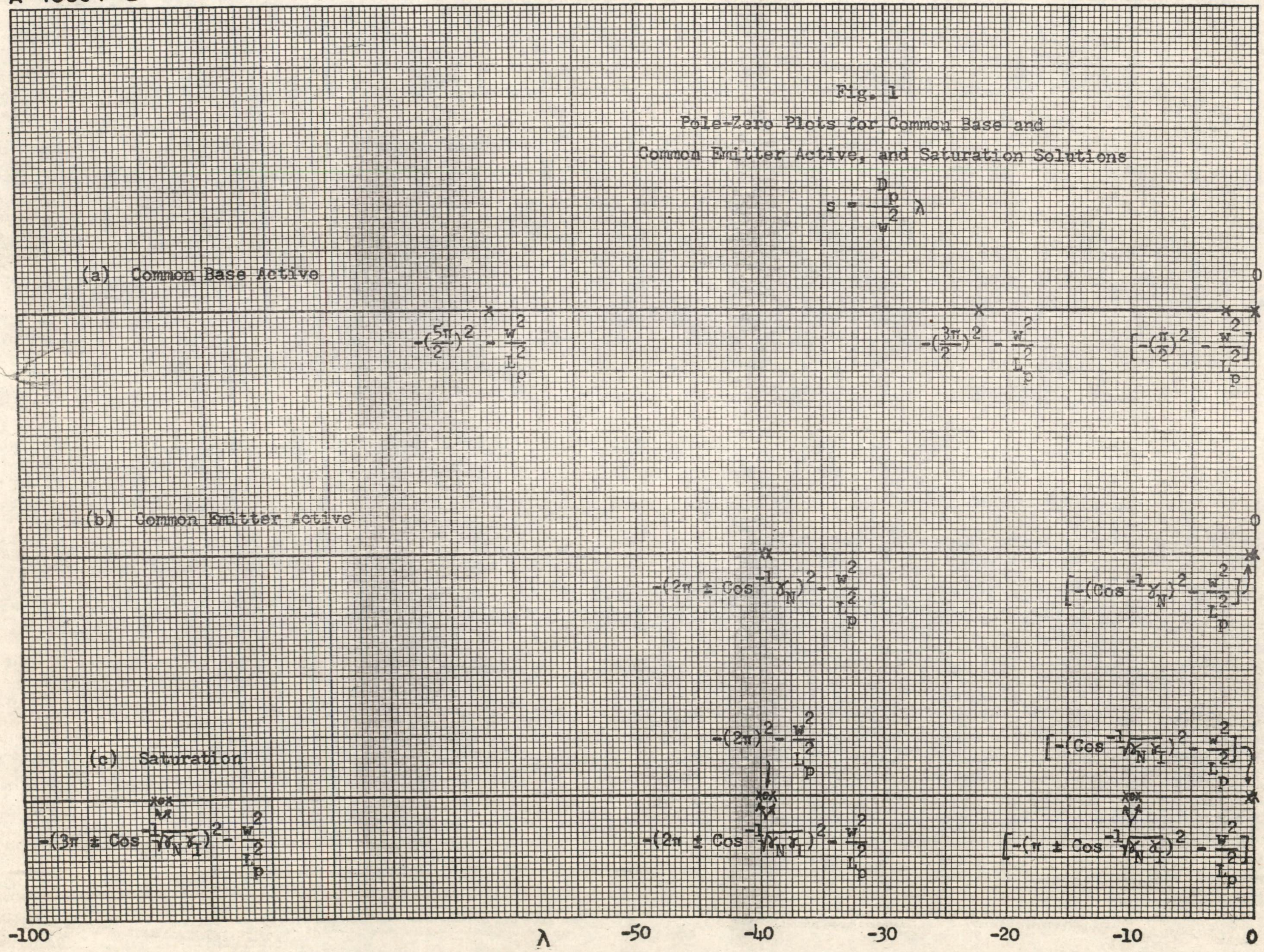
BIBLIOGRAPHY

1. Moll, J.L., "Junction Transistor Electronics," Proceedings of the IRE, December 1955
2. Kirk, C.T., Jr., "On the Behavior of Junction Transistors in Switching Circuits," Lincoln Laboratory Memorandum 6M-4780, January 17, 1957
3. Gardner, M.F., and Barnes, J.L., "Transients in Linear Systems," Wiley, New York, 1942, pages 245, 226
4. Ebers, J.J. and Moll, J.L., "Large-Signal Behavior of Junction Transistors," Proceedings of the IRE, December 1954
5. Kirk, C.T., Jr., "Transistor Lecture Notes", May 8, 1956
6. Konkle, K.H., "Hole Storage in a Saturated Grounded-Emitter Transistor Circuit," Master of Science Thesis, Massachusetts Institute of Technology, January 1957

Fig. 1

Pole-Zero Plots for Common Base and Common Emitter Active, and Saturation Solutions

$$s = \frac{D_p}{W^2}\lambda$$



A-48865-G

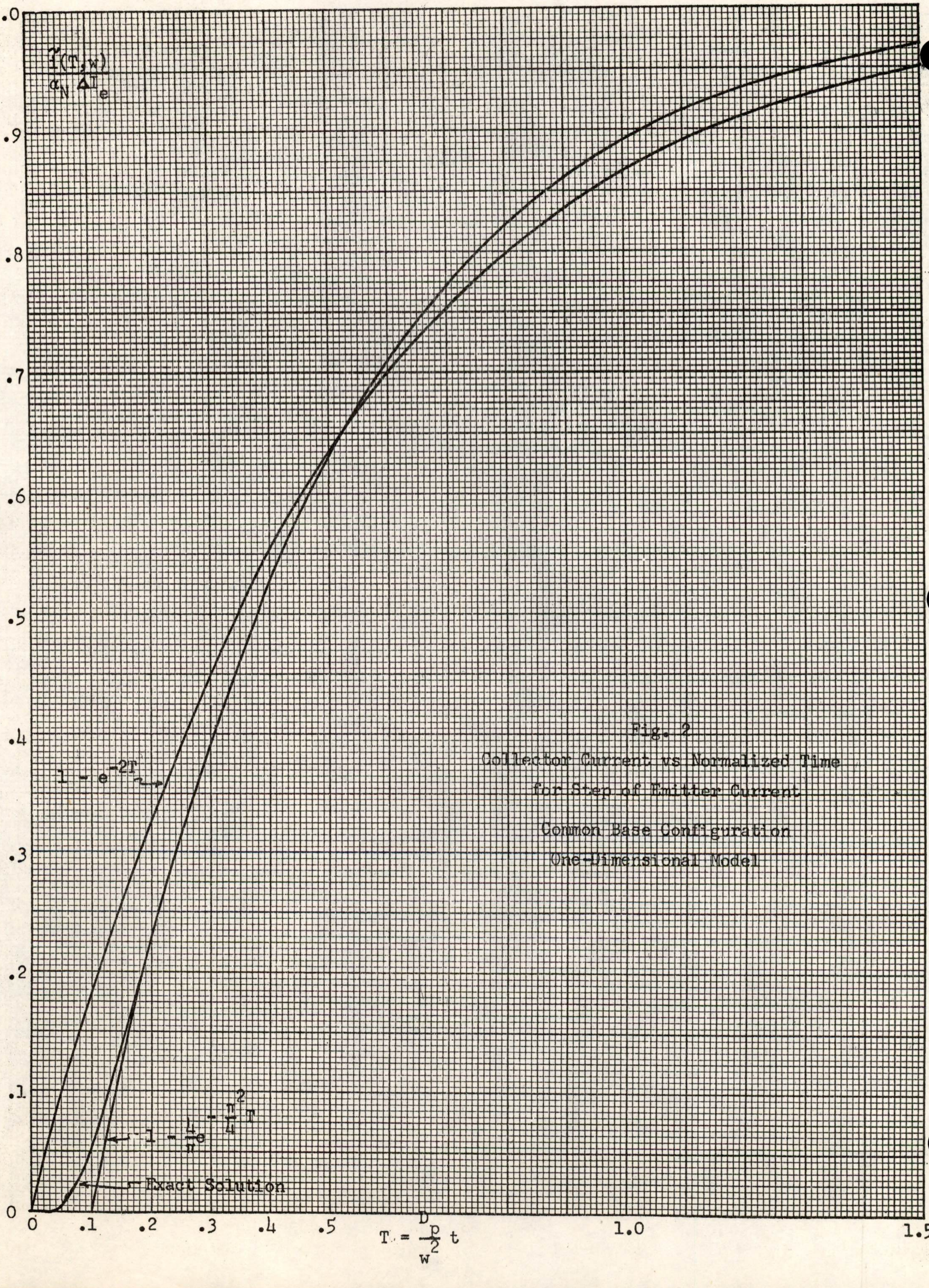


Fig. 2
Collector Current vs Normalized Time
for Step of Emitter Current
Common Base Configuration
One-Dimensional Model

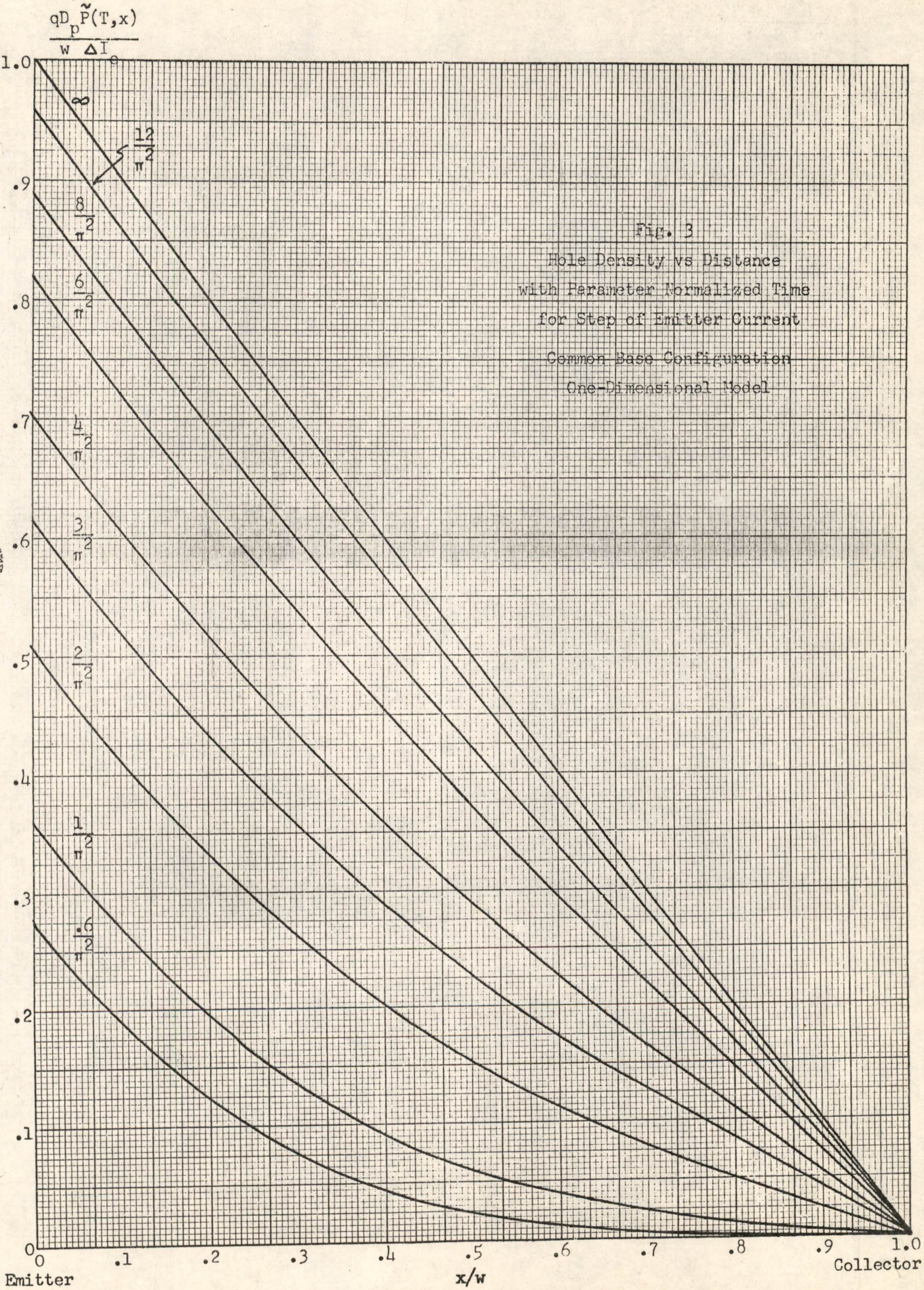


Fig. 3

Hole Density vs Distance
with Parameter Normalized Time
for Step of Emitter Current

Common Base Configuration
One-Dimensional Model

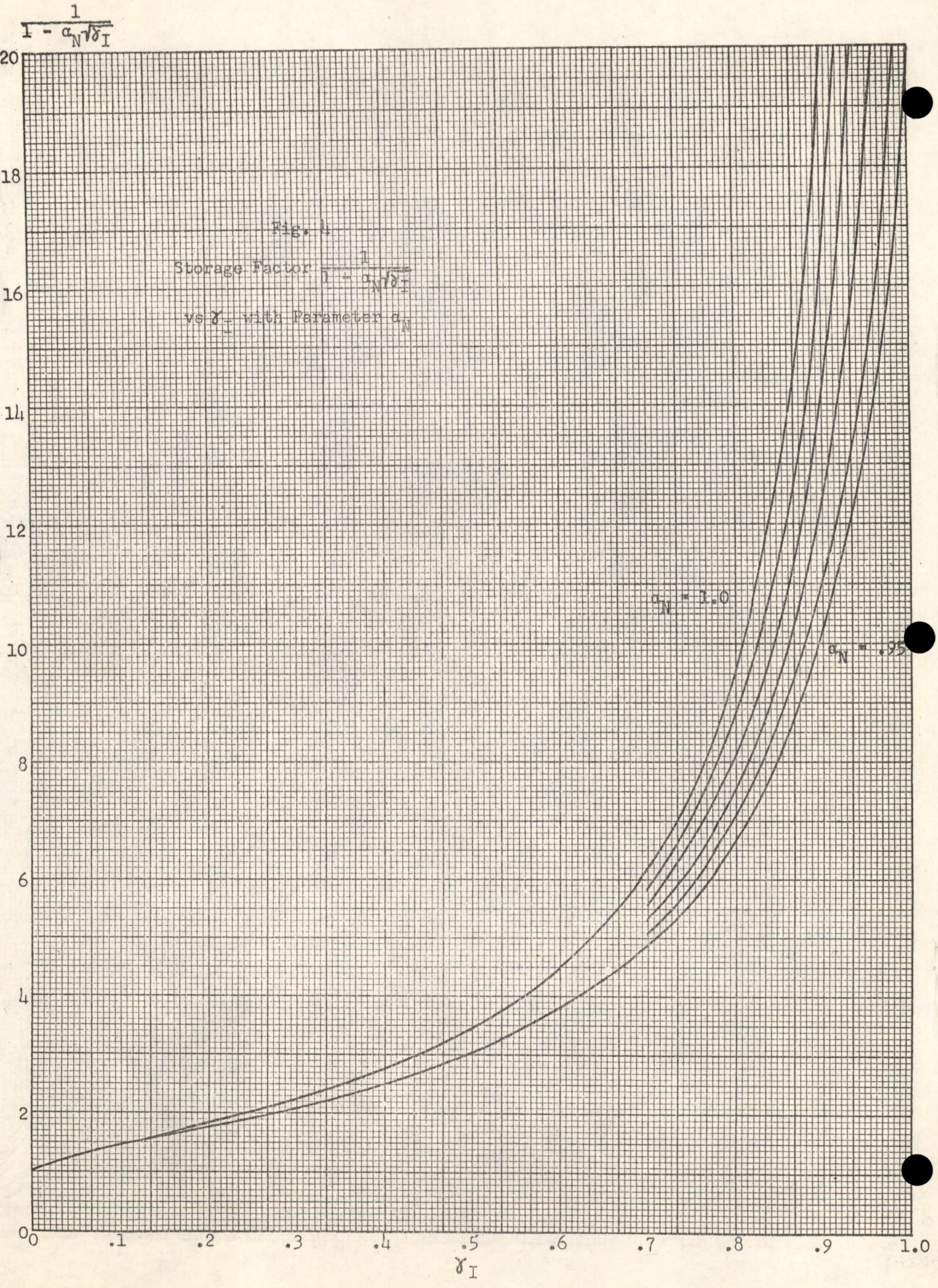


Fig. 4

Storage Factor $\frac{1}{1 - \alpha_N\sqrt{\gamma_I}}$

vs γ_I with Parameter α_N

R. L. Best

Memorandum 6M-4913, S-1 B063

Sheet 1 of 16 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: TRANSIENT RESPONSE OF JUNCTION TRANSISTORS -II

To: Donald J. Eckl

From: Ralph C. Johnston

Date: June 19, 1957

Approved: Donald J. Eckl
Donald J. Eckl

Abstract: The methods described in Part I are extended to the drift transistor. The current-step response is obtained in the common-base and common-emitter configurations for various values of built-in field. The sinusoidal behavior is also found for these conditions. The improvement due to the field is shown to depend on the circuit configuration and upon whether rise time or cutoff frequency is used as a reference. The improvement is less than that previously predicted.

The storage time for a one-dimensional model is found to be longer but this model seems to break down for the drift transistor.

Distribution list:

Baker, R. H.
Rediker, R. H.
Sawyer, D. E.
Neustadter, S. F.
Halpern, J.

Group 63 Staff

2. The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19 (122)-458.

INTRODUCTION

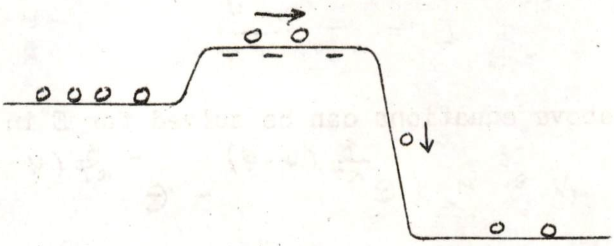
Part I of this paper (6M-4913) used Laplace transformation techniques to obtain the rise, storage, and fall times of homogeneous base transistors. Part II will use the same techniques as applied to the graded base or drift transistor. The order of treatment will be the same as that of Part I. However, first we shall consider the graded impurity density and the differences between the field produced and the constant field to be assumed.

THE DRIFT TRANSISTOR

For comparison, the operation of a homogeneous base transistor will be reviewed. Figure 5(a) shows a PNP transistor. In normal operation the emitter junction is forward biased and emits minority carriers (holes) into the n-type base. There they flow across by diffusion only and are collected at the reverse-biased collector junction. The flow is governed by the diffusion equation (3) as has been discussed.

Figure 5(c) represents the transistor on a potential energy basis, for no external voltages. The holes tend to flow to a lower potential and may be thought of as marbles on top of the potential line. The electrons tend to flow up because of their negative charge and may be represented as bubbles under the potential.

The effect of external voltages is to shift the potential levels. In normal operation the diagram becomes as shown below.



Those holes in the emitter having higher energies are able to surmount the barrier, diffuse across the base plateau and to be captured by the collector. Some of the electrons in the base get across the barrier into the emitter, but (in a good transistor) the number is small compared to the number of holes injected.

In the drift transistor, the potential in the base is tilted as shown in Figure 5(e). The "built-in" field forces a hole across the base to the collector with velocity

$$v = \mu_p E$$

where μ_p is the mobility and E the built-in field.

It is necessary to have a graded impurity distribution to produce this tilting as shown in Figure 5(d). It may be shown that

$$n = n_i \, e^{\frac{q}{kT}(\psi - \varphi)} \qquad (56)$$

and

$$p = n_i \, e^{-\frac{q}{kT}(\psi - \varphi)} \qquad (57)$$

where $n_i = 2.5 \times 10^{13}$ and $q/KT = 39$ at room temperature. ϕ is the Fermi level and ψ the position of the Fermi level in intrinsic material. For $(\psi - \varphi)$ positive, then $n > p$ and we have n type material. The requirement of charge neutrality results in

$$n - p = N_D - N_A = N \qquad (58)$$

where N is the net impurity density.
The field can be obtained from the slope of the potential

$$E = - \frac{d\psi}{dx} \qquad (59)$$

The four above equations can be solved for E in terms of N as follows:

$$N = n_i \left(e^{\frac{q}{kT}(\psi - \varphi)} - e^{-\frac{q}{kT}(\psi - \varphi)} \right)$$

$$\frac{dN}{dx} = \frac{q}{kT} n_i \left(e^{\frac{q}{kT}(\psi - \varphi)} + e^{-\frac{q}{kT}(\psi - \varphi)} \right) \frac{d\psi}{dx} = \frac{q}{kT}(n + p)\frac{d\psi}{dx}$$

$$E = - \frac{kT}{q} \frac{1}{n + p} \frac{dN}{dx} \qquad (60)$$

The impurity density distribution necessary to obtain a constant field can be obtained by solving (60). Assume that $p << n$.

$$E = -\frac{kT}{q}\,\frac{1}{N}\,\frac{dN}{dx} \tag{61}$$

$$\frac{dN}{N} = -\frac{q}{kT}\,E\,dx$$

$$N = N_0\,e^{-\frac{q}{kT}Ex} \tag{62}$$

The exponential distribution is obtained approximately by diffusion of impurities into an almost intrinsic semiconductor which may be either n or p type. Suppose we diffuse donors into p type. At some point ($N_D - N_A$) will be zero and the field according to (61) will be infinite. At that point, however, the material is intrinsic, p = n, and the approximation $p << n$ is not valid. In figure 6 are shown the curves of qE/kT vs. x for distributions of $10^{17}\,e^{-x} + 10^{14}$ and $10^{17}\,e^{-x} - 10^{14}$. The results obtained by using (61) are dotted while the exact solutions using (60) are solid.

In their paper entitled "The Dependence of Transistor Parameters on the Distribution of Base Layer Resistivity"[1], Moll and Ross assumed an impurity distribution of the form ($N_o\,e^{-x} - N_1$) for the exponential case, and used the approximate equation (61) in the solution. The extent that this affects the results is unknown.

The drop off in field is not as severe as depicted in Figure 6. Actually, the high resistivity portions of the base are likely to be in the space charge region of the collector junction. One might expect this region to extend into about x = 6 for reasonable collector voltages. In view of the difficulty in taking the exact field into account, and of calculating the extent of the collector junction, a constant field will be assumed. This assumption was also made by Krömer.[2]

BASIC EQUATIONS

The continuity equation for holes is unaffected by the field. However, it must be restated in terms of total instead of excess hole density.

[1] Moll, J.L., and Ross, J.M., "The Dependence of Transistor Parameters on Base Layer Resistivity," Proceedings of the I.R.E., vol. 44, No. 1, (January, 1956).

[2] Krömer, H., "Zur theorie des diffusions und des drifttransistors," Archiv Der Elektrischen Ubertragung, vol. 8, pp. 223-8, 363-369, 499-504, 1954.

$$-\frac{1}{q}\frac{dI_p(x,t)}{dx} - \frac{p(x,t)-p_o(x,t)}{\tau_p} = \frac{dp(x,t)}{dt} \tag{63}$$

In (1) we were able to subtract out p_o because it is constant in a homogeneous base transistor. There was an approximation then in (7a) which is actually

$$P(s,w) = -p_o \tag{64}$$

By neglecting p_o here we have in effect subtracted out I_{co} from the collector current.

The transport equation has a term due to the field as well as to the diffusion.

$$\frac{I_p(x,t)}{q} = -D_p\frac{dp(x,t)}{dx} + \mu_p E(x)\, p(x,t) \tag{65}$$

We now differentiate (65), substitute this in (63) and obtain the differential equation governing holes in the base.

$$D_p\frac{d^2p(x,t)}{dx^2} - \mu_p\left[E(x)\frac{dp(x,t)}{dx} + p(x,t)\frac{dE(x)}{dx}\right] - \frac{p(x,t)-p_o(x)}{\tau_p} = \frac{dp(x,t)}{dt} \tag{66}$$

The $dE(x)/dx$ term is zero since only a constant field will be considered. $p_o(x)$ is an exponential function of x and it is not possible to subtract it out as before. To simplify the solution, recombination will be neglected. The assumption is justifiable for two reasons. Equations (28)-(32) showed that the emitter efficiency, γ_N, and the transport factor β_N (which is due to volume recombination) may be combined into one term with little error. Also, in practical transistors volume recombination can be neglected beside surface recombination and emitter efficiency. With these two assumptions (66) becomes

$$\frac{d^2p(x,t)}{dx^2} - \left(\frac{\mu_p}{D_p}E\right)\frac{dp(x,t)}{dx} = \frac{1}{D_p}\frac{dp(x,t)}{dt} \tag{67}$$

The Laplace transformation with respect to time gives

$$\frac{d^2P(x,s)}{dx^2} - \left(\frac{2}{f}\right)\frac{dP(x,s)}{dx} = \frac{1}{D_p}\left[s\,P(x,s) - p(x,0)\right] \tag{68}$$

where 2/f is the notation of Krömer for $\frac{\mu_p}{D_p}E$. Note that f has the dimension of length. We let p(x,o) be zero to facilitate solution as before. The resulting second order differential equation can be solved with a substitution $p = e^{mx}$.

$$m = \frac{1}{f} \pm \sqrt{\frac{1}{f^2} + \frac{s}{D_p}} = \frac{1}{f} \pm \Gamma \qquad (69)$$

$$P(x,s) = e^{x/f}\left(A \operatorname{Cosh} \Gamma x + B \operatorname{Sinh} \Gamma x \right) \qquad (70)$$

The constants A and B may be found with the grounded-base, grounded-emitter, and storage boundary conditions and the inverse transforms obtained. Before we do this it is appropriate to consider the expression (w/f) which will appear as a parameter in the solutions.

$$\frac{w}{f} = \frac{\mu_p}{2D_p} E w = \frac{q}{2kT} E w = \frac{\Delta V}{2kT} \qquad (71)$$

ΔV is the potential energy drop a hole encounters in traversing the base.

A second useful expression for w/f is obtained from (61)

$$\frac{w}{f} = \frac{q}{2kT} E w = \frac{q}{2kT} \int_0^w E \, dx$$

$$= -\frac{1}{2} \int_0^w \frac{d \ln N}{dx} dx = \frac{1}{2} \ln \frac{N(0)}{N(w)} \qquad (72)$$

Four forms of this important parameter are used in the literature.

$$\frac{w}{f} = \eta = \frac{\Delta V}{2kT} = \frac{1}{2} \ln \frac{N_e}{N_c} \qquad (73)$$

The first is used by Krömer[2] and will be used here, the second is used by Lee[3], the third by Krömer[4] and the results of Moll and Ross[1] are stated

[3] Lee, C.A., A High Frequency Diffused Base Germanium Transistor, Bell System Technical Journal, January, 1956.

[4] Kromer, H., The Drift Transistor, Transistors I, RCA Laboratories 1956.

in terms of N_e/N_c. Values of w/f up to 4 are practical. The emitter efficiency drops for higher values which puts a limit on the amount of potential drop in the base.

COMMON-BASE ACTIVE SOLUTION

The solution will closely parallel that under the same heading in Part I. Boundary conditions are applied to (70) instead of (6). The two solutions reduce to the same thing if w/f and w/L_p are taken as zero, that is, if no field or recombination are present.

The boundary conditions on (70) are

$$\begin{cases} P(s,w) = 0 \\ I(s,0) = \gamma_N \dfrac{\Delta I_e}{s} \end{cases} \qquad (74)$$

The constants A and B may be found giving

$$P(s,x) = \frac{\Delta I_e \, \gamma_N \, e^{x/f}}{q \, D_p \, \Gamma s} \; \frac{\sinh \Gamma (w-x)}{\cosh \Gamma w + \frac{w/f}{\Gamma w} \sinh \Gamma w} \qquad (75)$$

The collector current is obtained by applying (65) to (75) and setting x = w.

$$I(s,w) = \frac{\Delta I_e \, \gamma_N \, e^{w/f}}{s} \; \frac{1}{\cosh \Gamma w + \frac{w/f}{\Gamma w} \sinh \Gamma w} \qquad (76)$$

The time scale will be normalized as before. The theorem states that if we multiply the function by a constant, say w^2/D_p; and multiply s by the same constant, then the time scale is multiplied by the reciprocal or D_p/w^2.

$$\tilde{I}(\lambda, w) = \frac{\Delta I_e \, \gamma_N \, e^{w/f}}{\lambda} \; \frac{1}{\cosh \sqrt{\lambda + \frac{w^2}{f^2}} + \frac{w/f}{\sqrt{\lambda + \frac{w^2}{f^2}}} \sinh \sqrt{\lambda + \frac{w^2}{f^2}}} \qquad (77)$$

where $\lambda = \frac{w^2}{D_p} s$ and $T = \frac{D_p}{w^2} t$

It would be possible at this point to shift the poles by w^2/f^2, but the result is just a change in notation and probably not worth the added complexity.

The above function has poles at $\lambda = 0$ and

$$\cosh\sqrt{\lambda + w^2/f^2} = \cos\sqrt{-\lambda - \frac{w^2}{f^2}} = \frac{-w/f \;\; \sin\sqrt{-\lambda - \frac{w^2}{f^2}}}{\sqrt{-\lambda - \frac{w^2}{f^2}}}$$

or

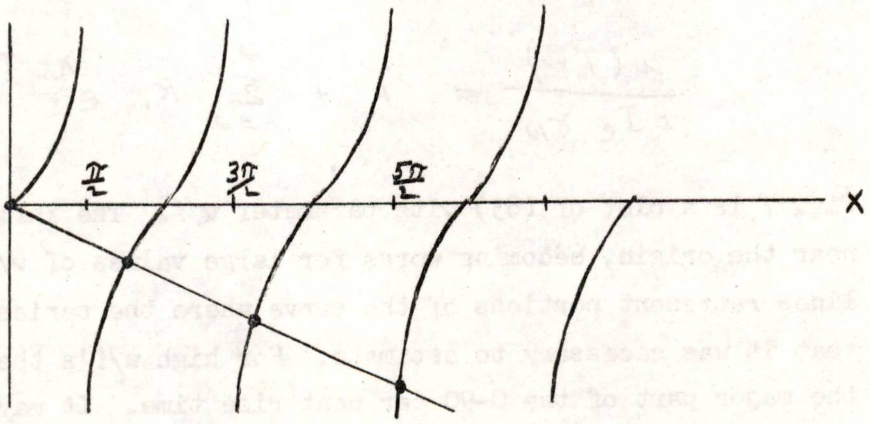
$$\sqrt{-\lambda - \frac{w^2}{f^2}} = -\frac{w}{f}\,\tan\sqrt{-\lambda - \frac{w^2}{f^2}} \qquad (78)$$

The solution to this transcendental equation may be seen from the intersection of the curves for Tan x and for

$$-\frac{x}{w/f}$$

where

$$x = \sqrt{-\lambda - w^2/f^2}$$



For w/f = 0,

$$x_{1n} = \left(\frac{2n+1}{2}\right)\pi \qquad \text{or} \qquad \lambda_{1n} = -\left(\frac{2n+1}{2}\right)^2\pi^2 \qquad (79)$$

which checks with (11). For w/f ≠ 0 the values of x_{1n} occur between

$$\left(\frac{2n+1}{2}\right)\pi \qquad \text{and} \qquad \left(\frac{2n+2}{2}\right)\pi$$

The values of λ_{1n} then are $-x_{1n}^2 - w^2/f^2$. Table I(a) gives λ_{1n} for various values of w/f.

The inversion of (77) is accomplished as follows:

$$\frac{\tilde{I}(\lambda,w)}{\Delta I_e \gamma_N} = \frac{e^{w/f}}{\lambda\left(\cos x + \frac{w/f}{x}\sin x\right)} = \frac{K_0}{\lambda} + \sum_{n=0}^{\infty}\frac{K_{1n}}{\lambda-\lambda_{1n}} \qquad (80)$$

$$K_0 = \frac{e^{w/f}}{\cos j\, w/f + \frac{1}{j}\sin j\, w/f} = 1 \qquad (81)$$

$$K_{1n} = \left.\frac{e^{w/f}(\lambda-\lambda_{1n})}{\lambda\left(\cos x + \frac{w/f}{x}\sin x\right)}\right|_{\substack{\lambda=\lambda_{1n}\\ x=x_{1n}}} = \frac{-2e^{w/f}x_{1n}}{\lambda_{1n}\left[\left(1+\frac{w/f}{x_{1n}^2}\right)\sin x_{1n} - \frac{w/f}{x_{1n}}\cos x_{1n}\right]} \qquad (82)$$

Values of the residues, K_{1n}, are given in Table I(b). The inversion of (80) gives

$$\frac{\hat{i}(T,w)}{\Delta I_e \gamma_N} = 1 + \sum_{n=0}^{\infty} K_{1n}\, e^{\lambda_{1n} T} \qquad (83)$$

Fig. 7 is a plot of (83) with parameter w/f. The infinite series diverges near the origin, becoming worse for large values of w/f. The dotted lines represent portions of the curve where the series diverged so badly that it was necessary to estimate. For high w/f's the delay represents the major part of the 0-90 per cent rise time. It may be shown that if diffusion is neglected, the response is a delayed step with delay

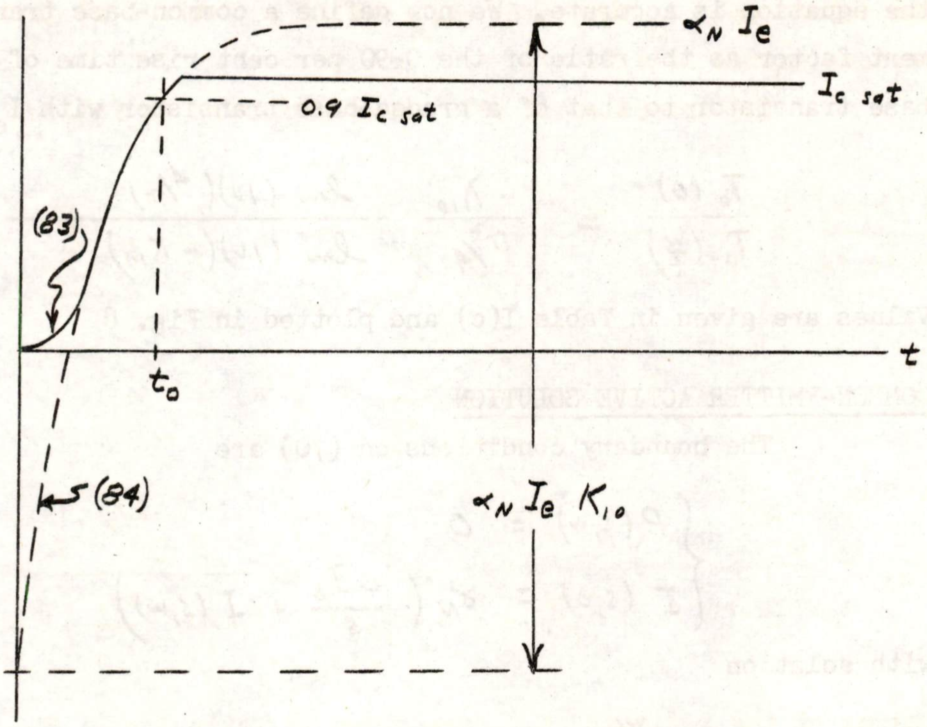
$$T = \frac{1}{2\, w/f}$$

For large values of w/f the actual response approaches this delayed step.

To find rise times, the series in (83) is terminated after one term.

$$\frac{\hat{i}(T,w)}{\Delta I_e \gamma_N} = 1 + K_{10}\, e^{\lambda_{10} T} \qquad (84)$$

For w/f = 0 this reduces to (15).



The rise time may be found using the above figure and (84).

$$\alpha_N I_e \left(1 + K_{10}\, e^{\lambda_{10} T_0}\right) = 0.9\, I_{c\,sat}$$

$$T_0 = \frac{1}{-\lambda_{10}} \ln \frac{(-K_{10})\, I_e}{I_e - \frac{0.9}{\alpha_N} I_{c\,sat}} \qquad (85)$$

For w/f = 0 this reduces to (17). Due to the slow convergence of the series, (84) becomes less accurate for large values of w/f. Fig. 2 shows that for 5 per cent error, the approximation holds for the 20-100 per cent portion of the rise time. For w/f = 4 the range is only 65-100 per cent. Equation (85) must be used with caution then for large values of w/f and for I_e larger than 1.5-2.0 times $I_{c\ sat}/\alpha_N$. However, for $I_e = I_{c\ sat}/\alpha_N$, the equation is accurate. We now define a common-base transient improvement factor as the ratio of the 0-90 per cent rise time of a homogeneous-base transistor to that of a graded-base transistor with $I_e = I_{c\ sat}/\alpha_N$.

$$\frac{T_0\,(0)}{T_0\,(\frac{w}{f})} = \frac{-\lambda_{10}}{\pi^2/4}\;\frac{\ln\,(10)(4/\pi)}{\ln\,(10)(-K_{10})} \tag{86}$$

Values are given in Table I(c) and plotted in Fig. 8

COMMON-EMITTER ACTIVE SOLUTION

The boundary conditions on (70) are

$$\begin{cases} P(s,w) = 0 \\ I\,(s,0) = \gamma_N\left(\dfrac{\Delta I_b}{s} + I\,(s,w)\right) \end{cases} \tag{87}$$

with solution

$$P(s,x) = \frac{\Delta I_b\,\gamma_N\,e^{w/f}}{q\,D_p\,\Gamma s}\;\frac{\sinh\Gamma(w-x)}{\cosh\Gamma w + \frac{w/f}{\Gamma w}\sinh\Gamma w - \gamma_N\,e^{w/f}} \tag{88}$$

and

$$I(s,w) = \frac{\Delta I_b\,\gamma_N\,e^{w/f}}{s}\;\frac{1}{\cosh\Gamma w + \frac{w/f}{\Gamma w}\sinh\Gamma w - \gamma_N\,e^{w/f}} \tag{89}$$

Next we normalize the time scale by letting $\lambda = w^2 s/D_p$ as before.

$$\tilde{I}(\lambda,w) = \frac{\Delta I_b\,\gamma_N\,e^{w/f}}{\lambda}\;\frac{1}{\cosh\sqrt{\lambda+\frac{w^2}{f^2}} + \frac{w/f}{\sqrt{\lambda+\frac{w^2}{f^2}}}\sinh\sqrt{\lambda+\frac{w^2}{f^2}} - \gamma_N\,e^{w/f}} \tag{90}$$

This has a pole at $\lambda = 0$, one for $\sqrt{\lambda + w^2/f^2}$ real, and an infinite number for $\sqrt{\lambda + w^2/f^2}$ imaginary or complex.

Fig. 1(b) shows the pole locations for $w/f = 0$. Application of the field causes the pole next to the origin to shift in a negative direction but not as fast as $-w^2/f^2$. The outer poles are shifted by $-w^2/f^2$, and apparently break into two single poles, then starting at minus infinity, recombine into second order poles and split into complex pairs. Further study on the behavior of these outer poles would be interesting, but because we are going to neglect their contribution to the response, such study would be only of academic interest.

The location of λ_1, the pole nearest the origin, is given in Table I(d) and (f) for $\alpha_N = \gamma_N = 0.9$ and 0.95. These values were found by a trial and error solution of the denominator of (90).

To invert (90) we set it equal to

$$\frac{K_0}{\lambda} + \frac{K_1}{\lambda - \lambda_1}$$

and solve for the residues in the usual manner.

$$K_0 = \frac{\Delta I_b \ \gamma_N \ e^{w/f}}{\operatorname{Cosh}\frac{w}{f} + \operatorname{Sinh}\frac{w}{f} - \gamma_N e^{w/f}} = \Delta I_b \frac{\alpha_N}{1-\alpha_N} \qquad (91)$$

where $\alpha_N = \gamma_N \beta_N = \gamma_N$

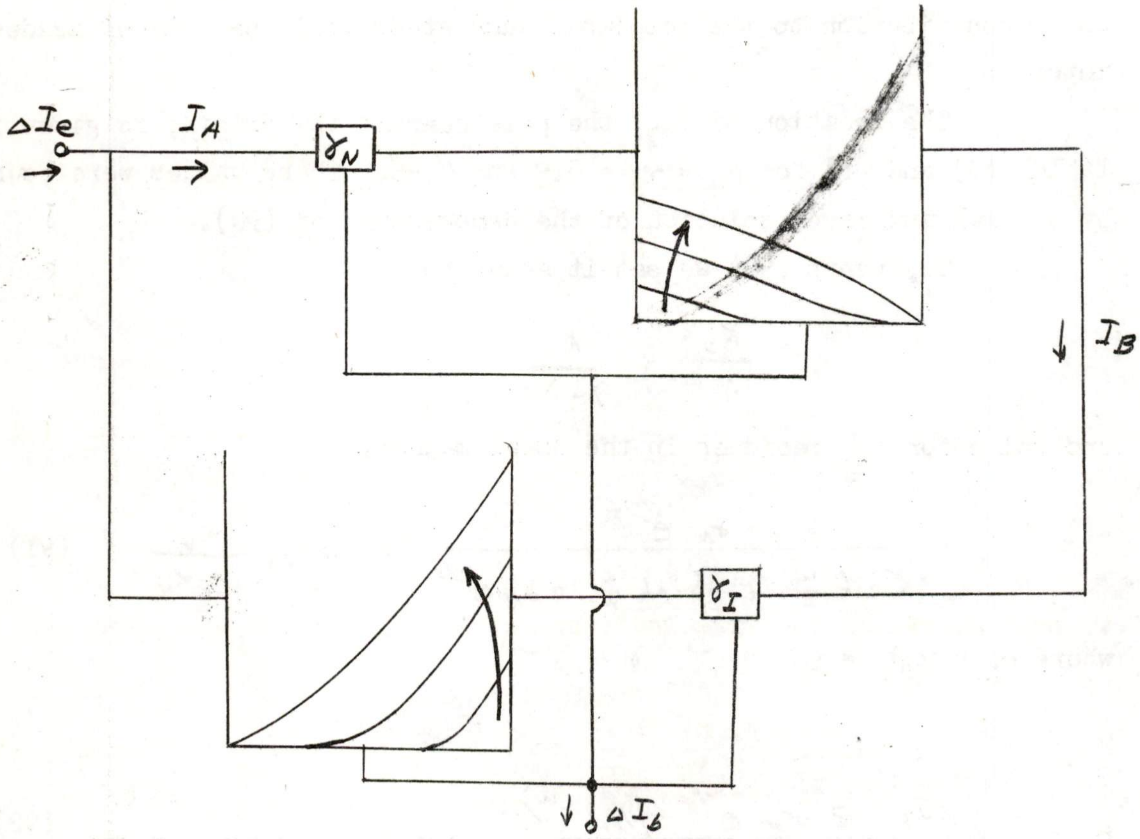
$$K_1 = \frac{\Delta I_b \ 2\alpha_N \ e^{w/f} \sqrt{\lambda_1 + w^2/f^2}}{\lambda_1 \Big[\Big(1 - \frac{w/f}{\lambda_1 + \frac{w^2}{f^2}}\Big) \operatorname{Sinh}\sqrt{\lambda_1 + \frac{w^2}{f^2}} + \frac{w/f}{\sqrt{\lambda_1 + w^2/f^2}} \operatorname{Cosh}\sqrt{\lambda_1 + \frac{w^2}{f^2}}} \qquad (92)$$

Numerical evaluation of these residues shows that K_1 is within a few percent of $-K_o$. The neglecting of the outer poles is therefore justified.

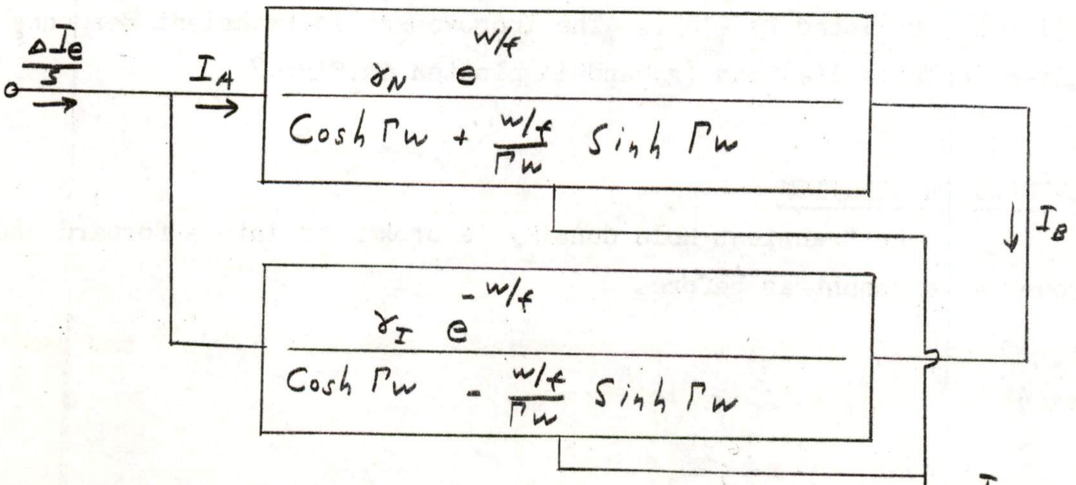
The rise and fall times are given by (33) and (34) with $2(1 - \alpha_N)$ replaced by $-\lambda_1$. The improvement in transient response is given in Table I(e) and (g) and is plotted in Fig. 8

SATURATION SOLUTION

The transient hole density is broken up into a forward and a reverse component as before.



Note that in the forward direction the field aids the flow while in the reverse direction it hinders it and requires a large gradient to force the holes across. The active region hole density plots may be replaced by the transfer functions derived in the common base analysis. The sign of w/f is negative for the reverse transfer function.



$\frac{\Delta I_e}{s}$
I_A
$\dfrac{\gamma_N\, e^{w/f}}{\operatorname{Cosh}\Gamma w + \frac{w/f}{\Gamma w}\operatorname{Sinh}\Gamma w}$
I_B
$\dfrac{\gamma_I\, e^{-w/f}}{\operatorname{Cosh}\Gamma w - \frac{w/f}{\Gamma w}\operatorname{Sinh}\Gamma w}$
$\frac{\Delta I_b}{s}$

$$I_A(s) = \frac{\Delta I_e}{s} + \frac{\gamma_N \gamma_I \quad I_A}{\operatorname{Cosh}^2 \Gamma w - \left(\frac{w/f}{\Gamma w}\right)^2 \operatorname{Sinh}^2 \Gamma w}$$

$$I_B(s) = \frac{\Delta I_e \, \gamma_N \, e^{w/f} \left(\operatorname{Cosh} \Gamma w - \frac{w/f}{\Gamma w} \operatorname{Sinh} \Gamma w\right)}{s\left[\operatorname{Cosh}^2 \Gamma w - \left(\frac{w/f}{\Gamma w}\right)^2 \operatorname{Sinh}^2 \Gamma w - \gamma_N \gamma_I\right]} \qquad (93)$$

$$P_c(s) = \frac{\Delta I_e \, \gamma_N \gamma_I \, e^{w/f} \operatorname{Sinh} \Gamma w}{q D_p \Gamma s\left[\operatorname{Cosh}^2 \Gamma w - \left(\frac{w/f}{\Gamma w}\right)^2 \operatorname{Sinh}^2 \Gamma w - \gamma_N \gamma_I\right]} \qquad (94)$$

We next normalize the time scale by letting $\lambda = w^2 s/D_p$.

$$\tilde{P}_c(\lambda) = \frac{\Delta I_e \, \gamma_N \gamma_I \, e^{w/f} \operatorname{Sinh}\sqrt{\lambda + w^2/f^2}}{q D_p \lambda \sqrt{\lambda + w^2/f^2}\left[\operatorname{Cosh}^2\sqrt{\lambda + w^2/f^2} - \left(\frac{w/f}{\sqrt{\lambda + w^2/f^2}}\right)^2 \operatorname{Sinh}\sqrt{\lambda + w^2/f^2} - \gamma_N \gamma_I\right]} \qquad (95)$$

This has a pole at $\lambda = 0$, one for $\sqrt{\lambda + w^2/f^2}$ real, and an infinite number for $\sqrt{\lambda + w^2/f^2}$ imaginary or complex. The outer poles will be neglected as in the common-emitter solution. The pole nearest the origin, λ_1, may be found by a solution of the equation below.

$$\operatorname{Cosh}^2\sqrt{\lambda_1 + w^2/f^2} - \frac{w^2/f^2}{\lambda_1 + w^2/f^2} \operatorname{Sinh}^2\sqrt{\lambda_1 + w^2/f^2} - \gamma_N \gamma_I = 0 \qquad (96)$$

$$\lambda_1 + w^2/f^2 = w^2/f^2 \, \frac{\operatorname{Tanh}^2\sqrt{\lambda_1 + w^2/f^2}}{1 - \frac{\gamma_N \gamma_I}{\operatorname{Cosh}^2\sqrt{\lambda_1 + w^2/f^2}}}$$

For large values of x we can approximate Cosh x by 1/2 e^{+x} and can consider e^{-x} small with respect to one.

$$\lambda_1 + w^2/f^2 \cong \frac{w^2}{f^2} \frac{\left(\dfrac{1 - e^{-2\sqrt{\lambda_1 + w^2/f^2}}}{1 + e^{-2\sqrt{\lambda_1 + w^2/f^2}}}\right)^2}{1 - 4\gamma_N\gamma_I e^{-2\sqrt{\lambda_1 + w^2/f^2}}}$$

$$\cong \frac{w^2}{f^2}\left(1 - 4e^{-2\sqrt{\lambda_1 + w^2/f^2}}\right)\left(1 + 4\gamma_N\gamma_I e^{-2\sqrt{\lambda_1 + w^2/f^2}}\right)$$

$$\cong \frac{w^2}{f^2}\left[1 - 4(1-\gamma_N\gamma_I)e^{-2\sqrt{\lambda_1 + w^2/f^2}}\right]$$

$$\lambda_1 \cong -4\frac{w^2}{f^2}(1-\gamma_N\gamma_I)e^{-2\sqrt{\lambda_1 + w^2/f^2}} \qquad (97)$$

With $\omega_N = \omega_I$, (46), the expression of Ebers and Moll, becomes

$$\lambda_1 = -\frac{w^2}{D_p}\frac{\omega}{2}(1-\alpha_N\alpha_I) = -(1-\alpha_N\alpha_I) \qquad (98)$$

The improvement factor is the ratio of (97) to (98).

$$\frac{\lambda_1(w/f)}{\lambda_1(0)} = 4\frac{w^2}{f^2}e^{-2\sqrt{\lambda_1 + \frac{w^2}{f^2}}} \cong 4\frac{w^2}{f^2}e^{-2w/f} \qquad (99)$$

A numerical solution of (96) shows that (99) is a good approximation for $w/f > 2$. This factor is given in Table I(h).

It is startling to note that this one-dimensional theory predicts longer storage times instead of shorter. For $w/f = 4$, the storage time (for the same base width and $\gamma_N\gamma_I$) is 39 times as long as for $w/f = 0$. In practice, however, measurements on the Philco drift transistor show little change in this normalized storage time.

To explain this it is necessary to find the final or steady-state value of the hole densities in the diagram on page 12. The current

i_B may be found by applying the final value theorem to (93). The final value of i_A is just i_B/γ_N.

$$i_B\Big|_{t=\infty} = \lim_{s\to 0} s\, I_B(s) = \frac{\Delta I_e\, \gamma_N\, e^{w/f}\left(\cosh w/f - \sinh w/f\right)}{\cosh^2\frac{w}{f} - \sinh^2\frac{w}{f} - \gamma_N\gamma_I} = \frac{\Delta I_e\, \gamma_N}{1-\gamma_N\gamma_I} \qquad (100)$$

The final value of the hole density of the forward solution is found from (75).

$$p_N(x,\infty) = \lim_{s\to 0} s\, P_N(x,s) = \frac{i_A\, \gamma_N\, e^{w/f}\, \sinh\frac{w-x}{f}}{q D_p\, w/f\left(\cosh w/f + \sinh\frac{w}{f}\right)}$$

$$= \frac{i_A\, \gamma_N}{q D_p}\; \frac{1 - e^{-2\frac{w}{f}\left(1-\frac{x}{w}\right)}}{2\, w/f} \qquad (101)$$

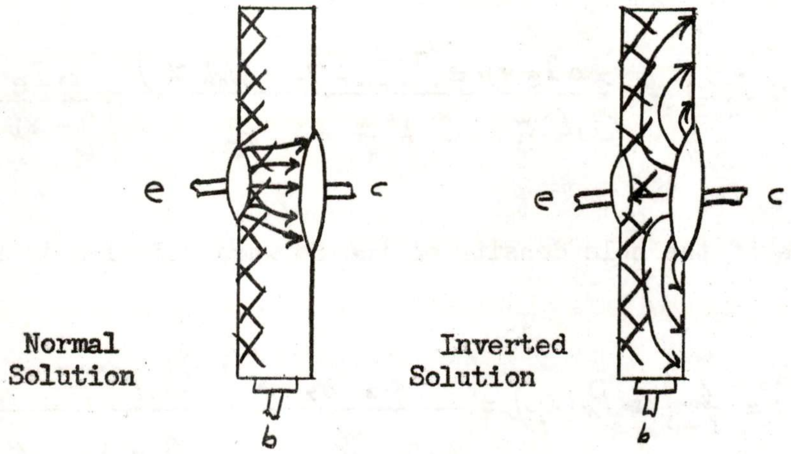
Similarly

$$p_I(x,\infty) = \frac{i_B\, \gamma_I}{q D_p}\; \frac{e^{2\frac{w}{f}\frac{x}{w}} - 1}{2\, w/f} \qquad (102)$$

Upon substituting for i_A and i_B, we can obtain the total hole density.

$$p_T = p_N + p_I = \frac{\Delta I_e\, \gamma_N}{q D_p\left(1-\gamma_N\gamma_I\right)}\left[\frac{1-e^{-2\frac{w}{f}\left(1-\frac{x}{w}\right)}}{2\, w/f} + \gamma_I\, \frac{e^{2\frac{w}{f}\frac{x}{w}}-1}{2\, w/f}\right] \qquad (103)$$

The functions in brackets are plotted in Fig. 9 for w/f = 0, 2, and 4. Notice that extremely high collector hole densities are necessary to overcome the field. Fig. 9 is correct for a one-dimensional transistor but in a practical device, the holes are deflected out towards the surface as shown below. The hole density at the collector is prevented from reaching a high value.



Normal Solution

Inverted Solution

Measurements of α_I confirm this behavior. It varies from 0.1 for the Bell drift transistor to 0.5 for the Philco drift transistor. Another factor contributing to the failure to reach large values of collector hole density is volume recombination in the outer base.

The one-dimensional model, then, predicts longer storage times, but as confirmed by α_I measurements and actual storage time measurements, the model breaks down.

SINUSOIDAL SOLUTION - COMMON BASE

Much of the work in this section has been done previously by Krömer and Lee, but will be repeated here as a comparison to the common-emitter solution and for completeness.

To solve the differential equation (67) for the sinusoidal case we let

$$P(x,t) = P(x,j\omega)\, e^{j\omega t}$$

and obtain

$$\frac{d^2 P(x,j\omega)}{dx^2} - \left(\frac{2}{f}\right)\frac{d P(x,j\omega)}{dx} = \frac{1}{D_p}\, j\omega\, P(x,j\omega) \qquad (104)$$

which is the same as (68) with s replaced by jω. All the transfer functions derived hold for the sinusoidal case if this replacement is made.

The common-base current transfer function, (76), becomes

$$\frac{I(j\omega, w)}{I_e(j\omega)} = \frac{\gamma_N e^{w/f}}{\operatorname{Cosh}\sqrt{\frac{w^2}{D_p} j\omega + \frac{w^2}{f^2}} + \frac{w/f}{\sqrt{\frac{w^2}{D_p} j\omega + \frac{w^2}{f^2}}} \operatorname{Sinh}\sqrt{\frac{w^2}{D_p} j\omega + \frac{w^2}{f^2}}} \tag{105}$$

which has been previously obtained by Krömer.[2]

We now define an ω_α, which is the frequency at which α is down 3 db.

$$\omega_\alpha = K \frac{D_p}{w^2} \tag{106}$$

The factor K may be found by solving (105) for the value of

$$\frac{w^2}{D_p} j\omega_\alpha = jK$$

for which α is 3 db down. The numerical work is aided by the relation Cosh $(x + jy)$ = Cosh x Cos y + j Sinh x Sin y. For $w/f > 2$ the hyperbolic functions Cosh $(x + jy)$ and Sinh $(x + jy)$ may be approximated by

$$\frac{1}{2} e^{x+jy}.$$

Values of K are given in Table I(i) for w/f = 0,2, and 4. Column (j) gives the improvement factor

$$\frac{K(w/f)}{K(0)} = \frac{\omega_\alpha(w/f)}{\omega_\alpha(0)}$$

which is plotted in Fig. 8 vs w/f. Krömer[4] obtained $(w/f)^{3/2}$ for an improvement factor which is shown dotted as a comparison.

Knowing K, we may now plot the magnitude and phase of α vs ω/ω_α. This is shown in Fig. 10 for w/f = 0 and 4. Lee[3] has plotted the magnitude of α vs ω for w/f = 0, ±1, ±2, and ±3 using equation (105) as derived by Krömer.

The first order approximation[5] for α with w/f = 0 can be obtained from (105). The first two terms of the power series for Cosh x are retained.

[5] Steele, E.L., "Theory of alpha for PNP diffused junction transistors," Proceedings of the I.R.E, vol 40 no 11, (Nov 1952)

$$\alpha(j\omega) = \frac{\gamma_N}{1 + j\omega \frac{w^2}{2D_p}} = \frac{\gamma_N}{1. + j\, \omega/\omega_\alpha} \qquad (107)$$

$$\text{with } K = 2$$

The magnitude and phase of this approximation are also shown in Fig. 10. Middlebrook[6] has plotted this approximation and the exact function for w/f = 0.

The additional phase shift introduced by the field is related to the relatively large delay that was found in the step response.

SINUSOIDAL SOLUTION - COMMON EMITTER

The common-emitter current transfer function, (89), becomes

$$\frac{I(j\omega, w)}{I_b(j\omega)} = \frac{\gamma_N e^{w/f}}{\operatorname{Cosh}\sqrt{\frac{w^2}{D_p} j\omega + w^2/f^2} + \frac{w/f}{\sqrt{\frac{w^2}{D_p} j\omega + w^2/f^2}} \operatorname{Sinh}\sqrt{\frac{w^2}{D_p} j\omega + w^2/f^2} - \gamma_N e^{w/f}} \qquad (108)$$

It is difficult to find a suitable symbol for (108). The Greek letter, β, is commonly used as the ratio of collector to base current, but it has also been used as the common-base transport factor in much of the literature (see (13)). For the purposes of this paper, the letter b will be used.

We now define an ω_b, which is the frequency at which $|b|$ is 3 db down.

$$\omega_b = K_e (1 - \alpha_N) \frac{D_p}{w^2} \qquad (109)$$

The factor K_e may be found as before and is given in Table I(k) for $\gamma_N = .95$. Column (1) gives the improvement factor

$$\frac{K_e(w/f)}{K_e(0)} = \frac{\omega_b(w/f)}{\omega_b(0)}$$

which is plotted in Fig. 8 vs w/f. Notice that it coincides with the common-emitter transient-response improvement factor.

The first-order approximation for b with w/f = 0 can be obtained from (108).

[6] Middlebrook, R.D., "An Introduction to Junction Transistor Theory," Page 200, Wiley 1957, New York.

$$b(j\omega) = \frac{\alpha_0}{1 + \frac{w^2}{2D_p} j\omega - \alpha_0} \qquad \frac{\frac{\alpha_0}{1-\alpha_0}}{1 + \frac{j\omega}{\omega_b}} \qquad (110)$$

which gives the value of K_e as 2.

Fig. 11 shows the magnitude and phase of $b(j\omega)/b(0)$ for $w/f = 0$, and 4, and of the first-order approximation (110).

The first neglected term of the expansion of Cosh x is multiplied by $(1 - \alpha_o)$ in (110) while it is not in the common-base equation (107). This explains why the first-order approximation (110) is so good, that is, why K_e is so near 2 and why the magnitude and phase of (108) closely approximate those of (110); and also accounts for (92) being approximately equal to (91).

CONCLUSIONS

The use of the Laplace transform seems to be a powerful method of finding the transient response of a one-dimensional transistor model. It yields exact results with little effort.

Attachments

Table I
Drawing Nos. A-81374 -
A-48870-G -
A-48871-G -
A-48872-G -
A-48873-G -
A-48869-G -
A-48868-G -

Signed R.C. Johnston
R. C. Johnston

COMMON-BASE				
		(a)	(b)	(c)
w/f	n	λ_{1n}	K_{1n}	$\frac{T_o(0)}{T_o(w/f)}$
0	0	$-(\pi/2)^2$	$-4/\pi$	1
	1	$-(3\pi/2)^2$	$+4/3\pi$	
	2	$-(5\pi/2)^2$	$-4/5\pi$	
2	0	-9.2395	-2.2665	3.05
	1	-29.8745	+2.1948	
	2	-69.544	-1.6235	
4	0	-22.607	-5.7031	5.77
	1	-44.666	+9.6242	
	2	-84.938	-9.1722	
	3	-144.478	+7.8609	
	4	-223.590	-6.6010	

	COMMON-EMITTER				SATURATION
w/f	(d) $\alpha_n = .95$	(e)	(f) $\alpha_n = .90$	(g)	(h)
	λ_1	$\frac{\lambda_1(w/f)}{\lambda_1(0)}$	λ_1	$\frac{\lambda_1(w/f)}{\lambda_1(0)}$	$\frac{\lambda_1(w/f)}{\lambda_1(0)}$
0	$-2(1-\alpha_n) =$ -0.1	1	-0.2	1	1.0
2	-0.2697	2.697	-.5478	2.739	0.299
4	-0.4662	4.662	-0.953	4.76	0.0218

w/f	(i) Sinusoidal common-base	(j)	(k) Sinusoidal common-emitter	(*l*)
	K	$\frac{\omega_\alpha(w/f)}{\omega_\alpha(0)}$	K_e $\alpha_N = .95$	$\frac{\omega_b(w/f)}{\omega_b(0)}$
0	2.4324	1	2.0169	1
4	17.67	7.26	9.17	4.55

TABLE I

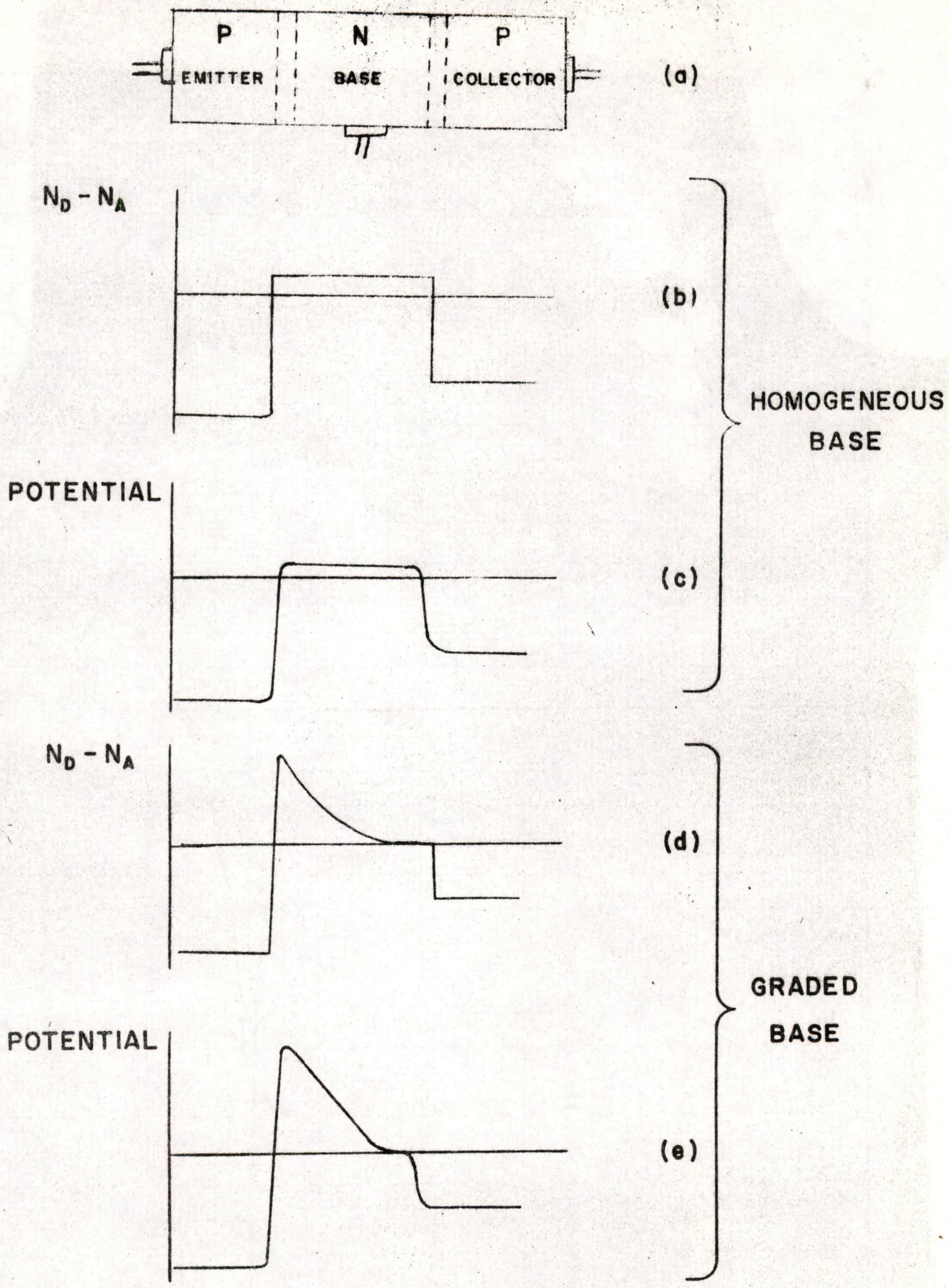


FIG. 5

IMPURITY DENSITY AND POTENTIAL DIAGRAMS

A-81374

A-48870-G

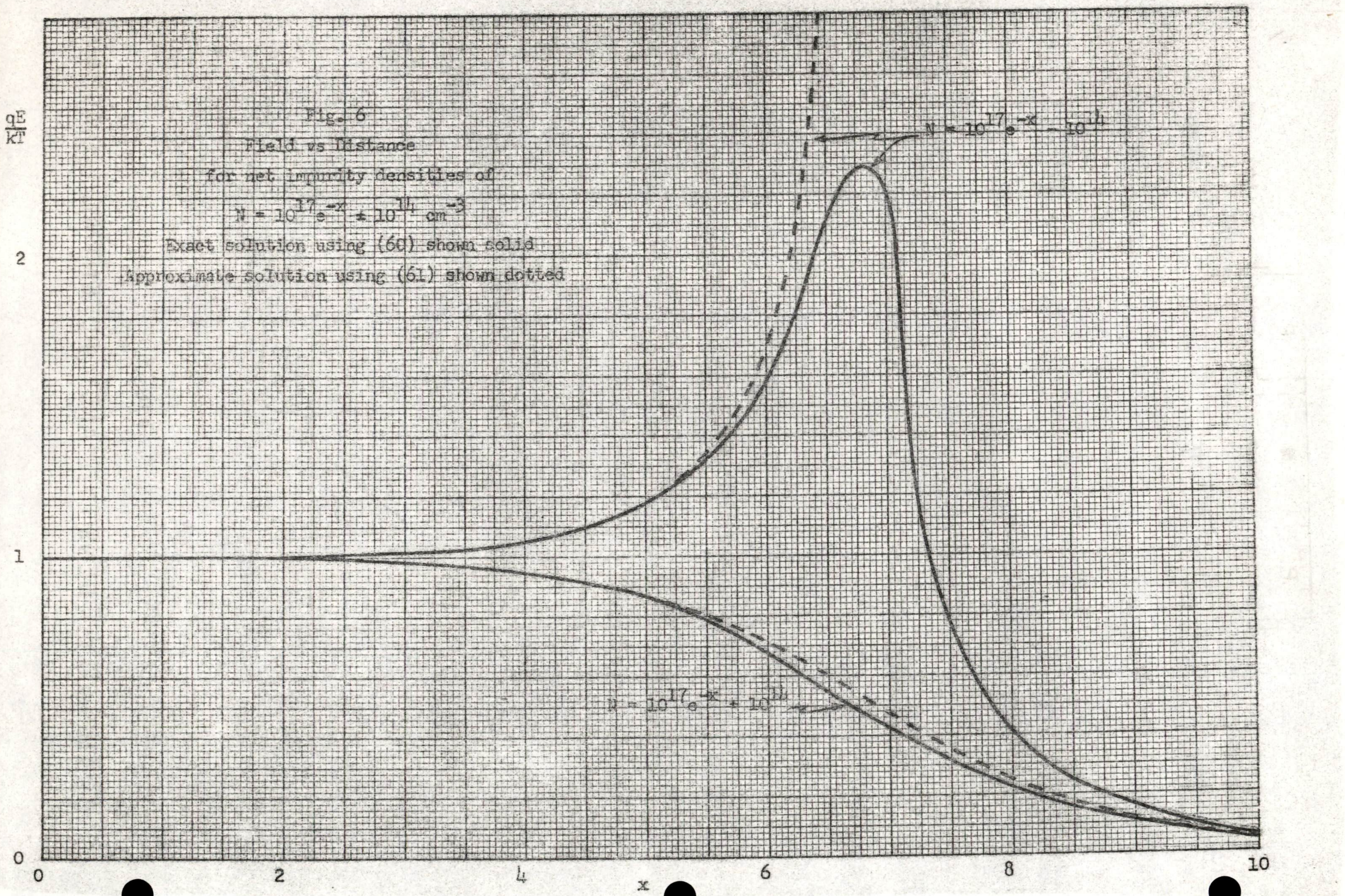


Fig. 6

Field vs Distance

for net impurity densities of

$N = 10^{17}e^{-x} \pm 10^{14}\ \text{cm}^{-3}$

Exact solution using (60) shown solid

Approximate solution using (61) shown dotted

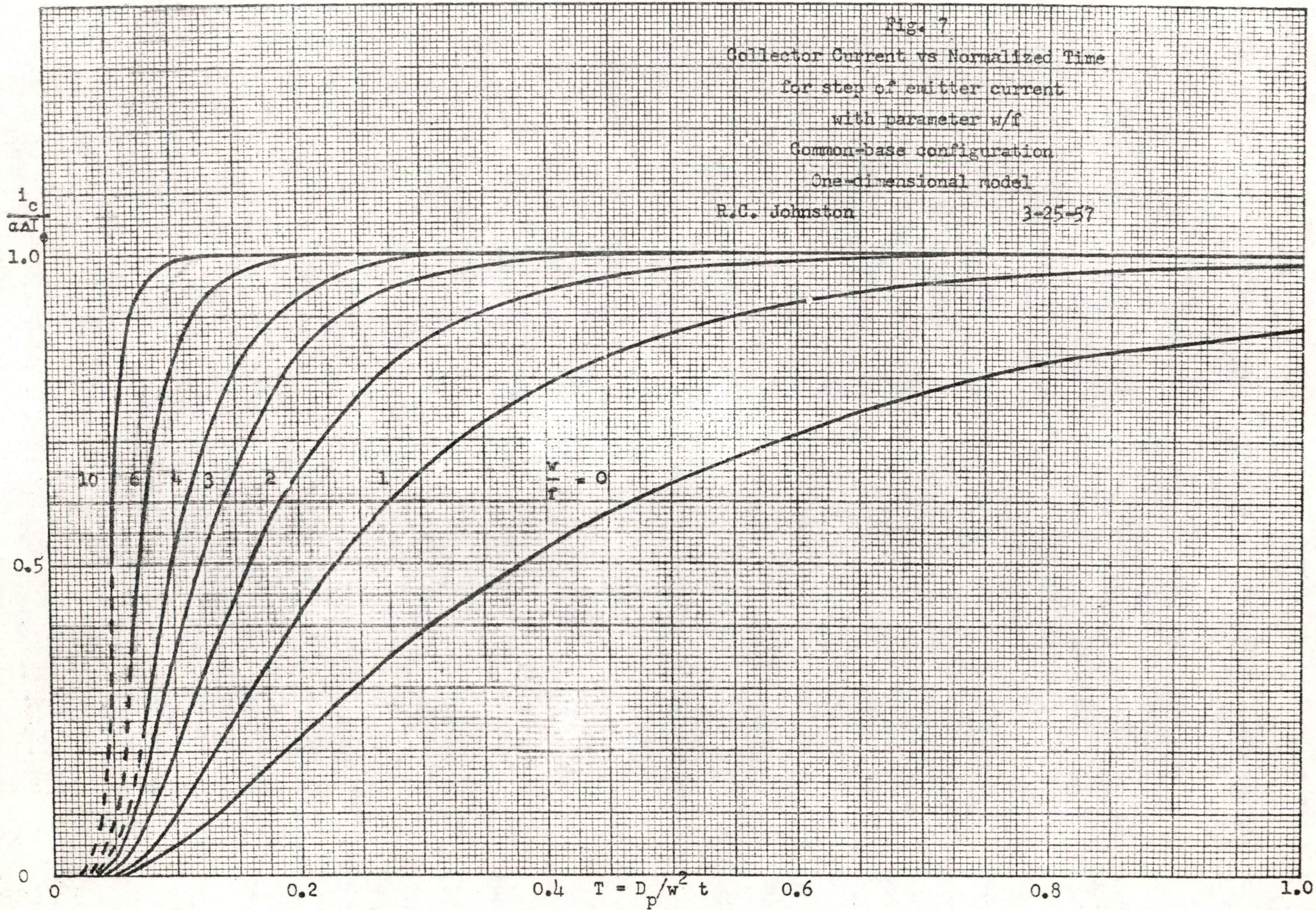


Fig. 7

Collector Current vs Normalized Time

for step of emitter current

with parameter w/f

Common-base configuration

One-dimensional model

R.C. Johnston 3-25-57

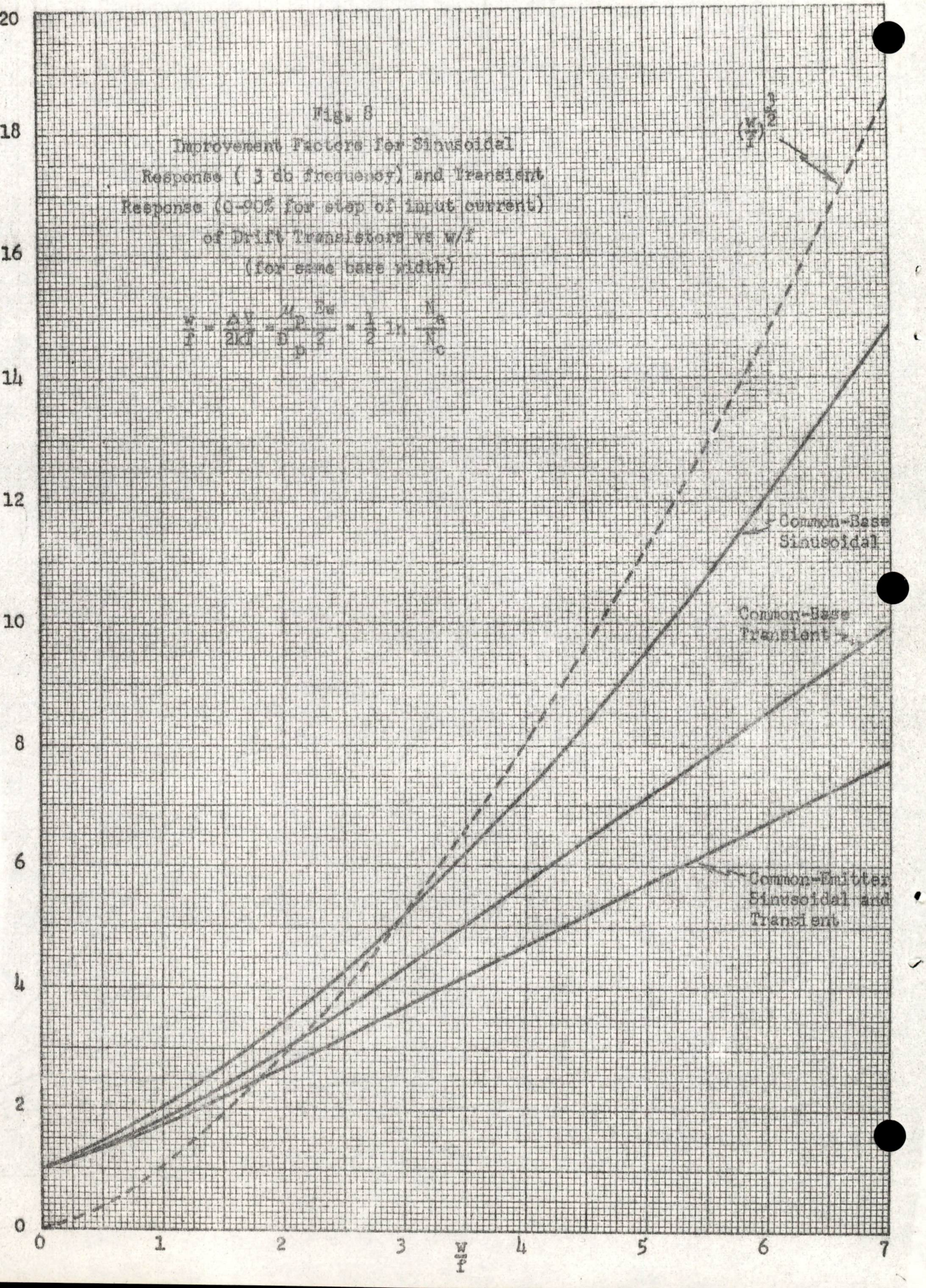


Fig. 8
Improvement Factors for Sinusoidal Response (3 db frequency) and Transient Response (0-90% for step of input current) of Drift Transistors vs w/f (for same base width)

$$\frac{w}{f} = \frac{\Delta V}{2kT} = \frac{\mu_p}{D_p}\frac{Ew}{2} = \frac{1}{2}\ln\frac{N_e}{N_c}$$

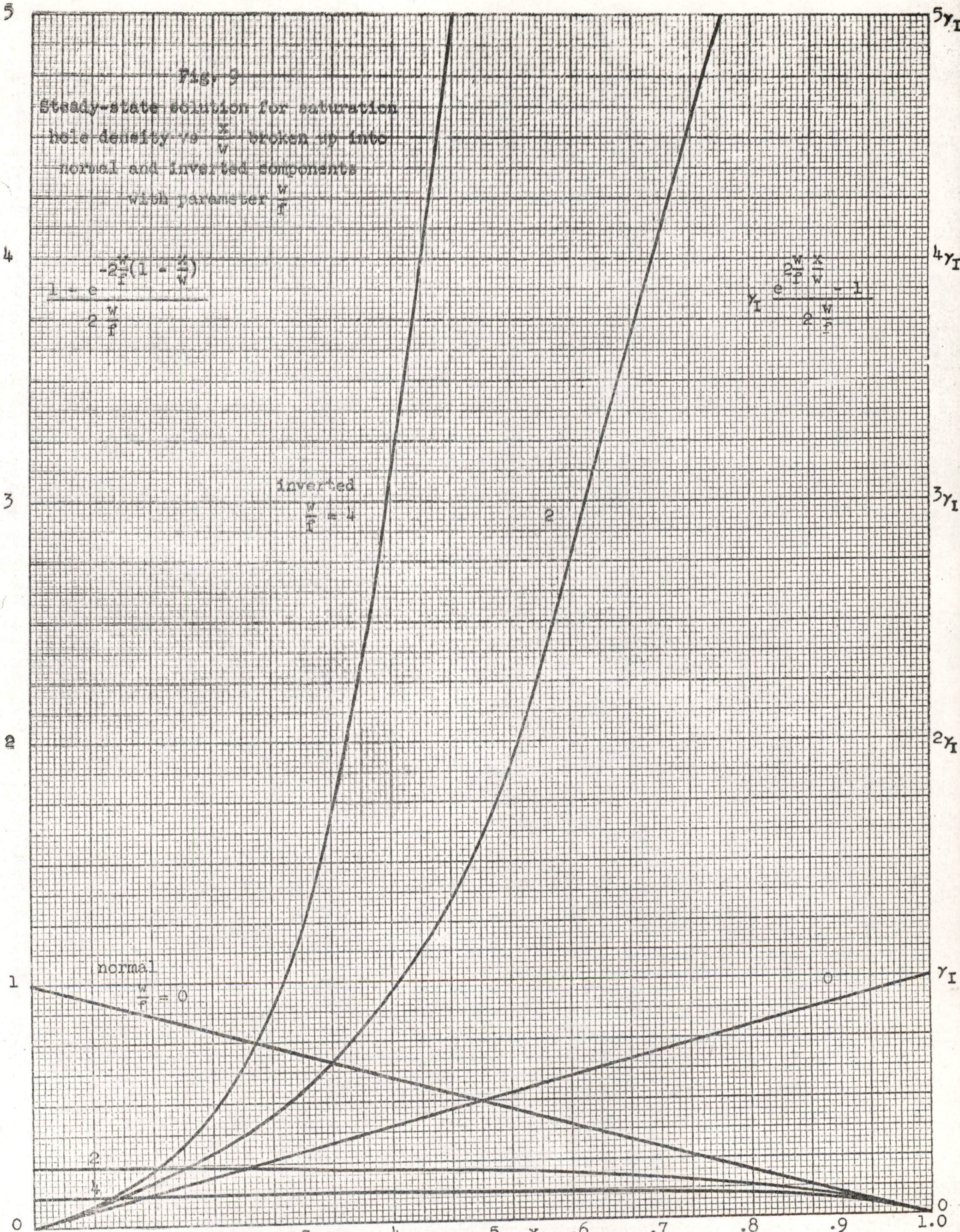


Fig. 9

Steady-state solution for saturation hole density vs $\frac{x}{w}$ broken up into normal and inverted components with paramster $\frac{w}{f}$

A-48873-G

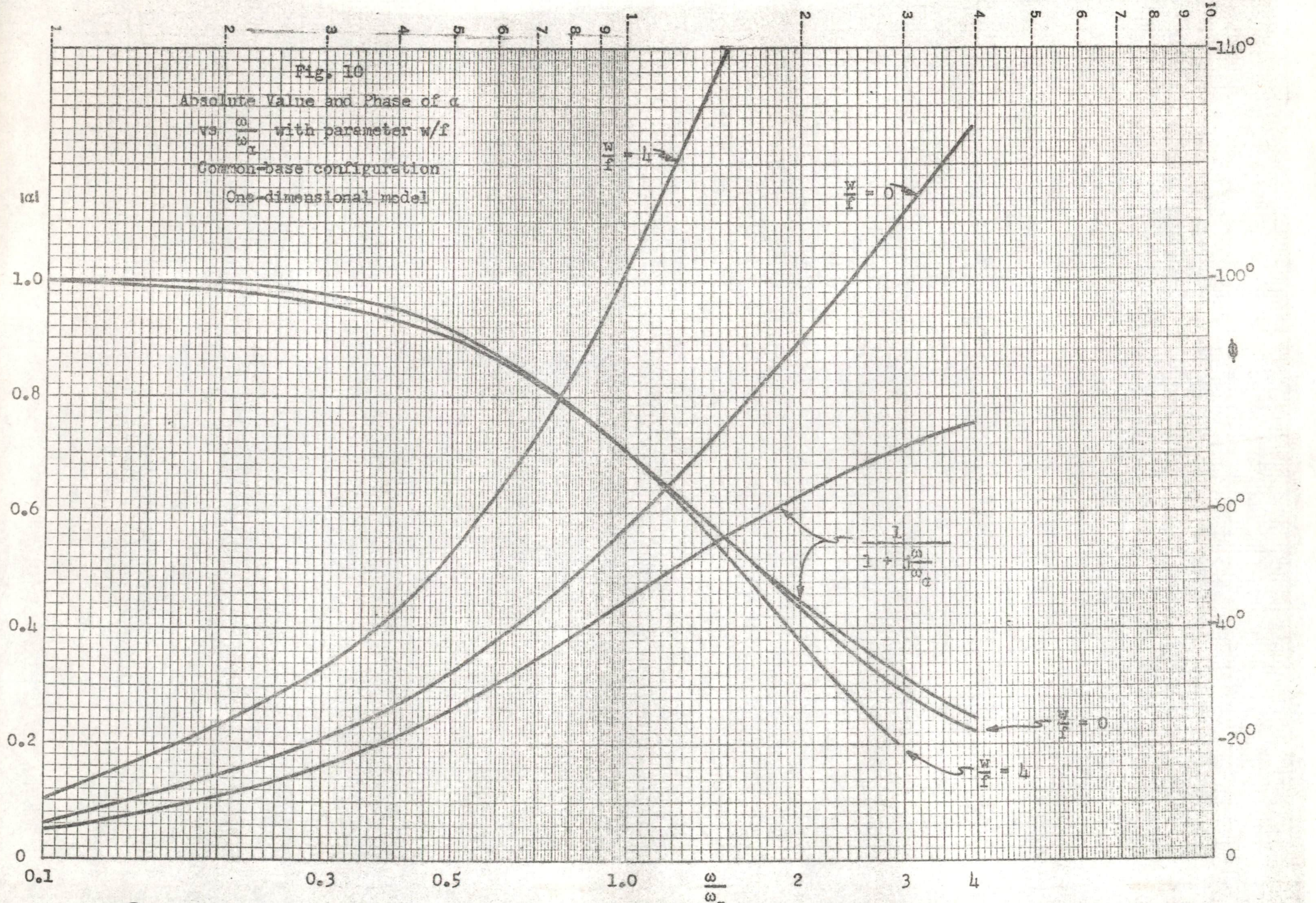


Fig. 10

Absolute Value and Phase of α

vs $\frac{\omega}{\omega_\alpha}$ with parameter w/f

Common-base configuration

One-dimensional model

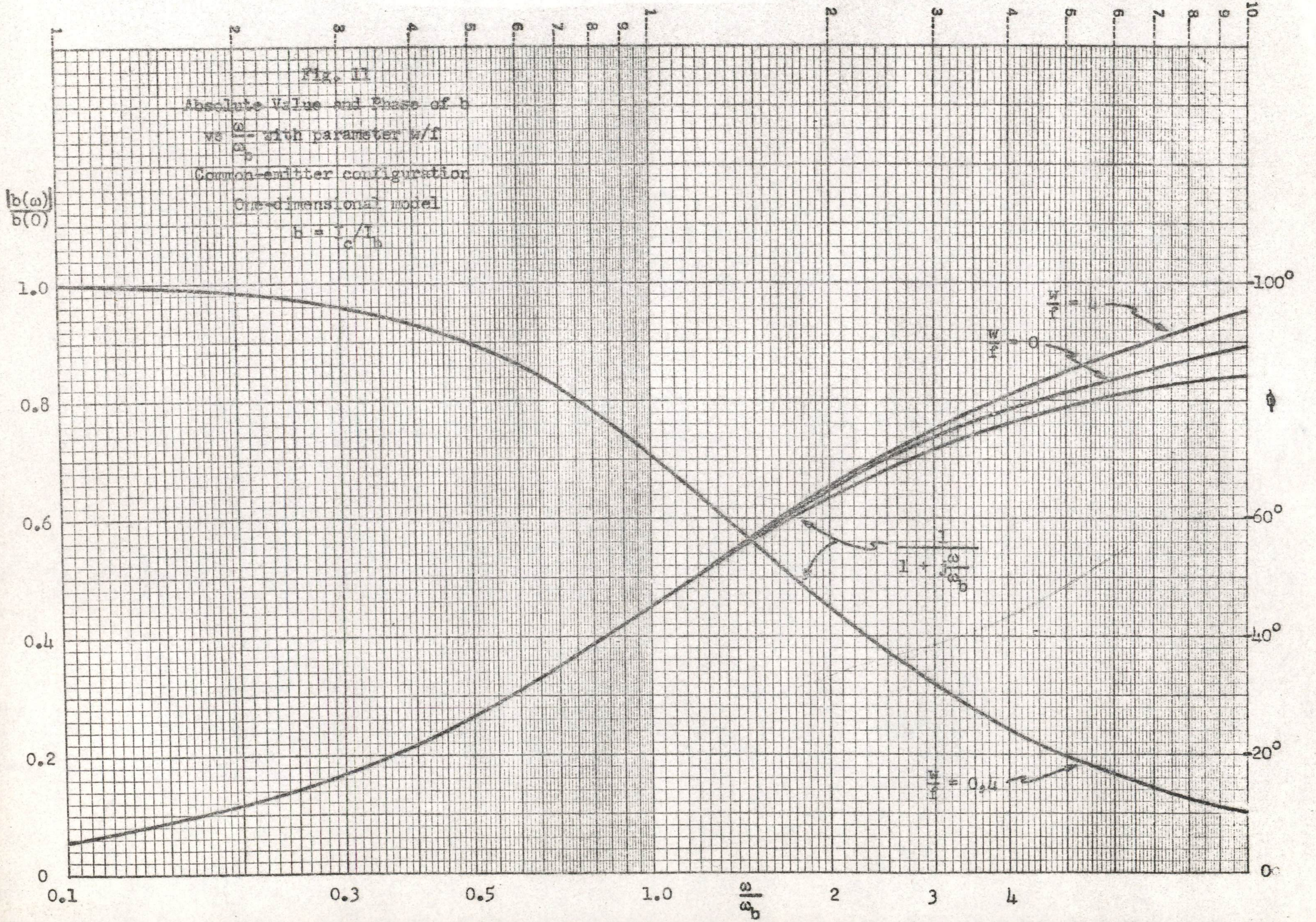


Fig. 11

Absolute Value and Phase of b vs $\frac{\omega}{\omega_b}$ with parameter w/f

Common-emitter configuration

One-dimensional model

$b = I_c/I_b$

R.L. Best

B077

Memorandum 6M-4955

Sheet 1 of 26 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: PARAMETER DISTRIBUTIONS FOR THE PHILCO L-5122 SURFACE-BARRIER TRANSISTOR

To: Group 63 Staff

From: Donald J. Eckl

Date: March 19, 1957

Approved: William N. Papian

Abstract: The Philco L-5122 is a high-frequency surface-barrier transistor intended for switching applications. Its bias voltages are the same as a pnp junction transistor. It has a maximum voltage rating of 6 volts and a maximum power limitation of 10 mw. Its junction temperature rises about 0.6°C per mw.

In brief the specifications at 3v and 0.5 ma I_e are as follows:

Common emitter current gain β	> 16
Common base output resistance	$> 200\ K\Omega$
Common base input impedance	$< 100\ \Omega$
Max frequency of oscillation	$>$ 30 mc/s.
$r_b' C_c$	$<$ 1500 μμsec.
I_{co} and I_{eo} at 5 volts	$<$ 3 μa.
$V_{ce\ sat\ 8}$	$<$ 100 mv.
$V_{ce\ sat\ 2}$	$<$ 70 mv.
Hole storage coefficient, K_s'	$<$ 105 mμsec.

It is the only commercially available transistor for high speed switching applications.

This note presents data obtained from the measurement of 30,000 transistors over a period of 2 years.

Distribution list:
Group 63 Staff

This document is issued for internal distribution and use only by and for Lincoln Laboratory personnel. It should not be given or shown to any other individuals or groups without express authorization. It may not be reproduced in whole or in part without permission in writing from Lincoln Laboratory.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

I.0 INTRODUCTION

The data presented here are the result of about 2 years experience with a total of some 30,000 surface-barrier transistors. Some of the early data refers to the Philco SB-100 unit which was specified for amplifier service. The L-5122 is an SB-100 type with specifications designed to insure satisfactory operation in TX-0 and TX-2 computer circuits. These specifications were developed over a period of time by close cooperation between Philco and Lincoln engineers.

The measured parameters fall into two general classes: those intended to serve as a quality control and those which determine circuit operating margins. Where complete data are available distributions are presented on 7000 production transistors. Where this was not possible smaller groups of typical units were carefully selected and used to obtain the necessary distributions. This is pointed out in the text and on the figures.

Much of this data does not appear explicitly on the L-5122 specification sheet and therefore is not guaranteed by Philco. However, we feel that transistors tested according to the published specifications will have characteristics substantially the same as those presented here.

2.0 CHARACTERISTIC CURVES FOR A TYPICAL L-5122 SWITCHING TRANSISTOR

The choice of a so-called "typical" transistor is always an arbitrary one since most transistors differ slightly in one or another respect. We have chosen transistors which have parameters near the mean values measured on a group of 7000 production transistors. For the collector characteristics the transistor used was:

LINCOLN NO.	10786
DATE RECEIVED	23 APRIL 1956
α_N(3v, 1/2 ma)	0.960
α_I(3v, 1/2 ma)	0.910
$V_{ce\ sat\ 8}$	69 mv
$V_{ce\ sat\ 2}$	50 mv
PUNCH THROUGH VOLTAGE	9.5 volts
STORAGE COEFFICIENT τ_s	40 mμsecs.
I_{co} (5 volts)	1.1 μa
I_{eo} (5 volts)	0.9 μa

The grounded-base collector characteristics are plotted in Fig. 1 out to a collector current of 17 ma, which is considerably above the normal rating of the transistor. Power, voltage, and current ratings, which are primarily determined by life and/or performance considerations, were not observed in plotting characteristics. Note particularly that the collector to base voltage is considerably positive in the saturation region and that the transistor has a high value of collector resistance at $V_{cb} = 0$ volts.

The grounded-emitter collector characteristics are shown in Fig. 2 out to about 18 ma. The points to notice here are the considerable fall off in gain and the decrease in collector resistance at high currents. V_{ce} remains negative at all times. A "blow-up" of the grounded-emitter characteristics around the origin is shown in Fig. 3. This shows that the curves intersect about 3 mv below the origin. The curves represent an expanded view of the saturation region near the origin.

The characteristics presented in Figs. 1, 2, and 3 were obtained on a Librascope x-y recorder using the circuit arrangement of Fig. 32.

The grounded-emitter saturation characteristics shown in Fig. 4 are not usually published and yet they rank in importance with the grounded-emitter collector characteristics for the d-c specification of a saturating switching transistor. Whereas, the normal collector characteristics spread out the active operating region and compress the saturation region, these curves spread out the saturation region and compress the active region. Both sets of curves are necessary.

The curves in Fig. 4 show that the active region starts at a V_{ce} of about 200 mv. The curvature of the curves in the active region, indicating a decrease in gain, is typical of the surface-barrier contact. The test point $V_{ce\ sat\ 2}$ is indicated on the graph. This is a quality control check rather than a circuit requirement. The saturation characteristics were obtained with the setup shown in Fig. 31.

Saturation current gain can be readily obtained from Fig. 4. The variation of current gain with collector current for various V_{ce} is plotted in Fig. 5. The same data as a function of emitter current is plotted in Fig. 6. These curves are obtained from the information in Fig. 4. A plot of $1/\beta$ against I_e should be nearly linear in the active region (200 mv curve). The test limits for the $V_{ce\ sat\ 8}$ and $V_{ce\ sat\ 2}$ check points are indicated. Similar sets of curves for a low β transistor are presented in Figs. 7 and 8.

A set of base input characteristics for a typical SBT is shown in Fig. 9. The curve for $R_L = \infty$ represents the limiting case for saturation. The $R_L = 150\,\Omega$ and V_c = const. curves are for the active region. Intermediate curves go from active to saturation at the break point. The slope of these curves represents the base input resistance. The base input circuit looks like about 100 Ω (saturated) or 200 Ω (active) in series with a − 0.2 to − 0.3 volt battery.

3.0 SPREADS IN CHARACTERISTIC CURVES USED IN CIRCUIT DESIGN

The spread in base input characteristics obtained from 50 surface-barrier transistors with zero collector current is shown in Figure 10. These curves represent the minimum base voltage which can be obtained in the saturation region. The input circuit has a minimum value in saturation at 1 ma base current which ranges from 90 Ω to 140 Ω in series with a 0.2 volt battery. The 50 transistors used to obtain this distribution were taken from early L-5122 production.

The next graph, Fig. 11, was included not because a 1K load represents any circuit configuration of particular interest, but rather because it was felt desirable to include some curves intermediate between complete saturation and fully active operation. The breaks in the curves between 0.1 and 0.2 ma base current represent the transition from active to saturation. The input circuit in this case looks like a 100 to 150 Ω resistance in series with a 0.25 to 0.3 volt battery.

Fig. 12 shows the spread of base input characteristics in the active region. It is immediately obvious that the spread is considerably greater than that obtained when the transistor is saturated. Here the transistor input circuit appears as a resistance of 140 Ω to 390 Ω in series with a 0.25 volt battery.

The spread in forward transfer characteristics (I_c vs V_{be}) for a collector to emitter voltage of 200 mv (bordering on active region) is presented in Fig. 13.

The 200 mv saturation characteristic spread is shown in Fig. 14. 200 mv is chosen since it is a limiting value for the saturation region. At a base current of 0.6 ma the output current ranges from 6 to 12 ma.

4.0 PARAMETER DISTRIBUTIONS OBTAINED FROM 7000 SURFACE-BARRIER TRANSISTORS

The next 9 graphs represent distributions obtained from incoming tests on 7000 surface-barrier transistors received during the period from September, 1955 to April, 1956. With the possible exception of punch-through voltage these parameters are of a "quality control" nature and have no direct correlation to circuit operating points. They obviously, however, affect circuit performance.

Forward and inverse beta distributions at 1 ma and 3 v are presented in Figs. 15 and 16. The test circuit is shown in Fig. 33. Although this is actually a d-c α test we have found it to agree, to within ± 3 units in the third place, with the small signal α measured at 3v and 1/2 ma.

The storage coefficient (τ_s) distribution is shown in Fig. 17. The test circuit is Fig. 34. This test is a measure of the microcoulombs of charge stored in the base region per ma of base current. These units turn out to be millimicroseconds. Thus the "time" units do not in any way represent the delay associated with saturation in normal circuit operation and should not be so construed. The actual storage delay depends on the circuit saturation-base-current and on the turn-off current drive.

The effective base lifetime τ_b is shown as a distribution in Fig. 18 for 500 transistors. This circuit (Fig. 36) and method of measurement is described by Lederhandler and Giacoletto in Proc. I.R.E., April, 1955. It gives an effective value which includes surface and volume effects.

The punch-through voltage distribution is given in Fig. 19. The test circuit is shown in Fig. 37. Measurement is made by increasing collector voltage until the emitter voltage is one volt. The punch-through voltage is then 1 volt less than the collector voltage.

The two $V_{ce\ sat}$ voltage distributions are given in Figs. 20 and 21 and the test circuit in Fig. 38.

The I_{co} and I_{eo} distributions are shown in Figs. 22 and 23.

A brief summary of the foregoing curves is presented in the table below:

Parameter	Lower Value	Median Value	Upper Value
β_n(3v, 1 ma)	10	27.6	~60
β_I(3v, 1 ma)	4	10.5	~25
τ_s μcoul/ma or mμsec	20	49	90
τ_b μsec	1	6.7	18
V_p volts	5	9.8	20
V_{ce} sat 8 mv	40	70	120
V_{ce} sat 2 mv	20	48	90
I_{co} μa	0.04	0.68	4.0 +
I_{eo} μa	0.03	0.55	~4.0

It will be noticed that some of the upper limits are above the published specs. This is the case only because some of these units were received before these specifications were firm. Later production meets the specifications quite satisfactorily in general.

5.0 SPECIAL PARAMETER DISTRIBUTIONS

The data presented in this section were obtained either from tests developed only recently or from those tests which are not considered necessary as a routine matter. This information was obtained from a group of about 50 transistors within specifications selected largely from units received either just previous to or just after the 7000 used for the other distributions. The original 7000 transistors were not available for retest.

Fig. 24 shows the effect of collector current on the d-c beta distribution of the surface barrier transistor. Note that the median beta drops from about 25 at 1 ma to 4 at 50 ma. Also the tail at the high-gain end disappears at higher currents.

The measurement of hole storage is essentially a measurement of charge. There are two ways that this can be done: (1) by measuring the time required to remove the charge with a constant current. This is the τ_s measurement. (2) by measuring the resultant voltage when the charge is placed on a known capacity. This is the K_s' measurement. The latter method requires somewhat less elaborate equipment and will undoubtedly be used for large-scale testing.

Initial measurements of the K_s' distribution for the 50 selected transistors gave a median value of 88 mμsec with limits of 60 and 130 mμsec. Previous experience has shown that this is higher than expected for a good group of surface-barrier switching transistors. A new group of 72 transistors was selected to give a τ_s distribution similar to that in Fig. 41. The τ_s distribution for this group had a median value of 56 mμsec with limits at 30 and 80 mμsec. This is still somewhat above the values of τ_s obtained in the distribution for the 7000 transistors but is acceptably close. The K_s' distribution for this group of 72 transistors is given in Fig. 25. Other work has shown that K_s' should be approximately 1.5 times τ_s. Using this relationship, we can plot on the same graph an estimated distribution of K_s' for the 7000 transistors based on their τ_s distribution. This appears to give somewhat lower values. The median value should lie between 74 and 80 mμsec.

The circuit used to make the K_s' measurement is shown in Fig. 35. The initial current distribution in the transistor during saturation is different from that in the τ_s measurement but correlation between the two is very good.

Frequency response distributions obtained from h_{FE} measurements are shown in Fig. 26. h_{FE} is the common-emitter current gain (at 10 mc in this case). fh_{FE} or f_h is the frequency where $\beta = 1$ or approximately the cutoff frequency. A more satisfactory measurement for the inverse case could be made at 5 mc. The test circuit is shown in Fig. 40. Note that the median forward f_h is about 3 times the median inverse frequency.

The forward and inverse $r_b'C_c$ distributions are shown in Fig. 27. A sketch of the a-c test circuit used appears in Fig. 41.

A distribution of the output capacity values is given in Fig. 28. A separate measurement gave about 1 µµf as the capacity of the header and can. With this value subtracted from C_o, the dotted C_c distribution results. The test circuit is shown in Fig. 42. The results obtained with 1 ma emitter current were essentially the same.

The r_b' distribution presented in Fig. 29 was obtained by dividing the C_c values into the measured values of $r_b'C_c$.

The $I_{c\ sat}$ distribution for 0.6 ma base current and 150 mv V_{ce} is shown in Fig. 30 and was obtained using the circuit of Fig. 39. This is a test of interest in circuit design.

The data contained in these distributions can be summarized in the following table:

Parameter	Lower Value	Median Value	Upper Value
K_s' µcoul/ma I_b or mµsec	50	80	160
f_{hF} mc	30	53.5	80
f_{hI} mc	8	18	40
$r_b'C_c$ F µµsec	450	680	1000
$r_b'C_c$ I µµsec	340	470	800
C_o µµf	2.9	3.7	4.8
r_b' ohms	150	255	435
$I_{c\ sat}$ ma (0.6 ma (150 mv	4.8	6.6	9.5

D. J. Eckl

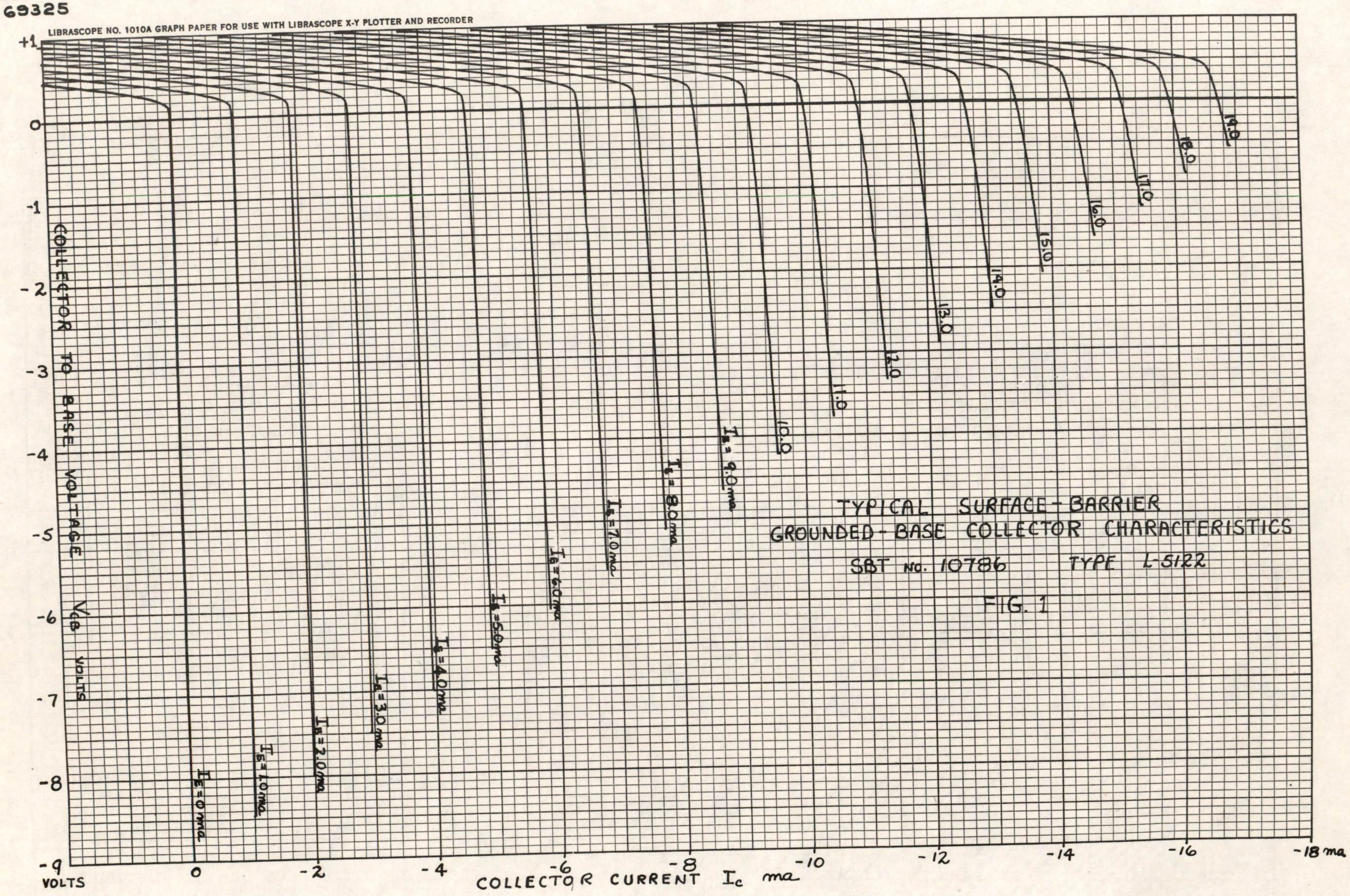
Donald J. Eckl

DJE/md

LIST OF ILLUSTRATIONS

B-69325

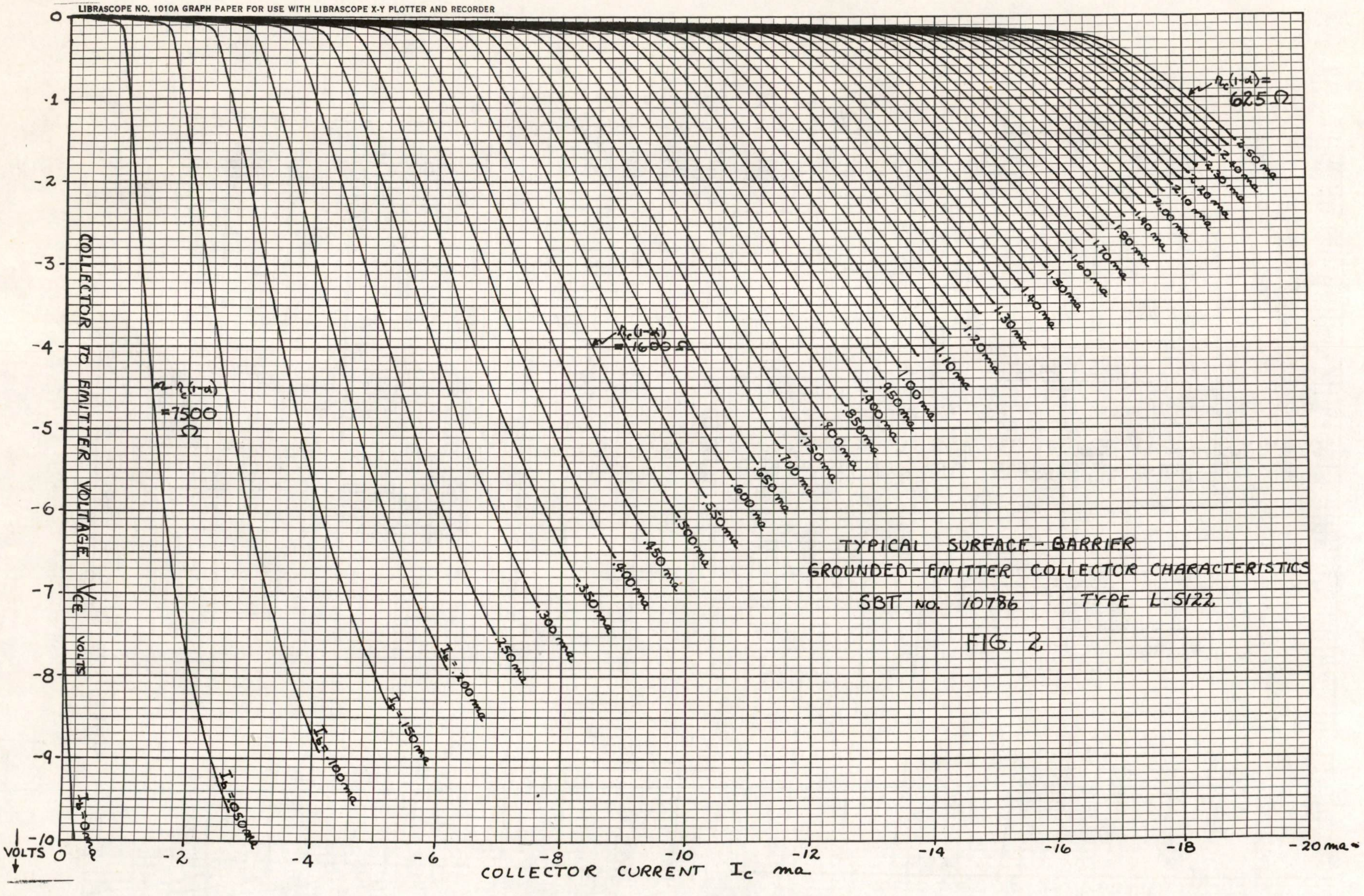
LIBRASCOPE NO. 1010A GRAPH PAPER FOR USE WITH LIBRASCOPE X-Y PLOTTER AND RECORDER



TYPICAL SURFACE-BARRIER
GROUNDED-BASE COLLECTOR CHARACTERISTICS
SBT No. 10786 TYPE L-5122

FIG. 1

B-69326



TYPICAL SURFACE-BARRIER
GROUNDED-EMITTER COLLECTOR CHARACTERISTICS
SBT NO. 10786 TYPE L-5122

FIG. 2

B-69327

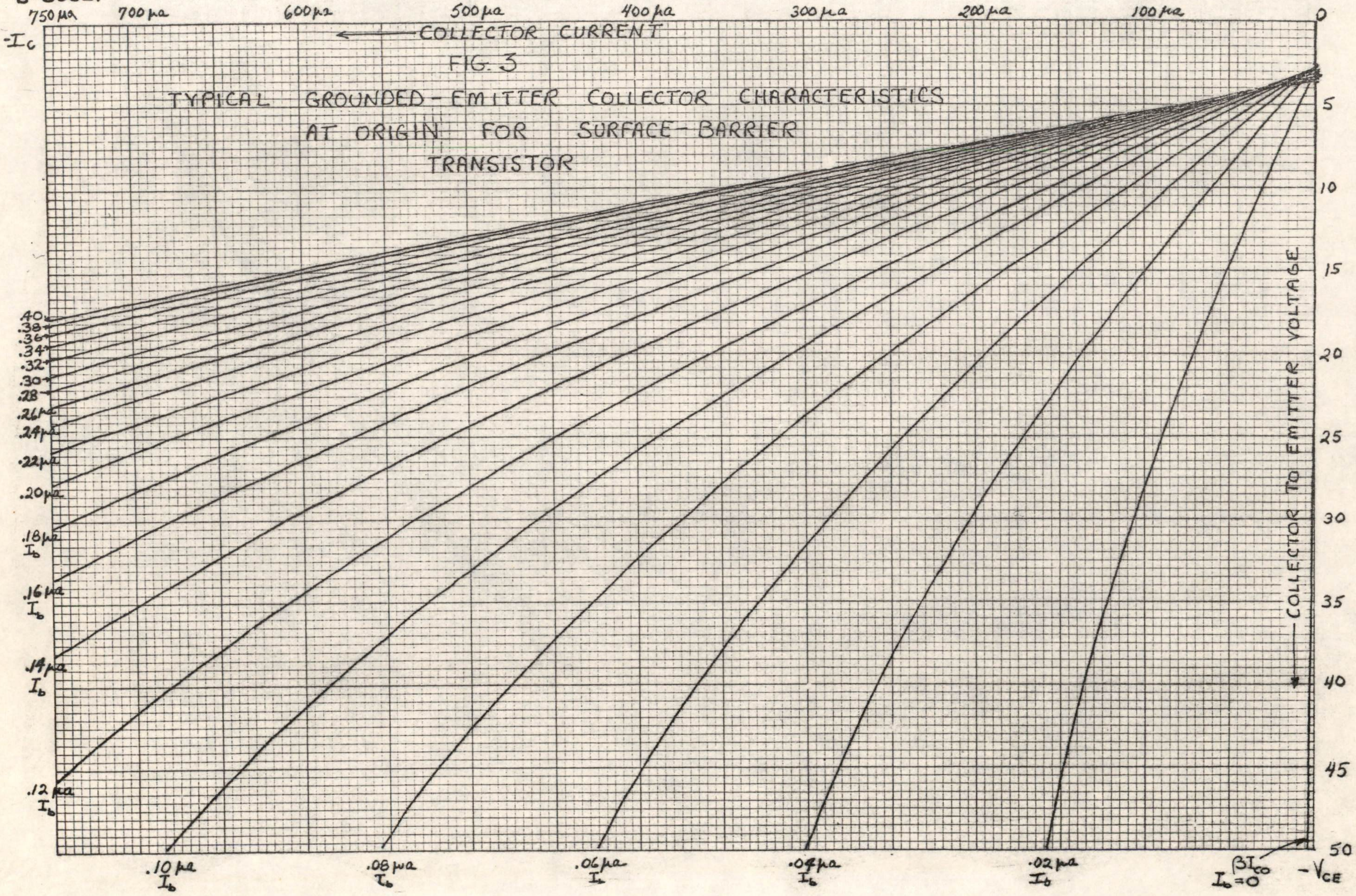
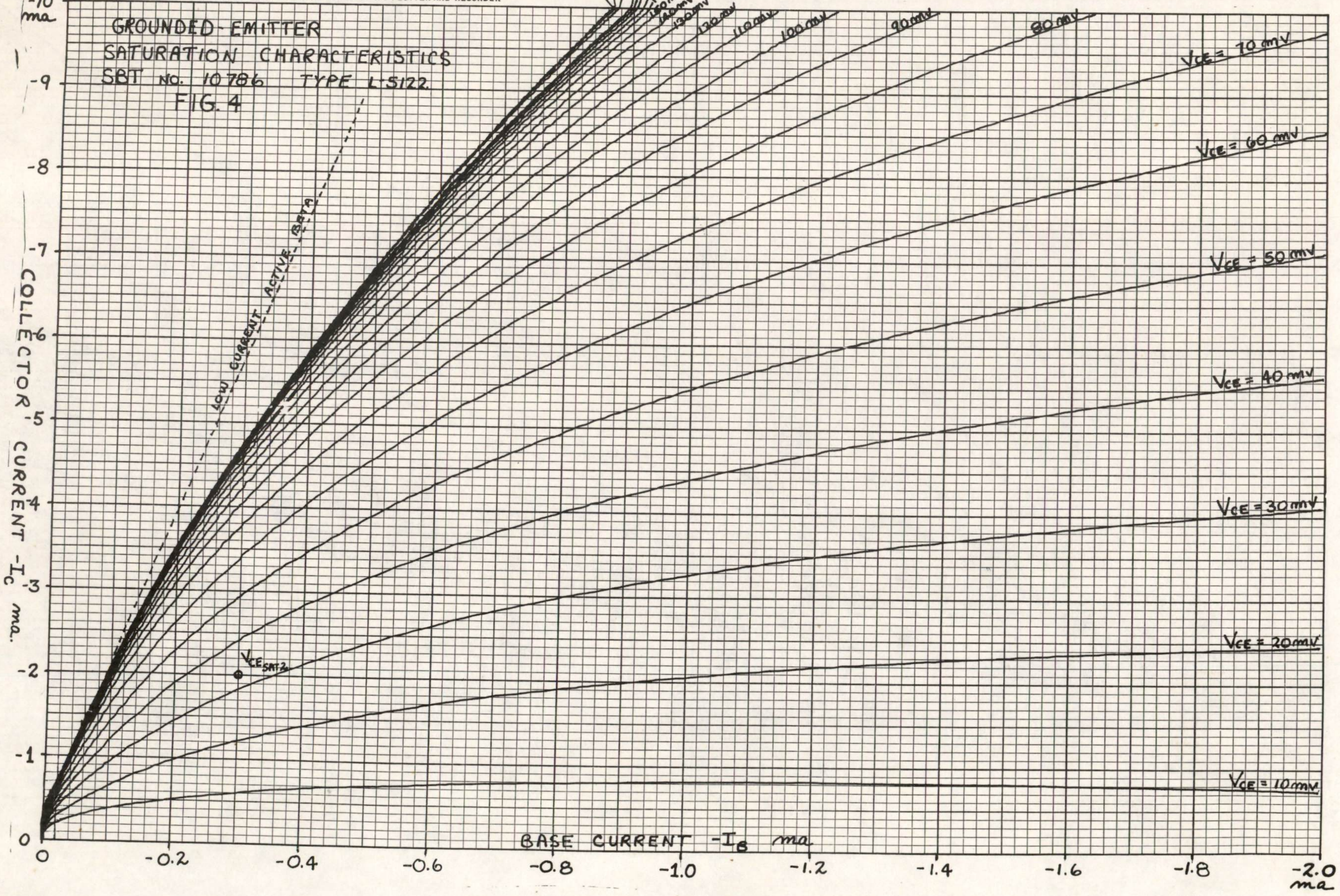


FIG. 3

TYPICAL GROUNDED-EMITTER COLLECTOR CHARACTERISTICS AT ORIGIN FOR SURFACE-BARRIER TRANSISTOR

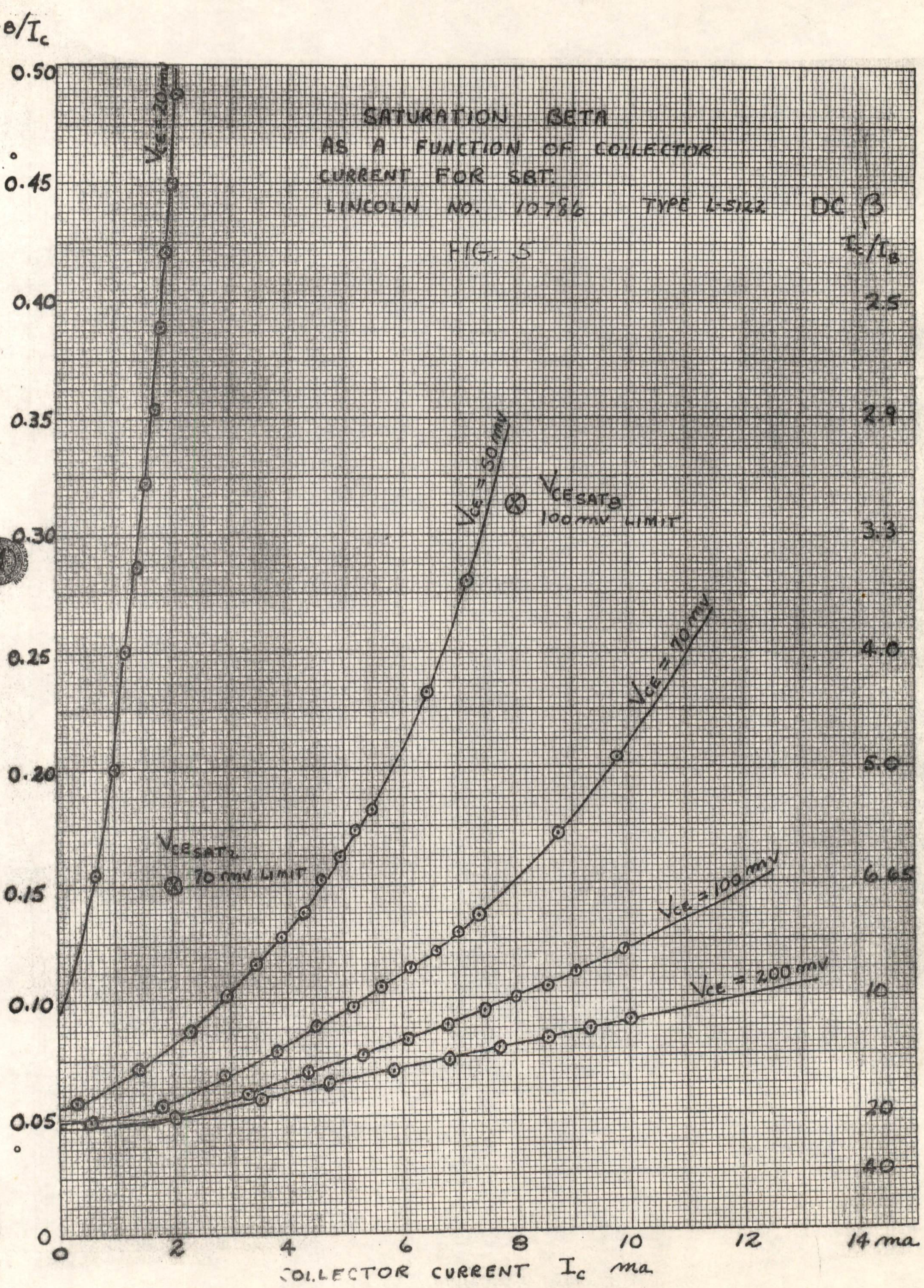
B-69328



GROUNDED-EMITTER SATURATION CHARACTERISTICS

SBT NO. 10786 TYPE L-5122

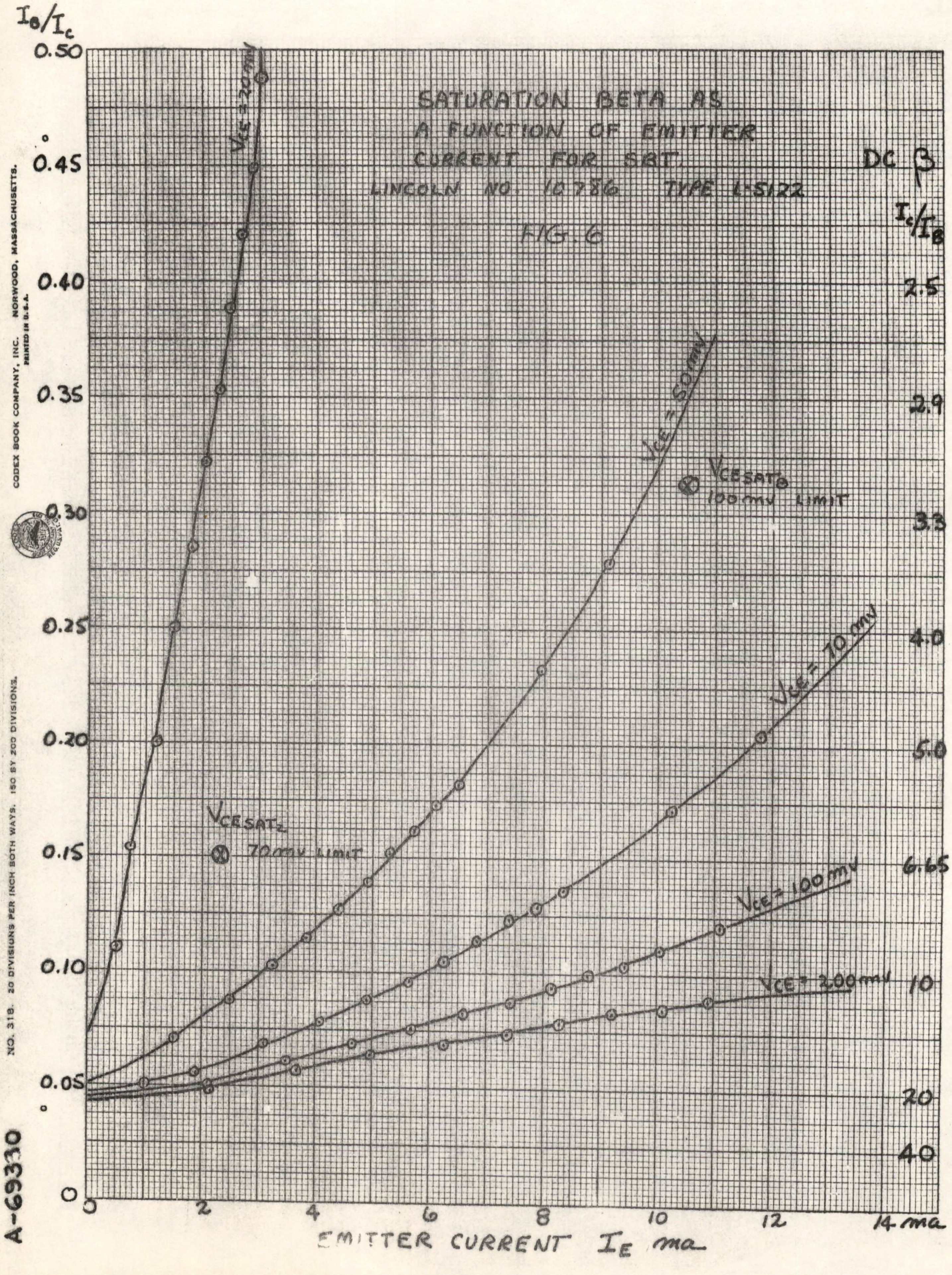
FIG. 4



SATURATION BETA AS A FUNCTION OF COLLECTOR CURRENT FOR SBT. LINCOLN NO. 10786 TYPE L-5122

FIG. 5

A-69329

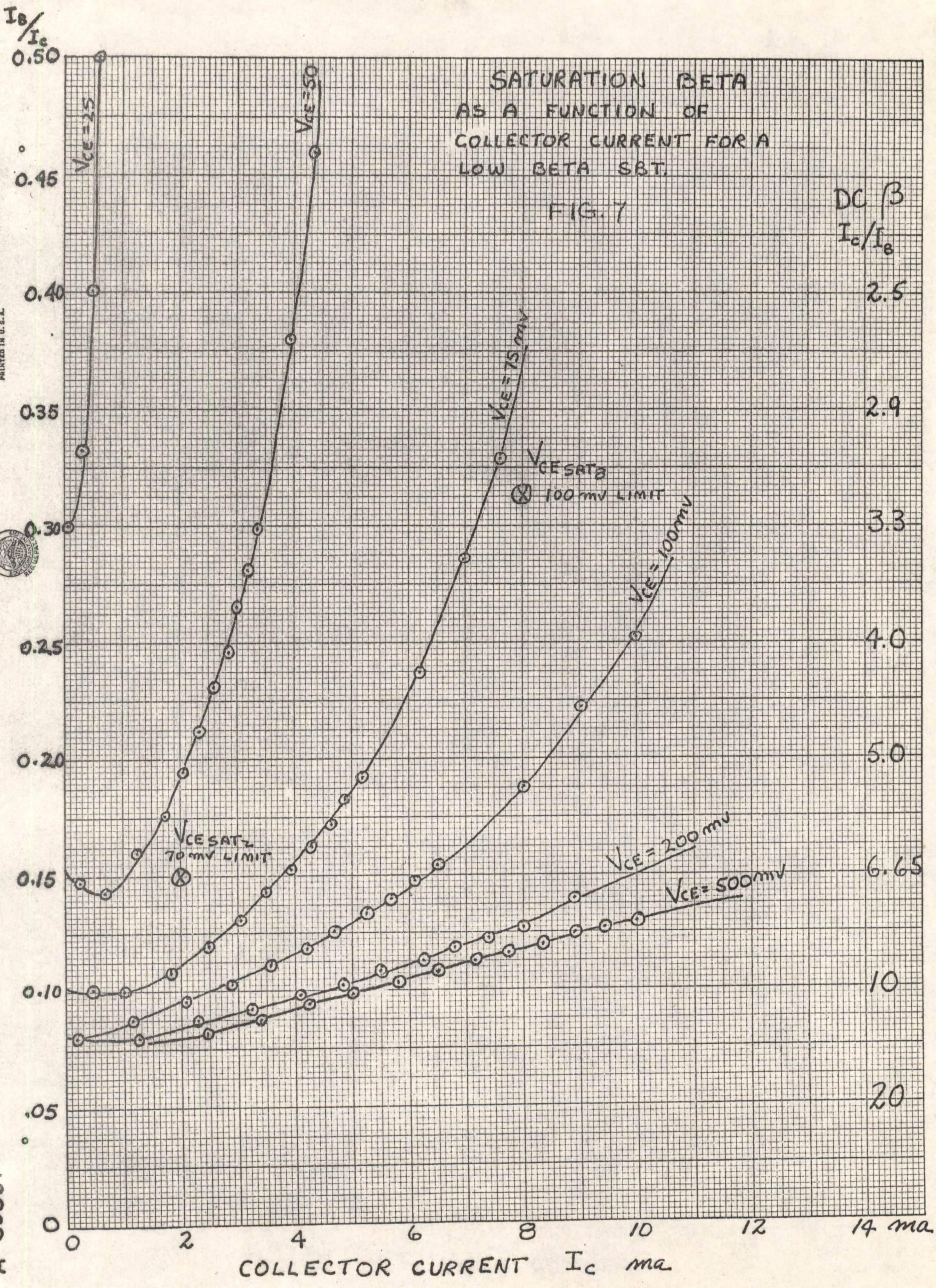


SATURATION BETA AS A FUNCTION OF EMITTER CURRENT FOR SRT.

LINCOLN NO. 10786 TYPE L-5122

FIG. 6

A-69330



SATURATION BETA AS A FUNCTION OF COLLECTOR CURRENT FOR A LOW BETA SBT.

FIG. 7

A-69331

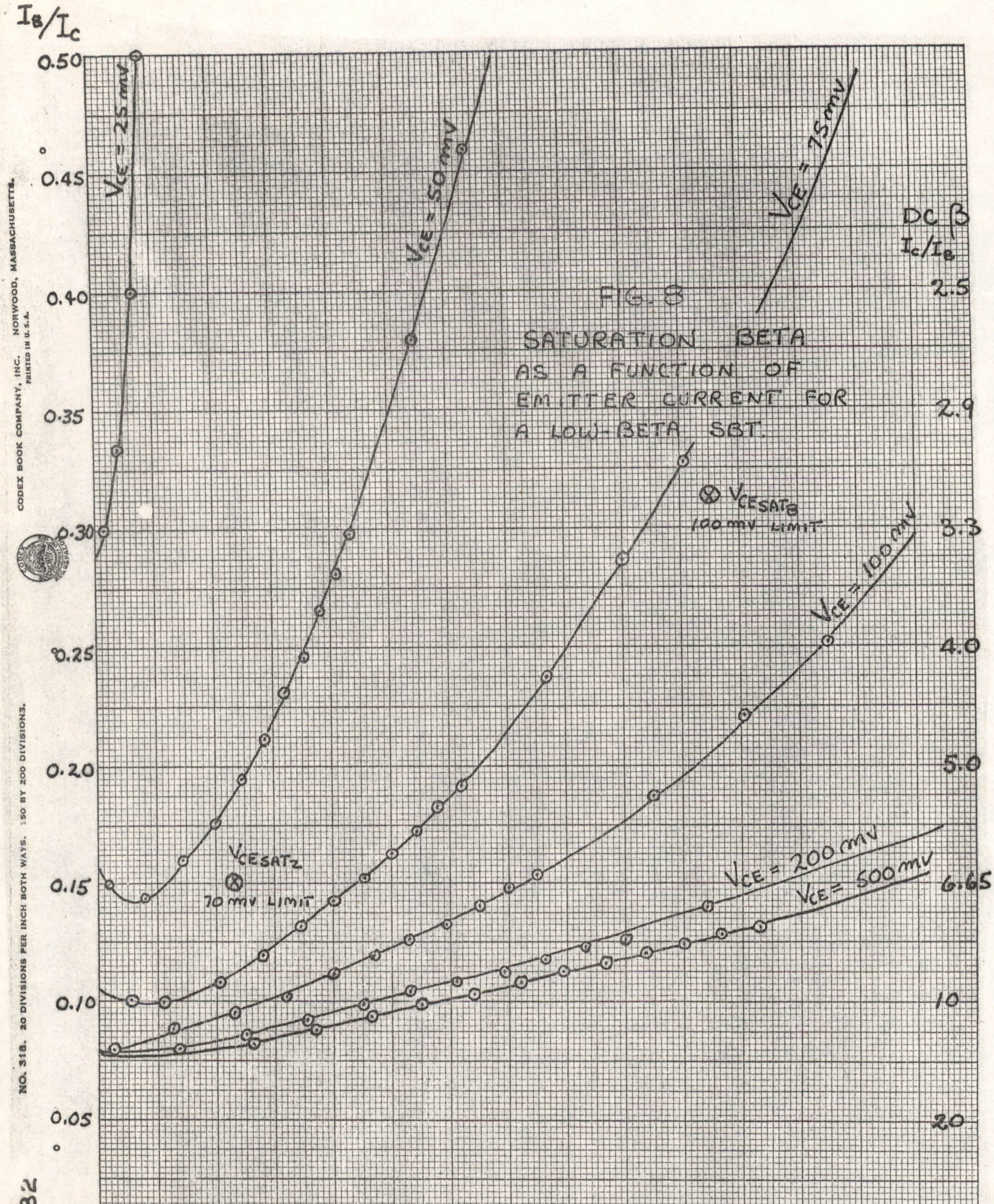
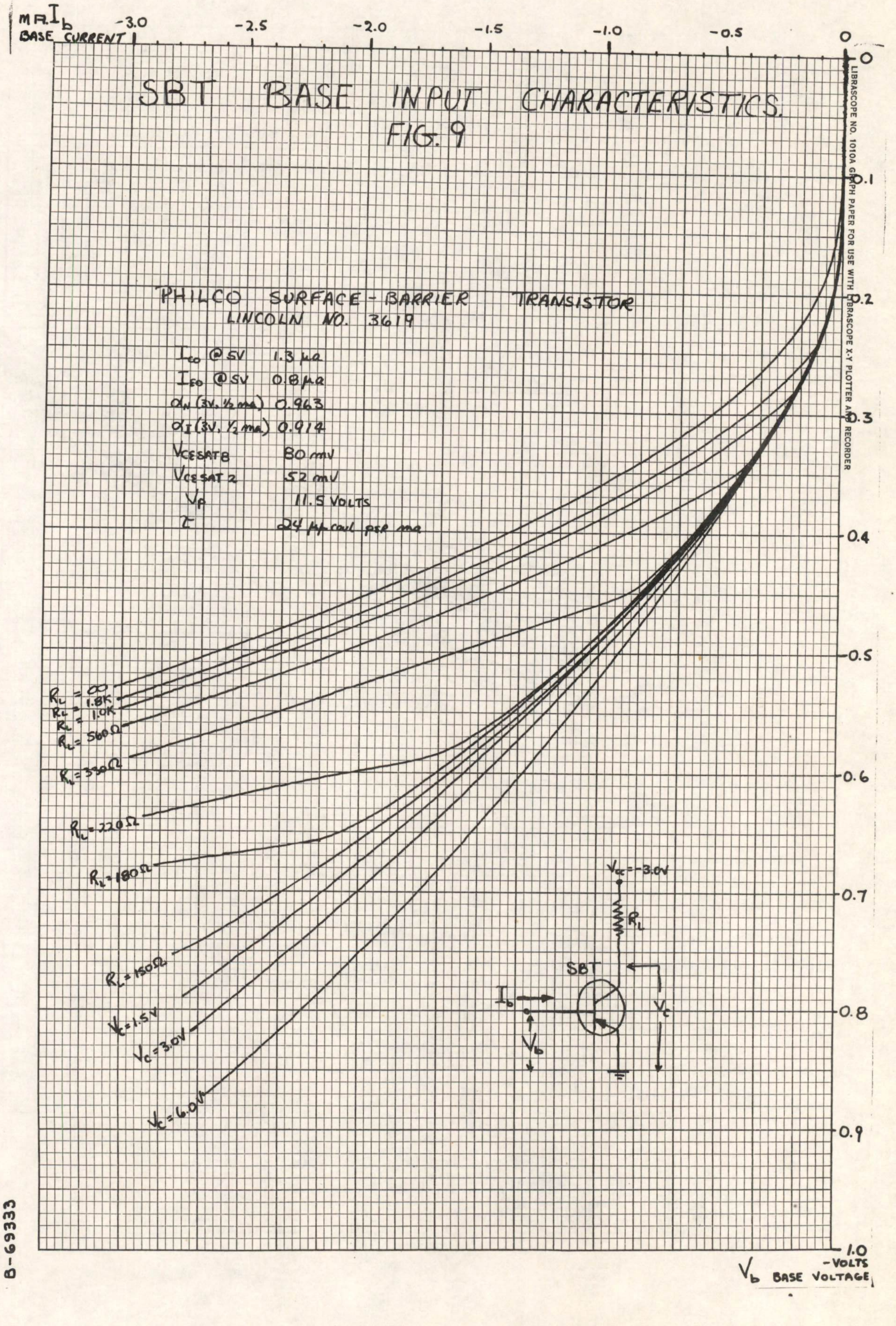


FIG. 8

SATURATION BETA AS A FUNCTION OF EMITTER CURRENT FOR A LOW-BETA SBT.

A-69332

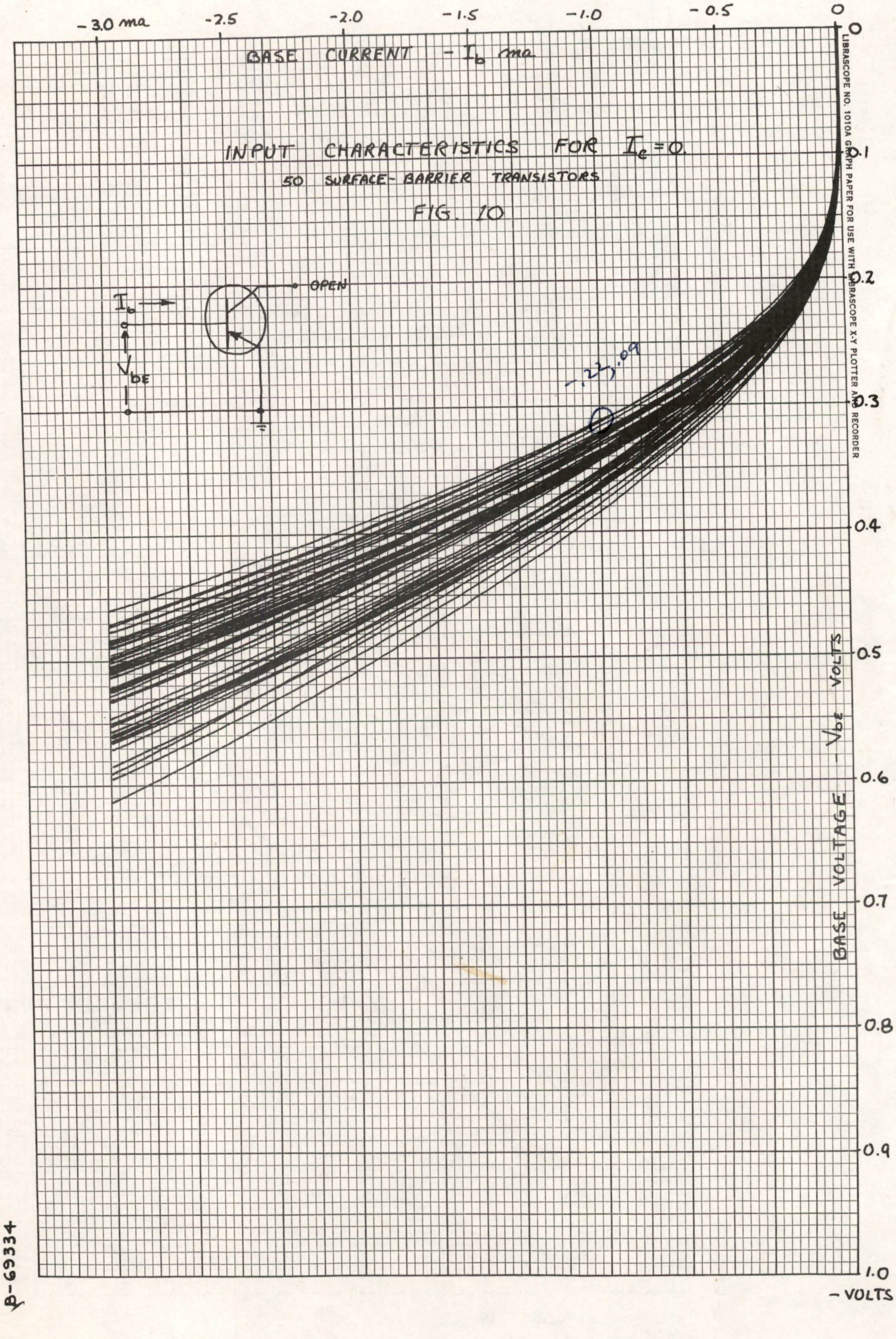


SBT BASE INPUT CHARACTERISTICS.

FIG. 9

PHILCO SURFACE-BARRIER TRANSISTOR
LINCOLN NO. 3619

I_{CO} @ 5V	1.3 μa
I_{EO} @ 5V	0.8 μa
α_N (3V, ½ ma)	0.963
α_I (3V, ½ ma)	0.914
$V_{CE SAT 8}$	80 mV
$V_{CE SAT 2}$	52 mV
V_P	11.5 VOLTS
τ	24 μμ coul per ma

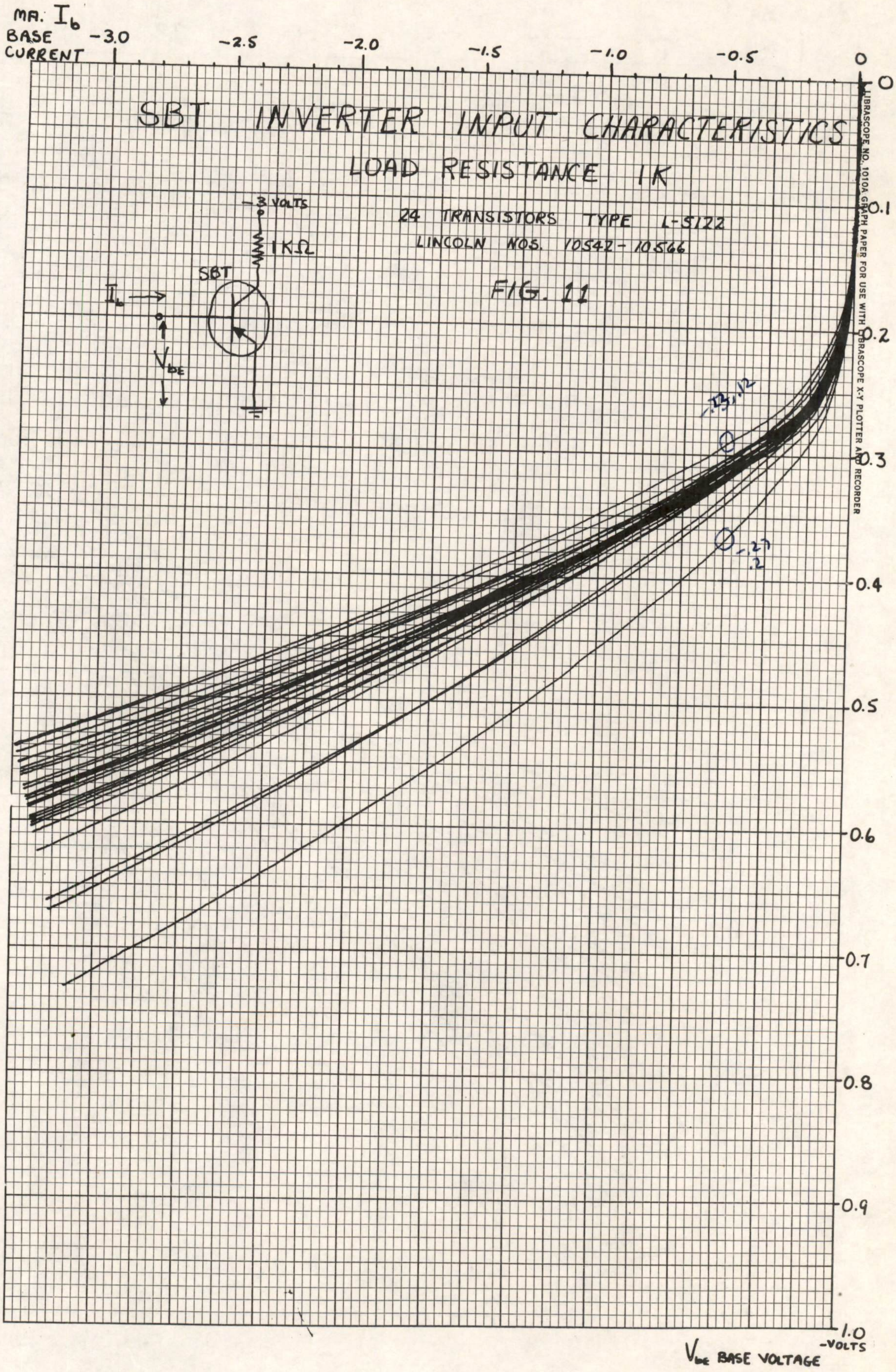


INPUT CHARACTERISTICS FOR $I_c = 0$.

50 SURFACE-BARRIER TRANSISTORS

FIG. 10

B-69334



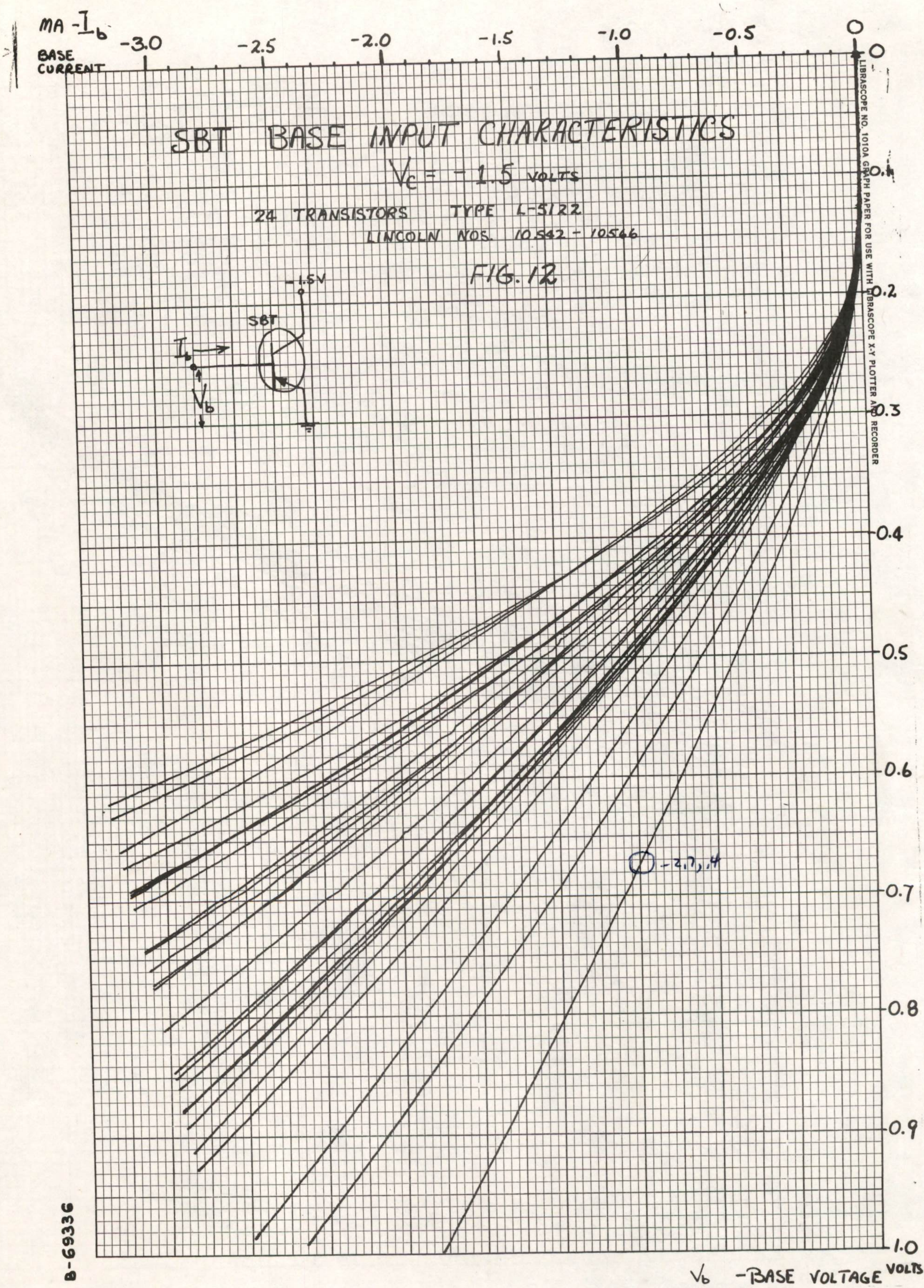
SBT INVERTER INPUT CHARACTERISTICS

LOAD RESISTANCE 1K

24 TRANSISTORS TYPE L-5122

LINCOLN NOS. 10542-10566

FIG. 11



SBT BASE INPUT CHARACTERISTICS

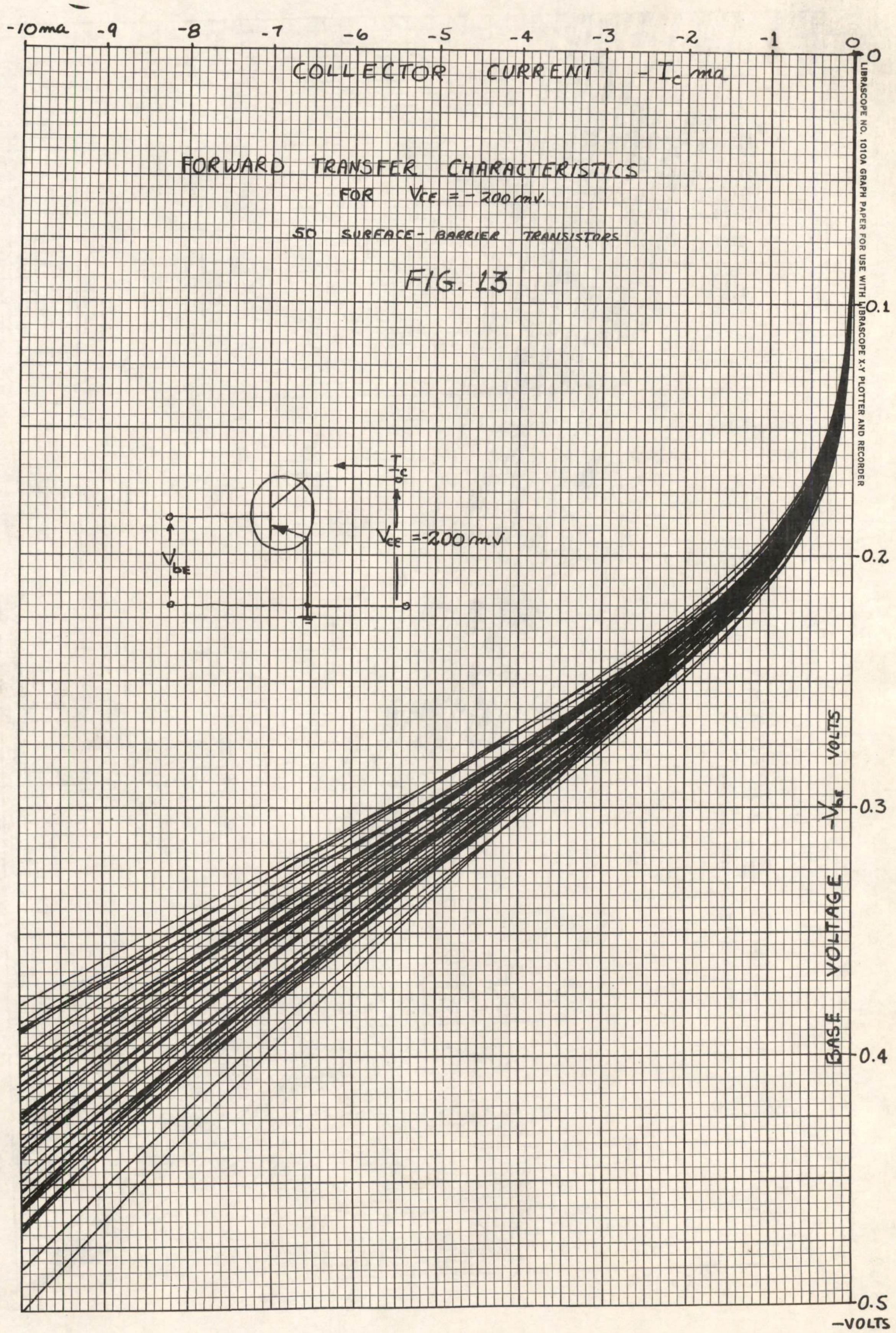
V_c = -1.5 VOLTS

24 TRANSISTORS TYPE L-5122

LINCOLN NOS. 10542 - 10566

FIG. 12

B-69336

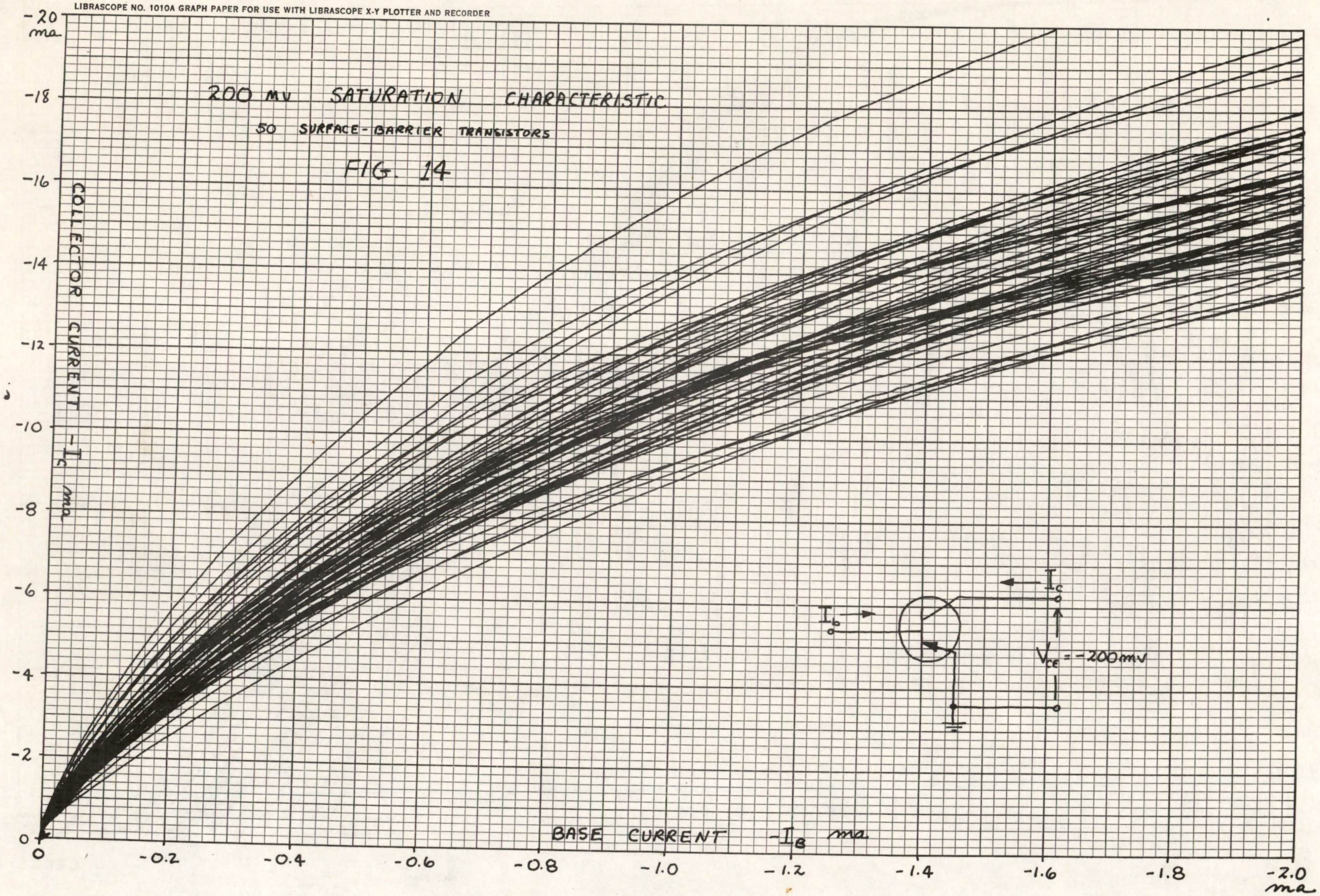


FORWARD TRANSFER CHARACTERISTICS
FOR $V_{CE} = -200$ mV.

50 SURFACE-BARRIER TRANSISTORS

FIG. 13

B-69338



200 MV SATURATION CHARACTERISTIC

50 SURFACE-BARRIER TRANSISTORS

FIG. 14

A-69339

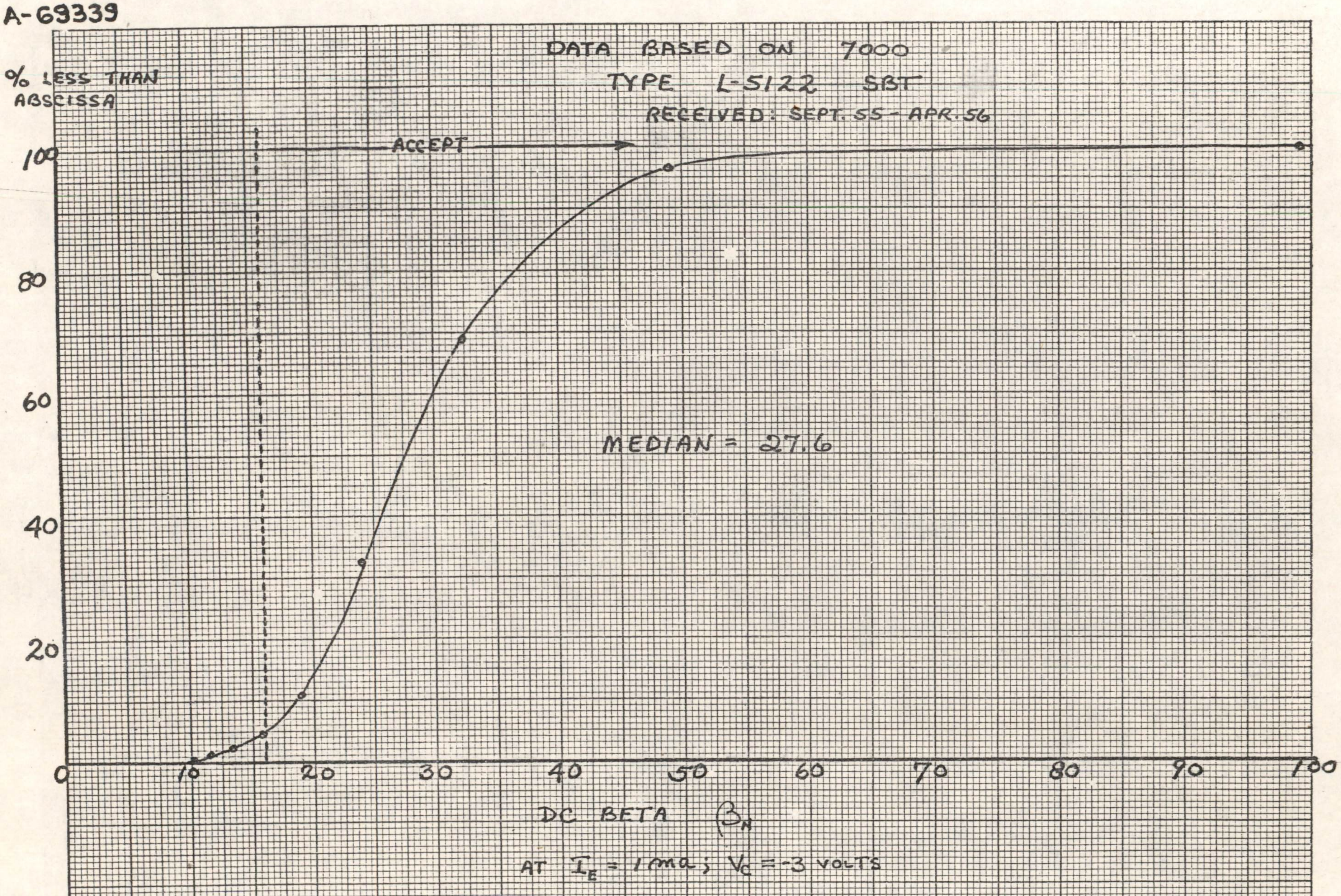
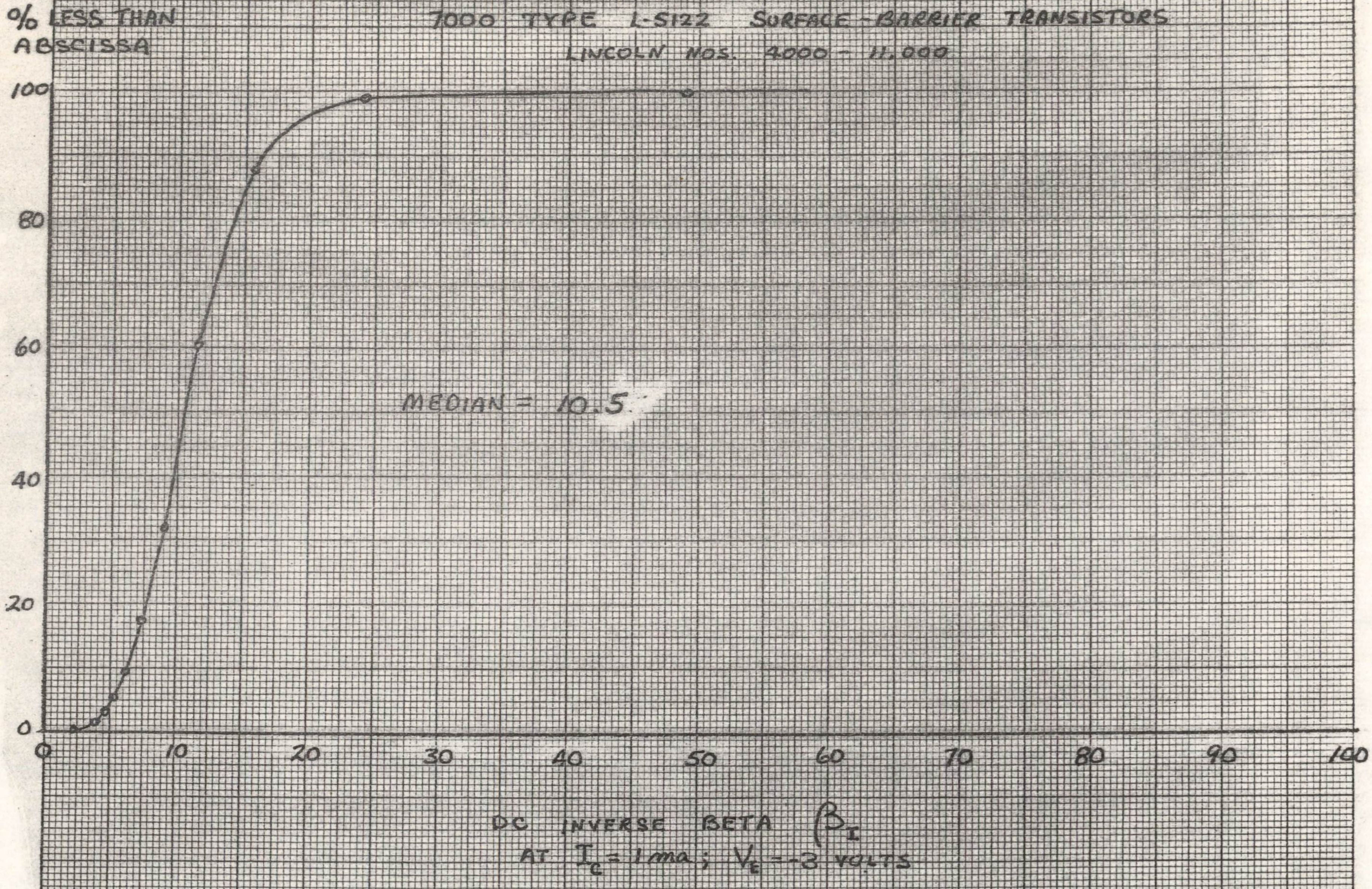


FIG. 15

A-69340



7000 TYPE L-5122 SURFACE-BARRIER TRANSISTORS
LINCOLN NOS. 4000 - 11,000

FIG. 16

A-69341

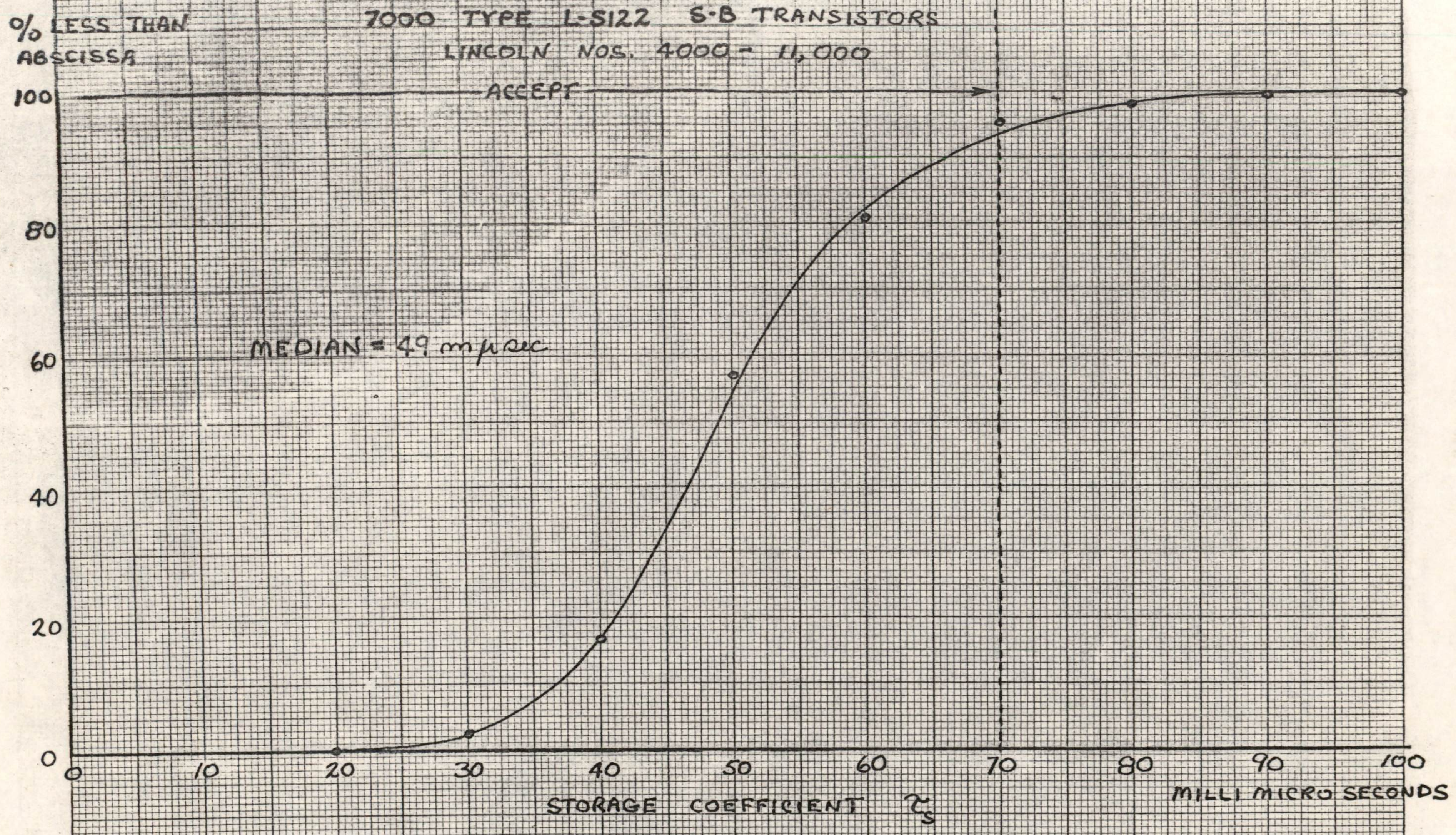


FIG. 17

A-69342

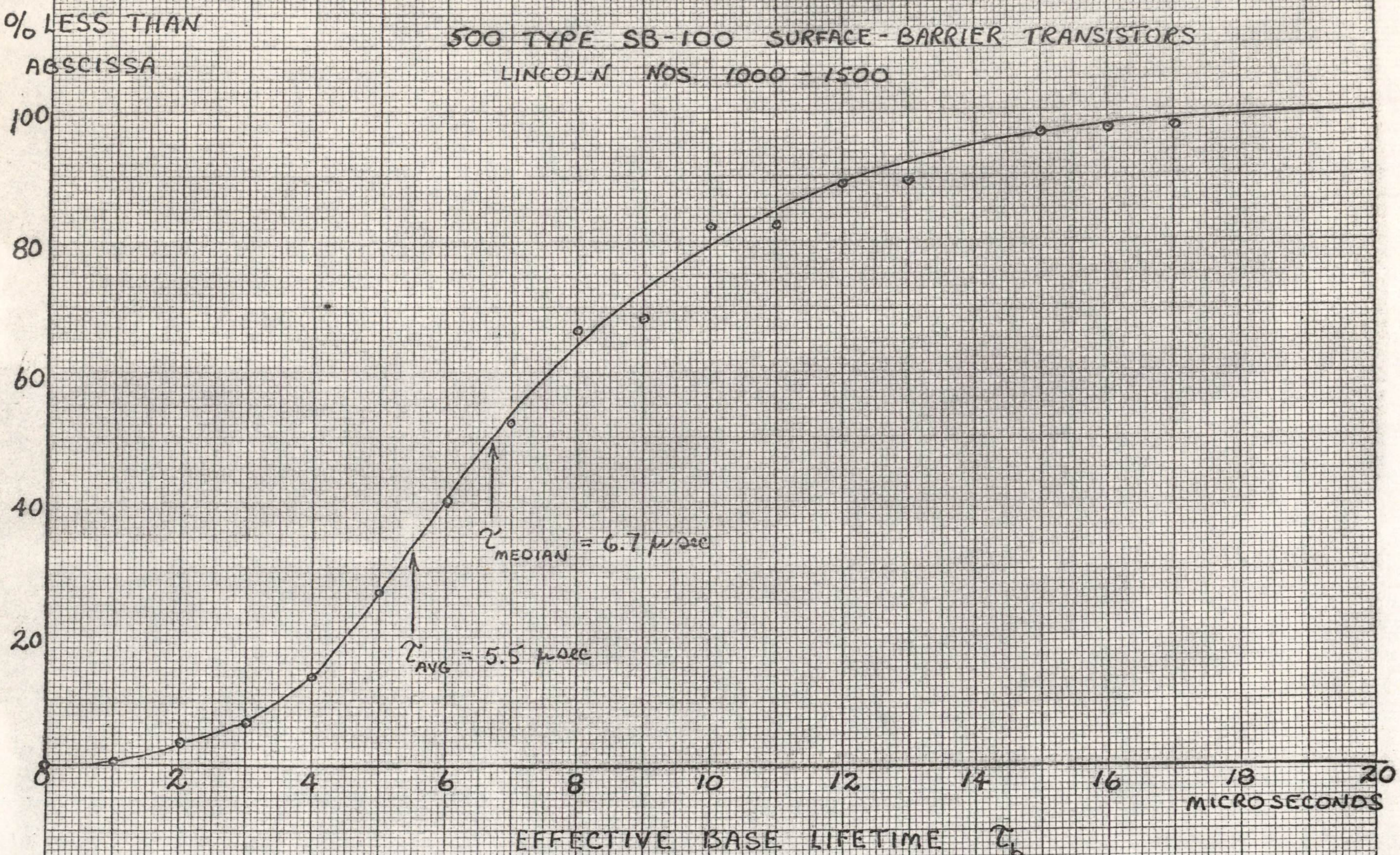


FIG. 18

A-65343

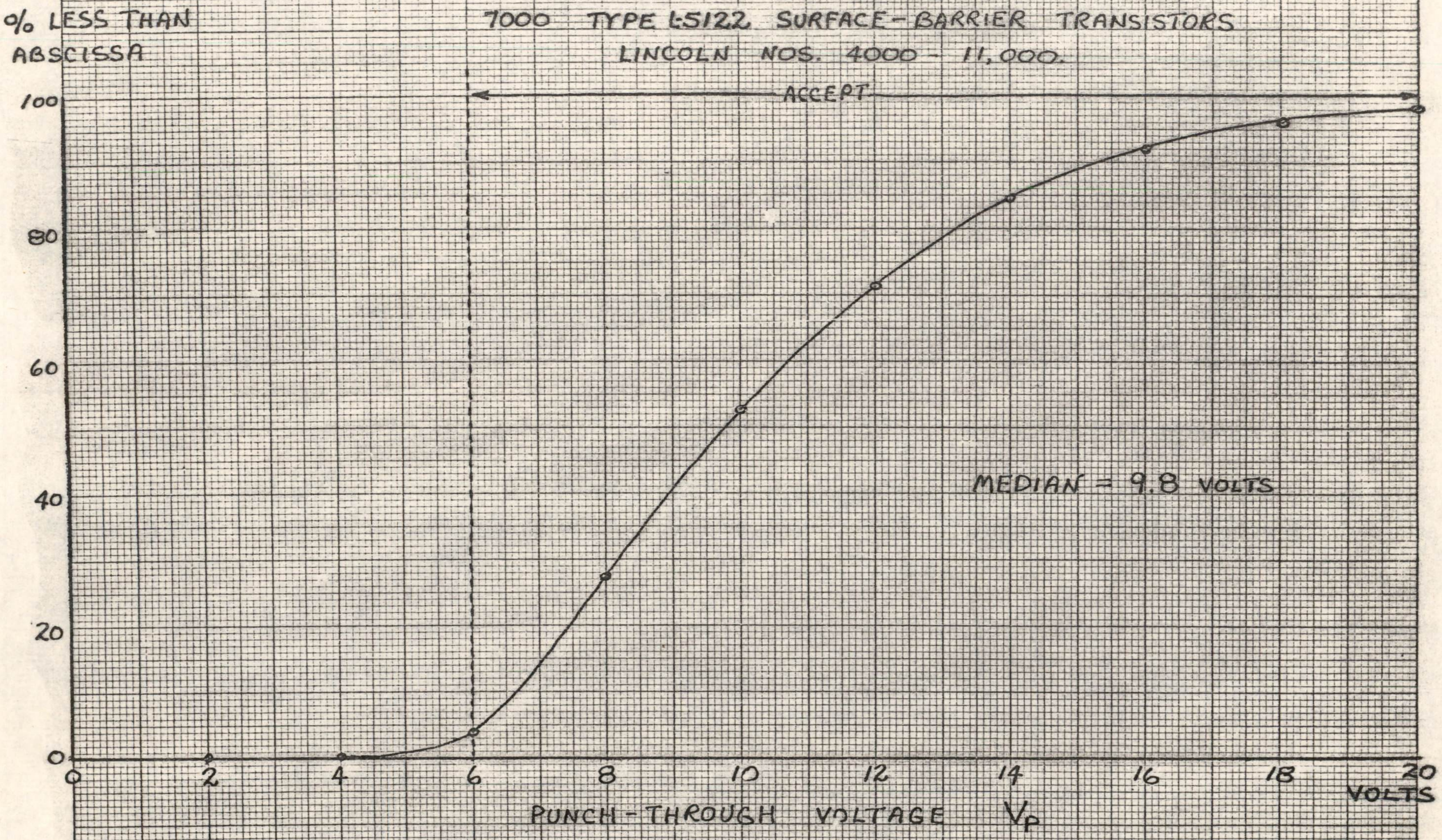
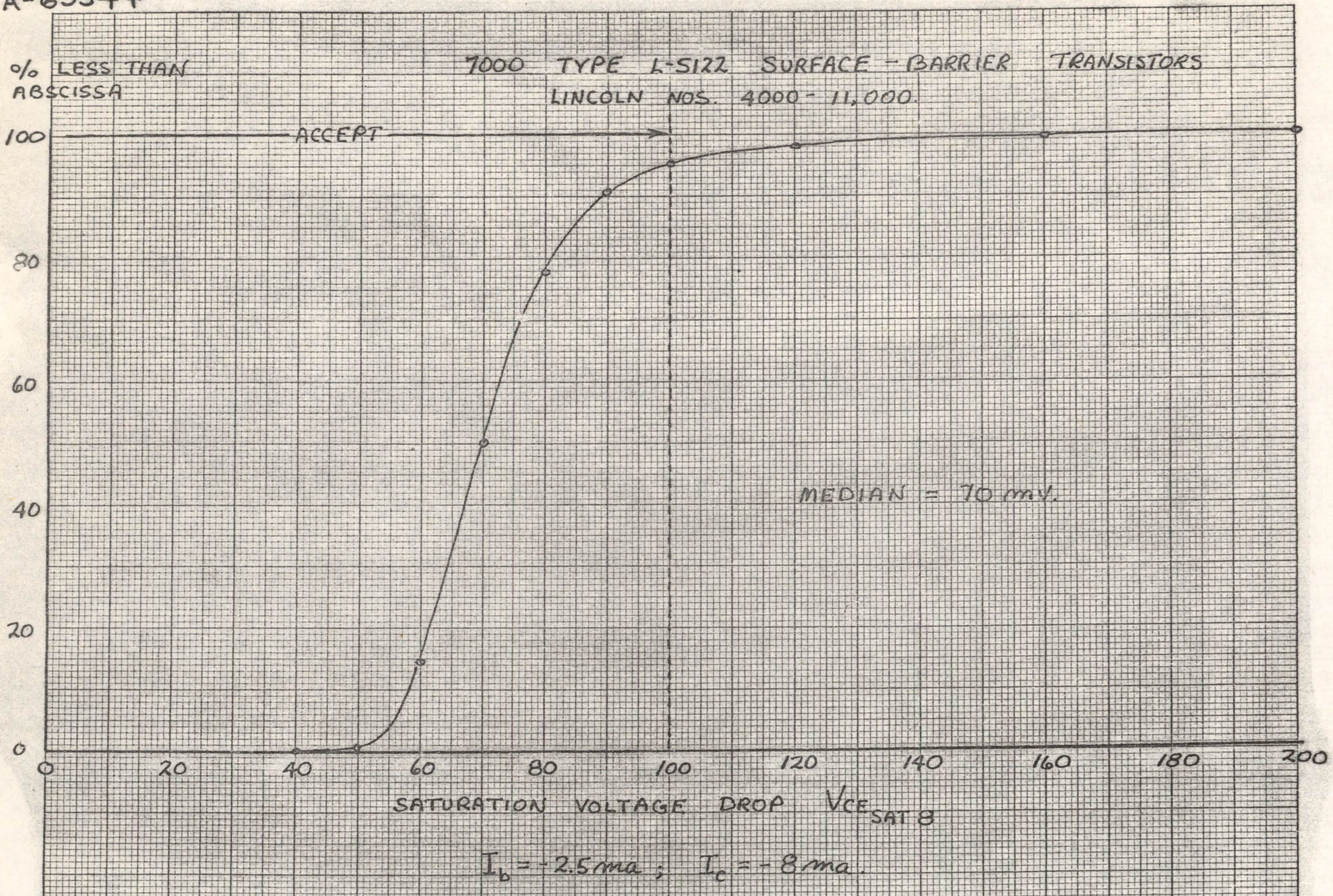


FIG. 19

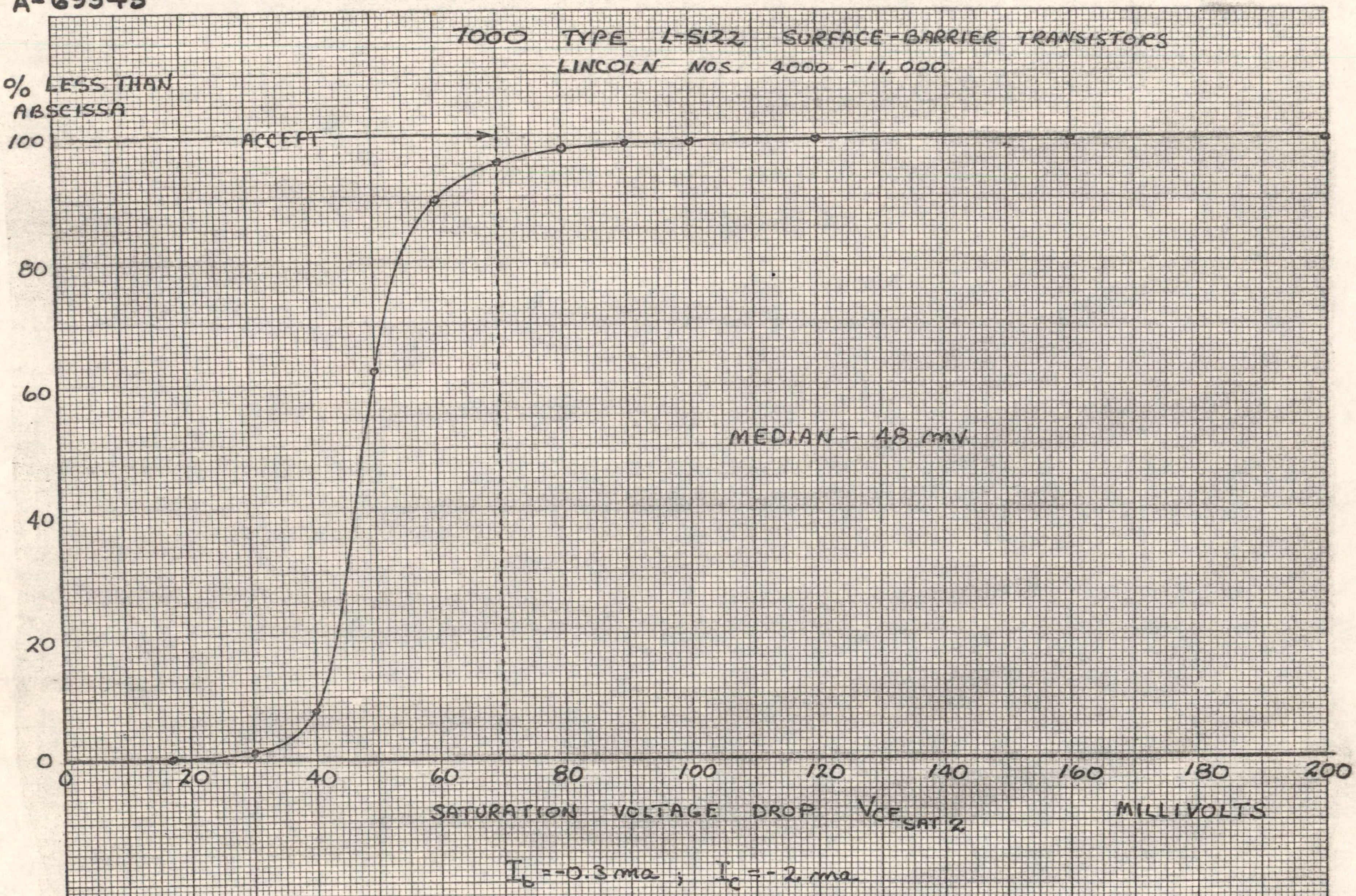
A-69344



7000 TYPE L-5122 SURFACE-BARRIER TRANSISTORS
LINCOLN NOS. 4000-11,000.

FIG. 20

A-69345



7000 TYPE L-5122 SURFACE-BARRIER TRANSISTORS
LINCOLN NOS. 4000 - 11,000

FIG 21

A-69346

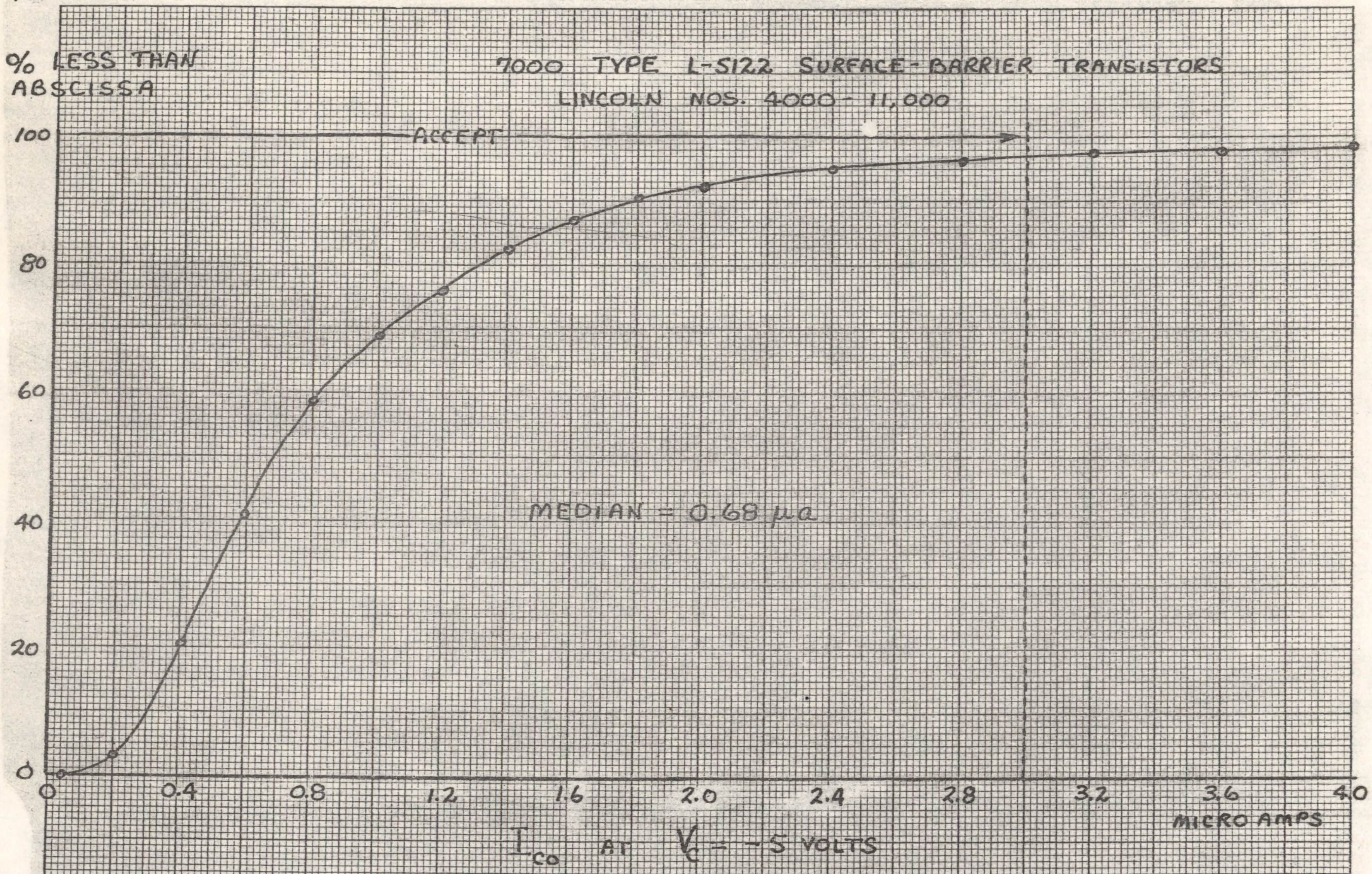
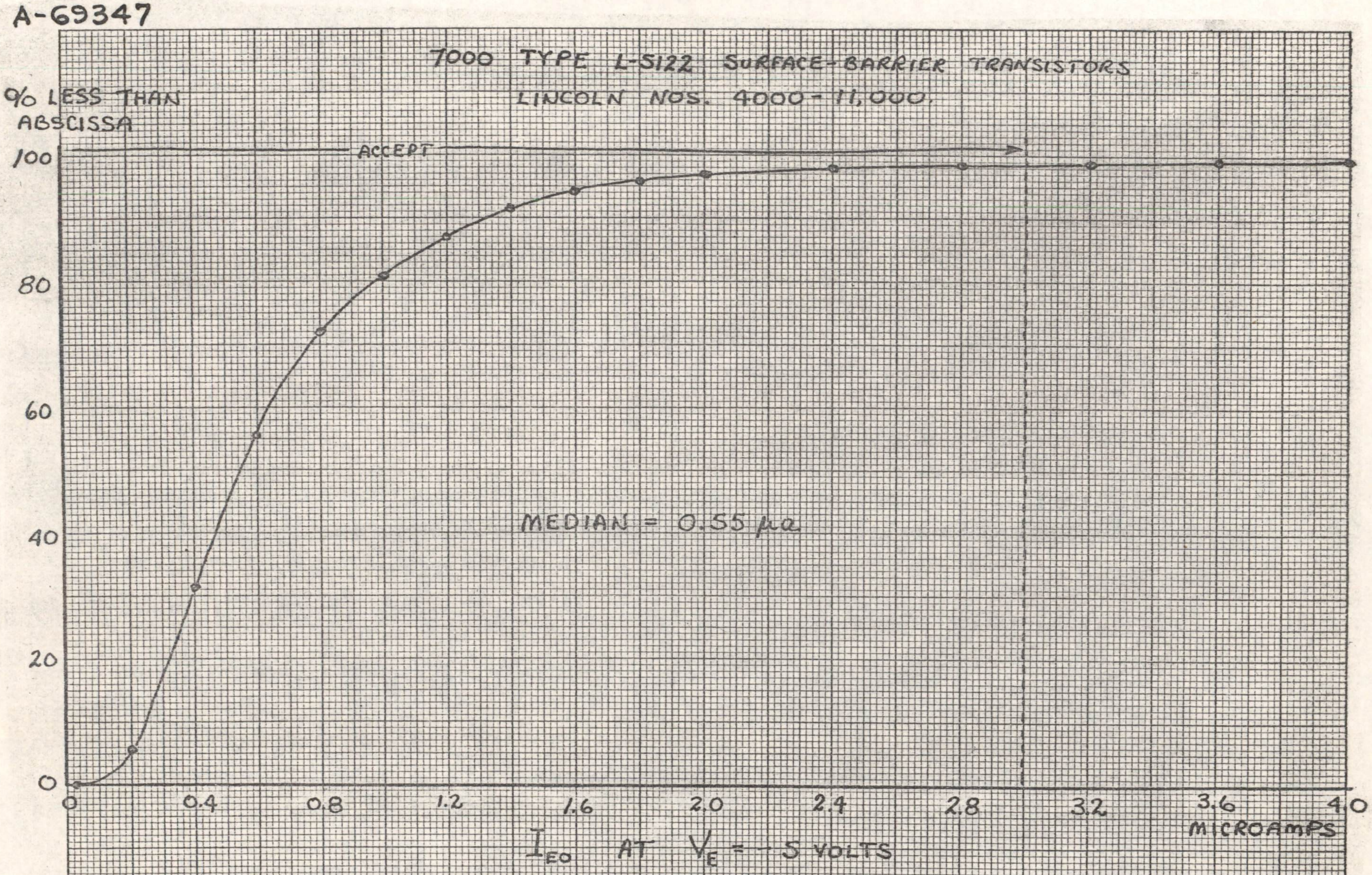


FIG. 22

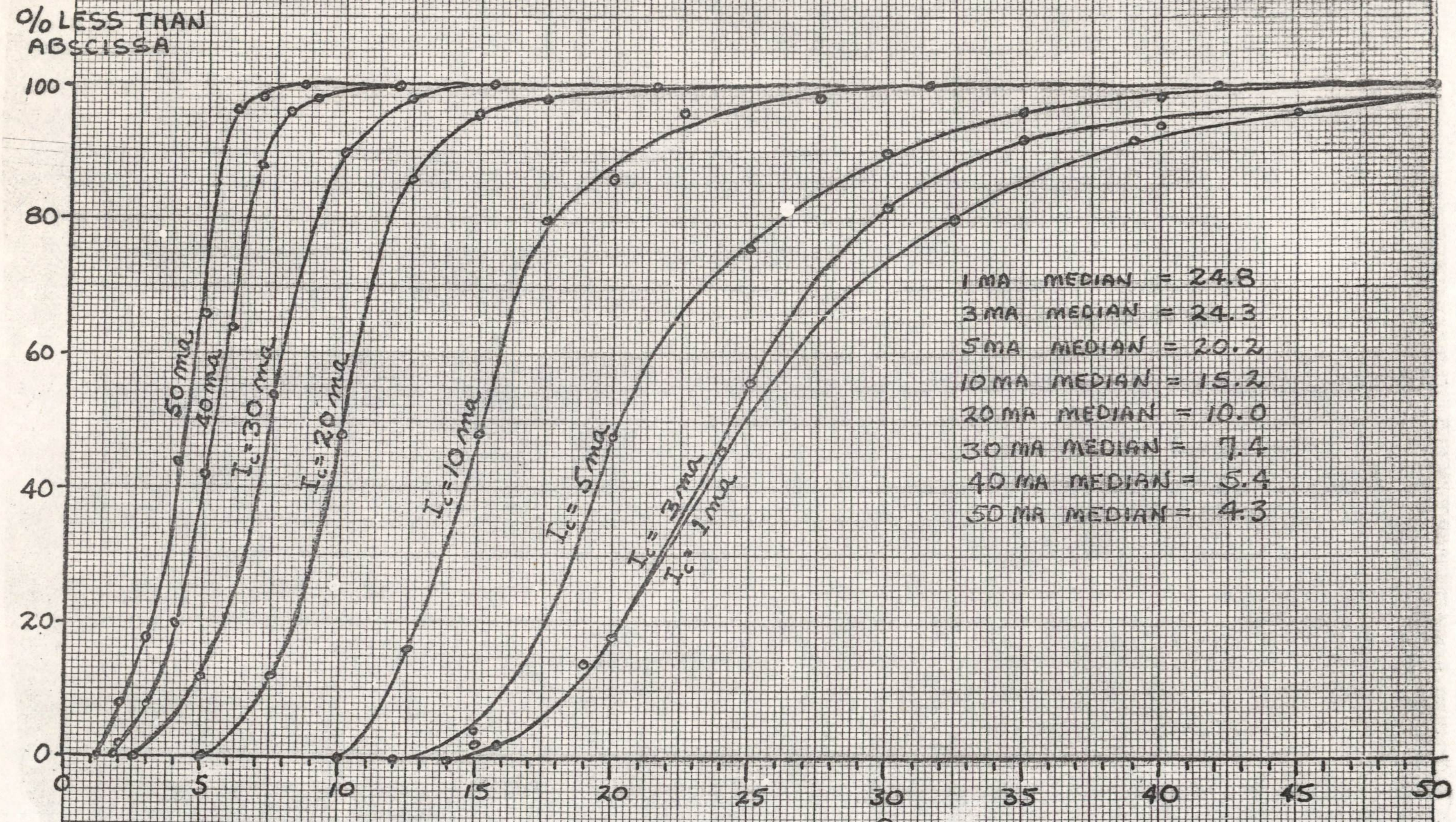
A-69347



7000 TYPE L-5122 SURFACE-BARRIER TRANSISTORS
LINCOLN NOS. 4000-11,000.

FIG. 23

A-69343



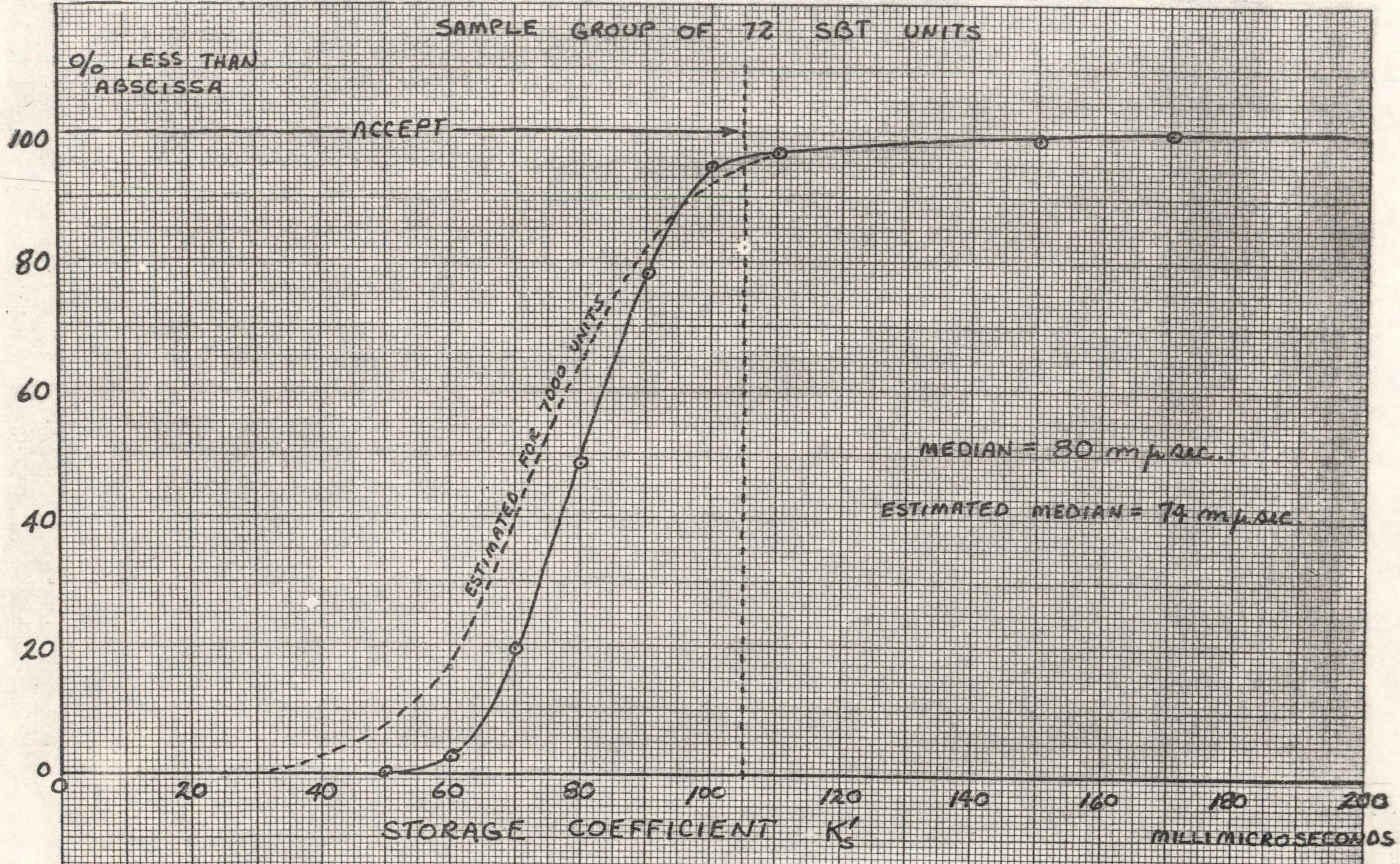
DC FORWARD β AT

VARIOUS COLLECTOR CURRENTS

50 SGT MEASURED AT $V_{CE} \cong 3$ VOLTS.

FIG. 24

A-69349



NOTE : $K_s' \approx 1.5\, \tau_s$

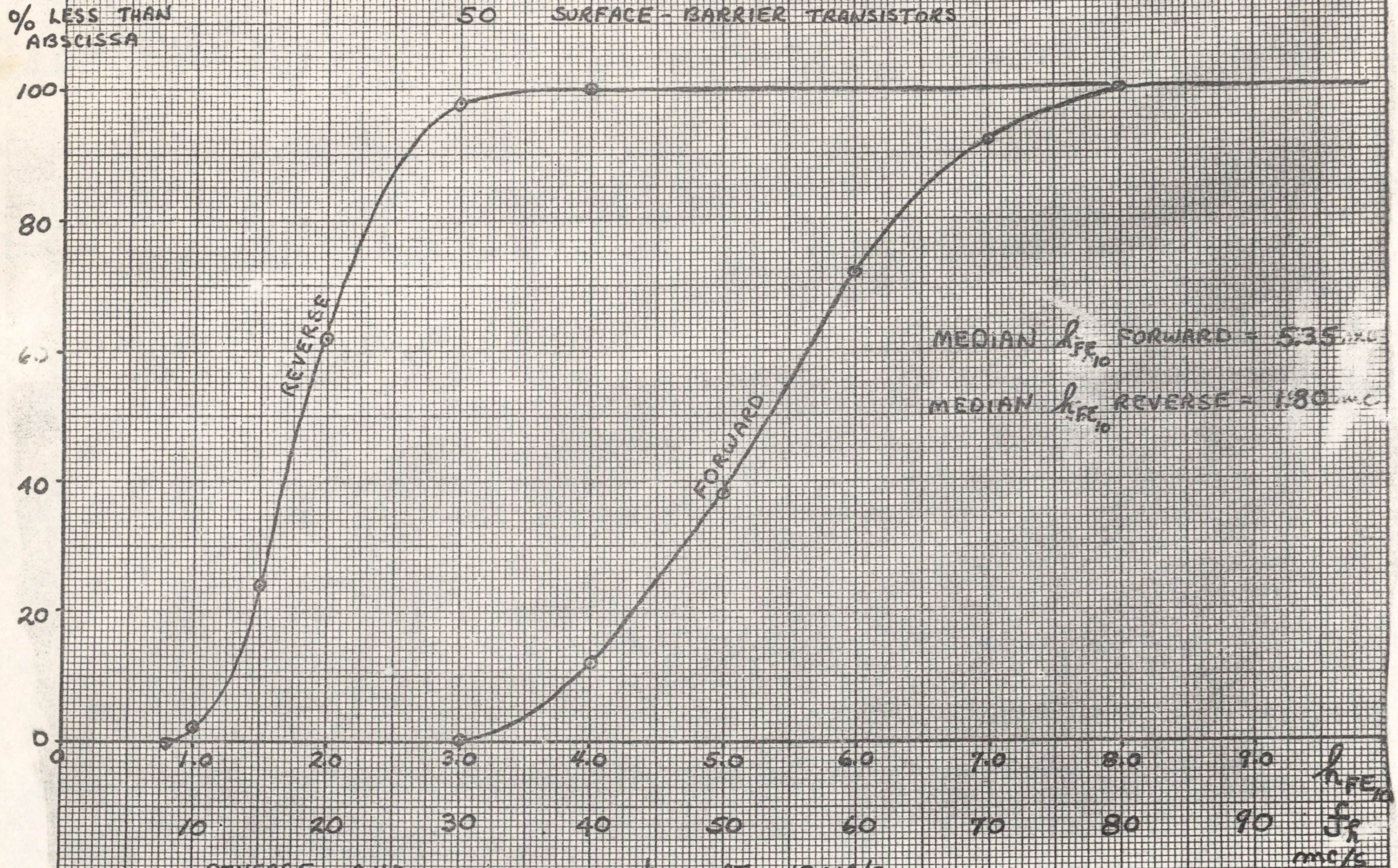
FIG. 25

A-69349

A-69350

FIG. 26

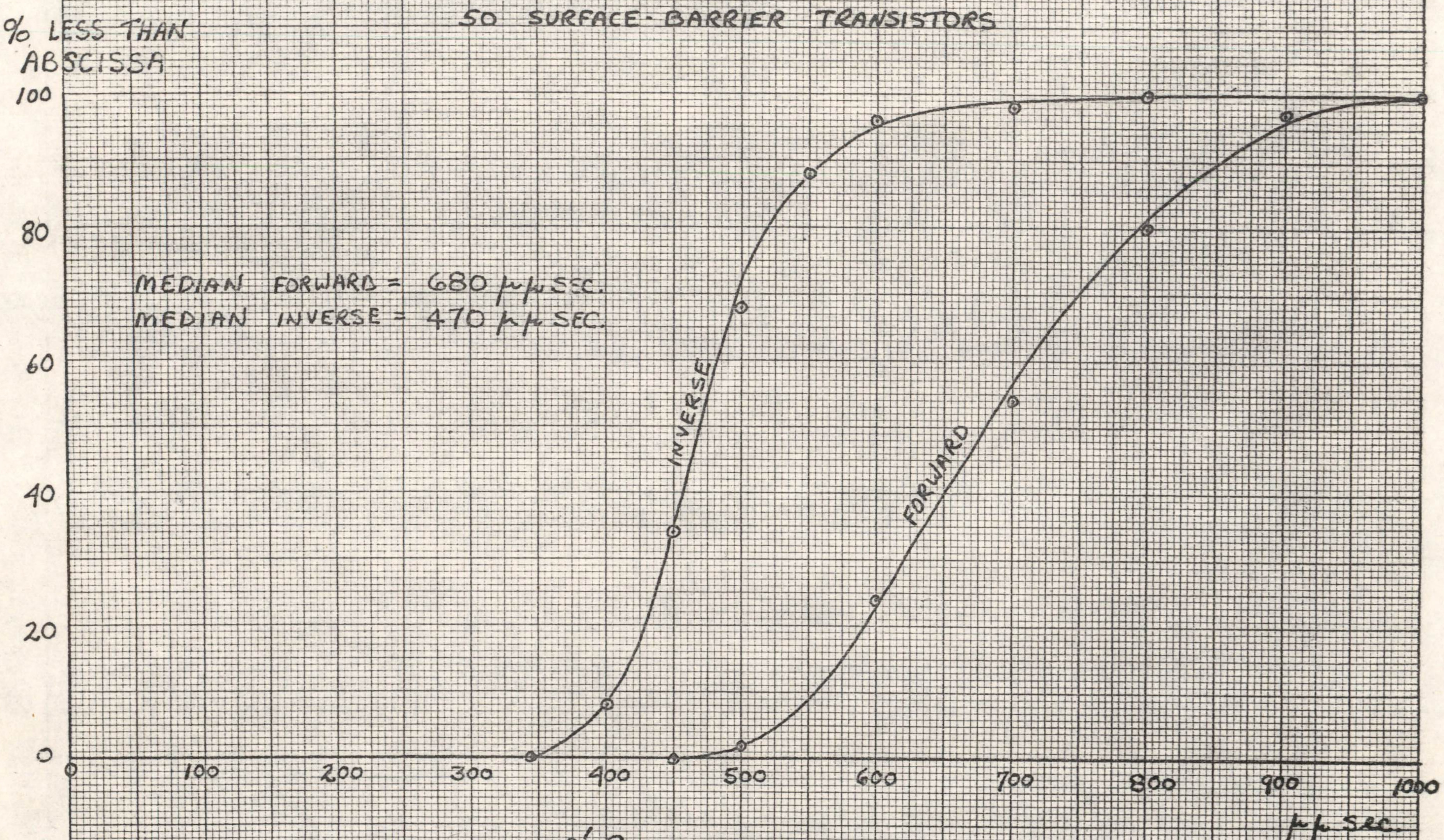
50 SURFACE - BARRIER TRANSISTORS



REVERSE AND FORWARD h_{FE} AT 10 mc/s

$I_b = 50 \mu a$ RF ; $I_E = 1 ma$; $V_c = 3$ VOLTS

A-80,10



50 SURFACE-BARRIER TRANSISTORS

MEDIAN FORWARD = 680 μμ SEC.
MEDIAN INVERSE = 470 μμ SEC.

$r_b' C_c$ DISTRIBUTIONS
MEASURED AT 4 MCS ; V_C = 3V I_C = 0.5 ma.

FIG. 27

A-80087

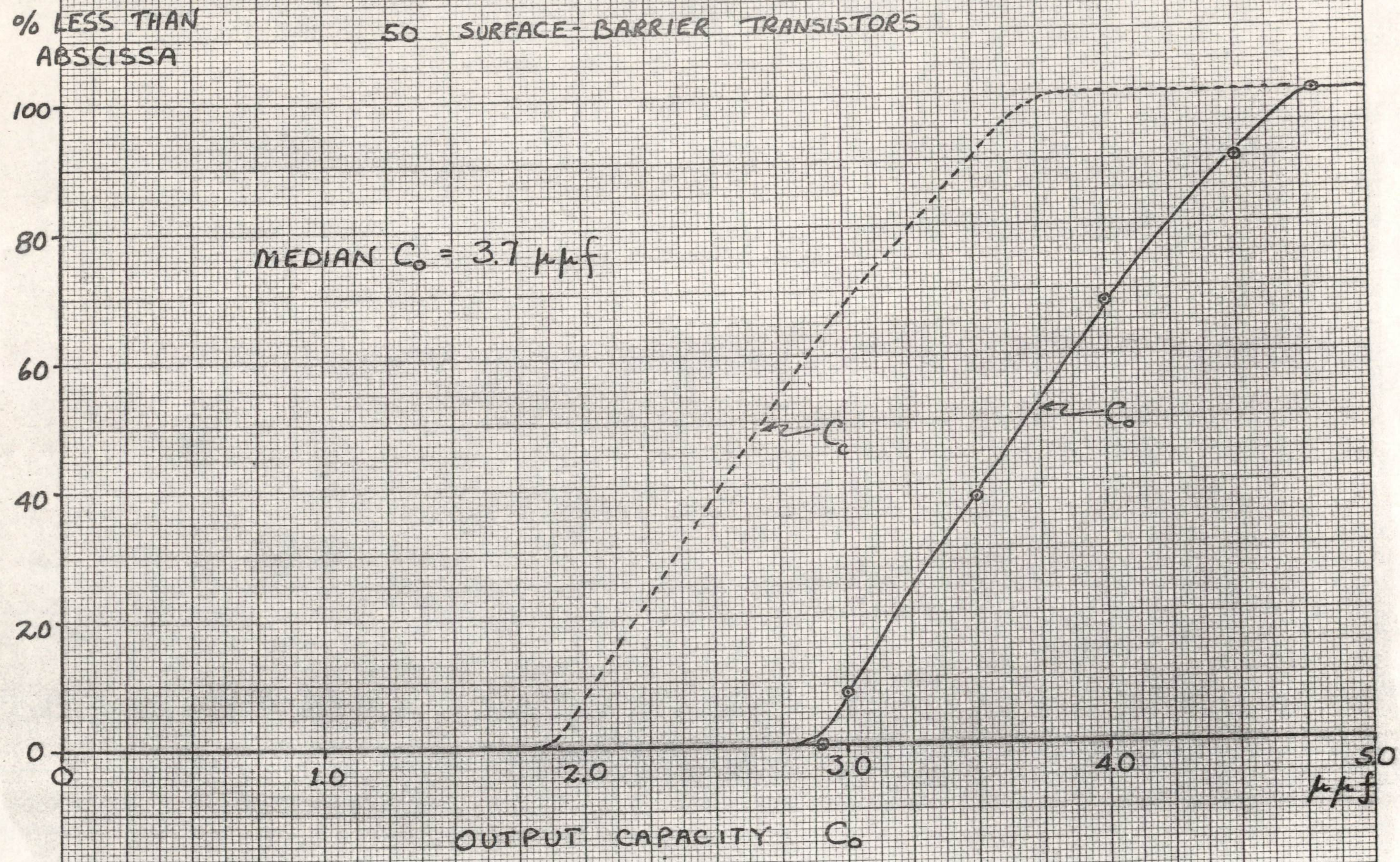


FIG. 28

A-80088

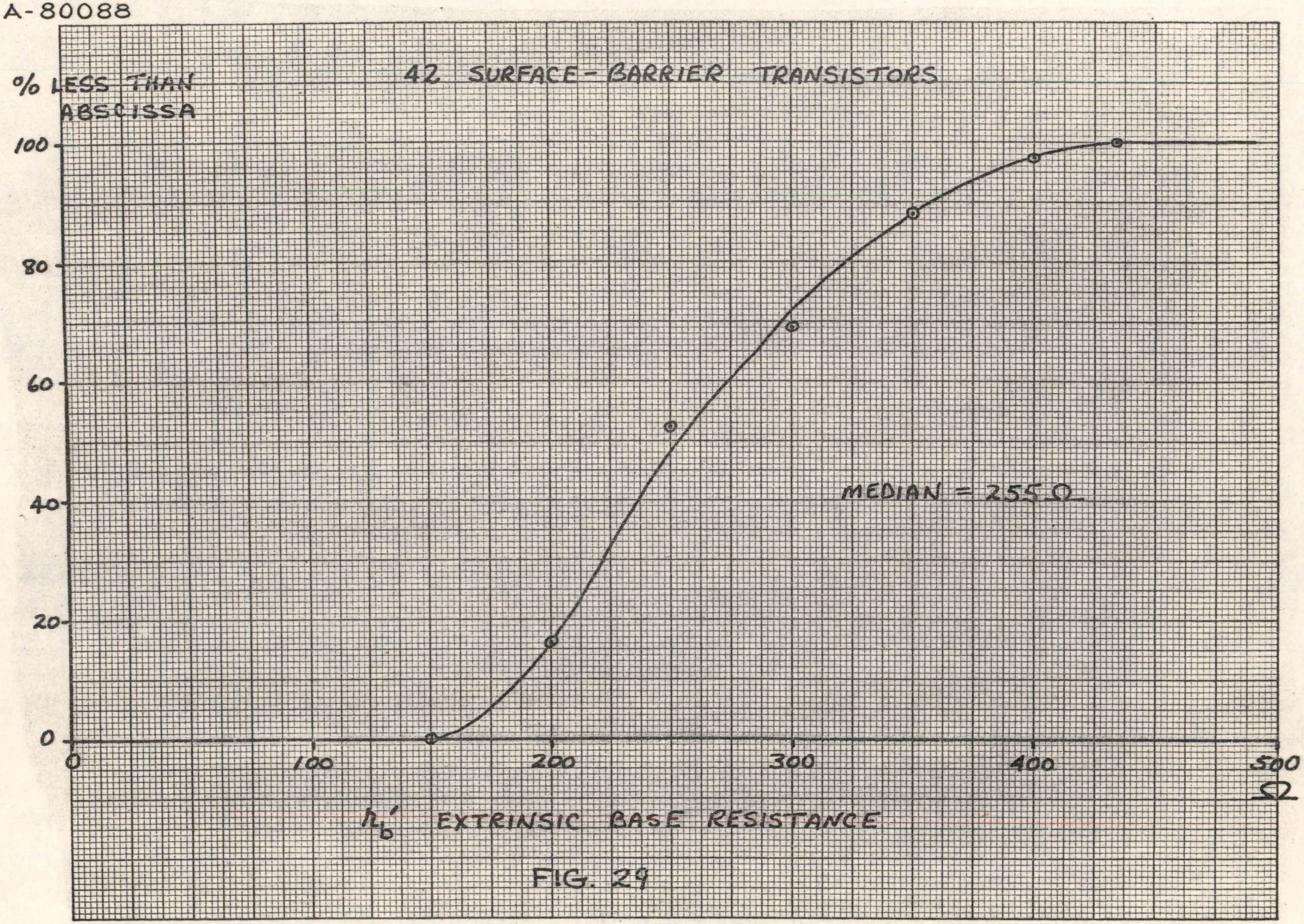


FIG. 29

A-80069

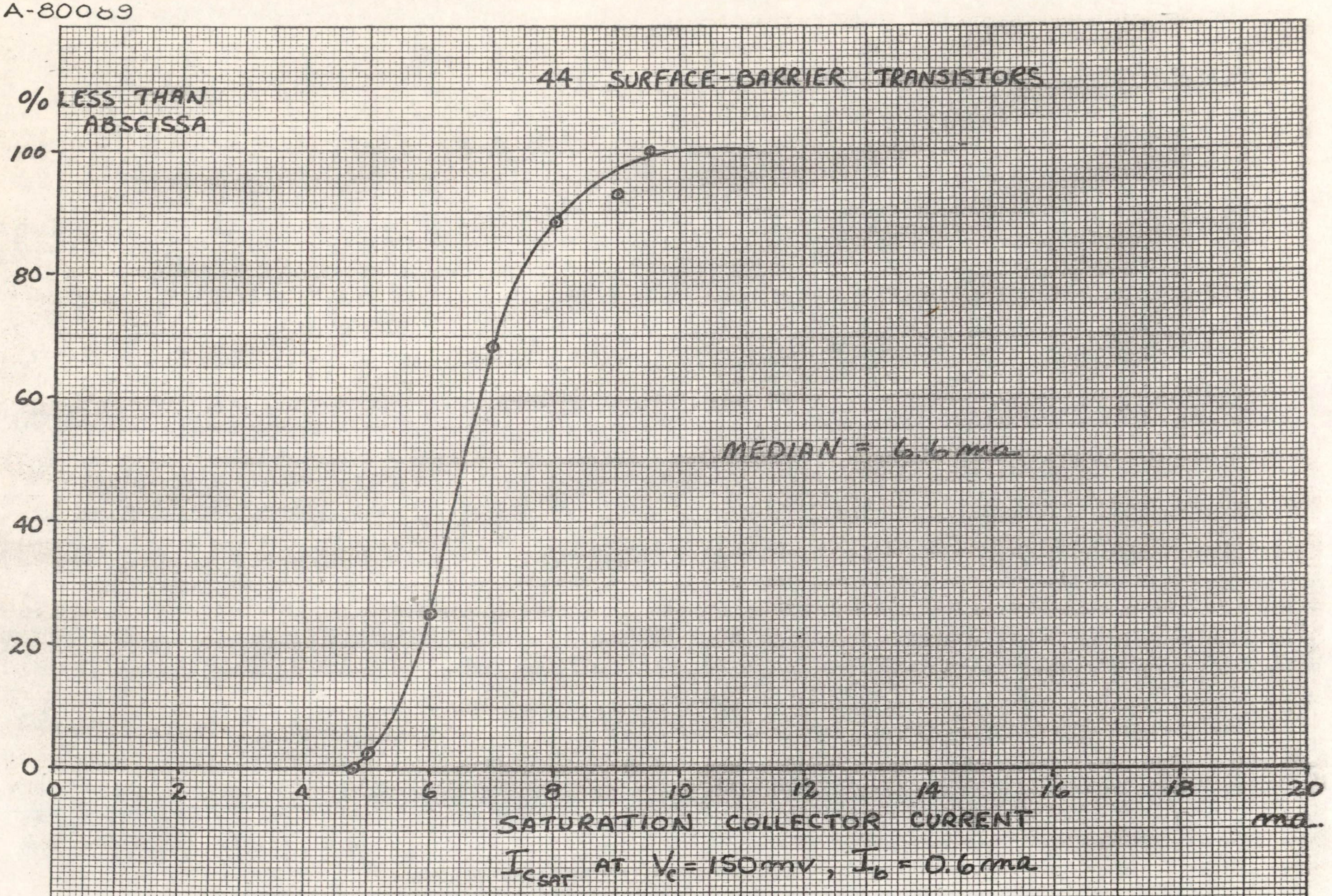


FIG. 30

A-80150

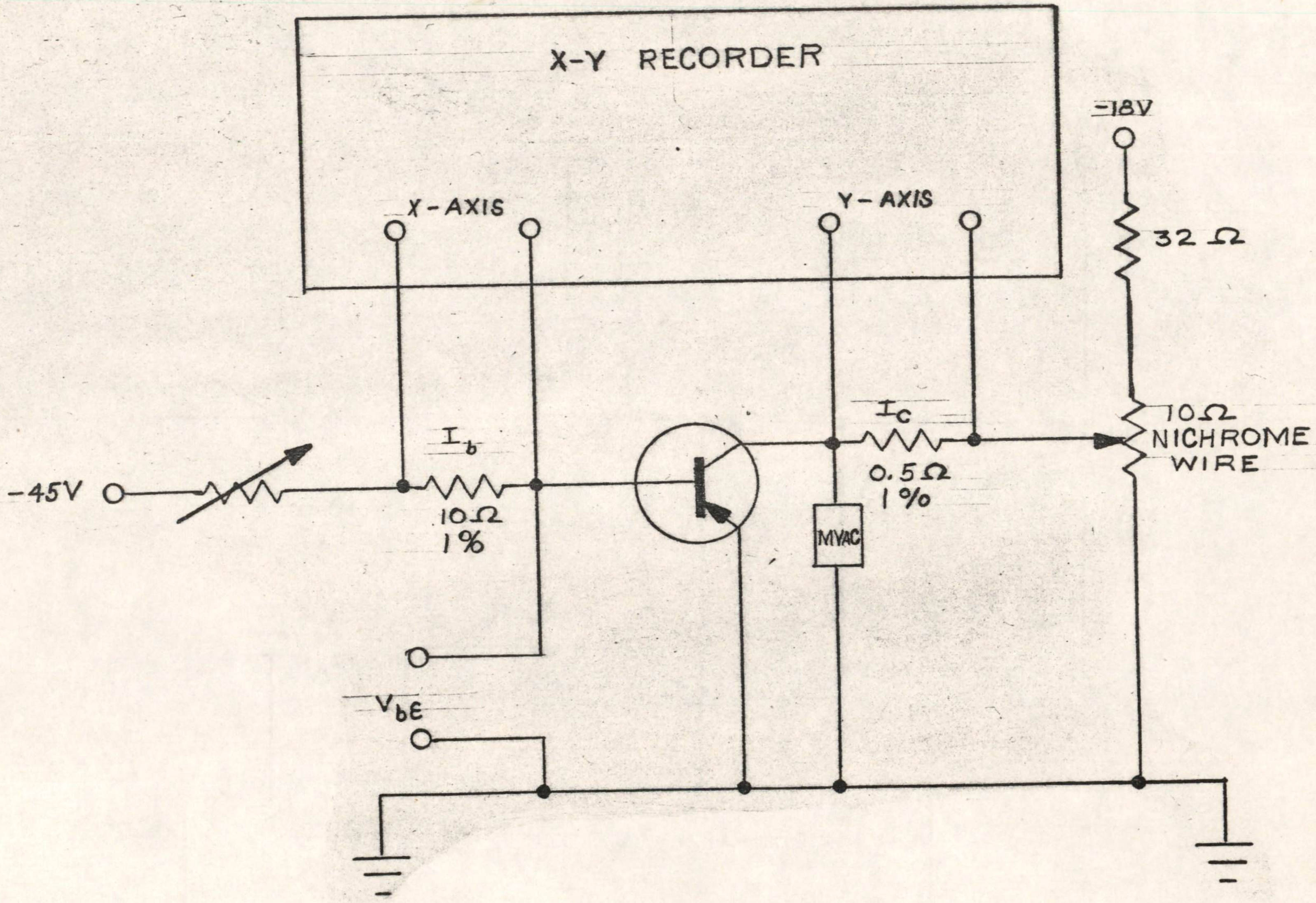


FIG. 31

CIRCUIT USED FOR OBTAINING SATURATION CHARACTERISTIC CURVES FROM X-Y RECORDER

A-80151

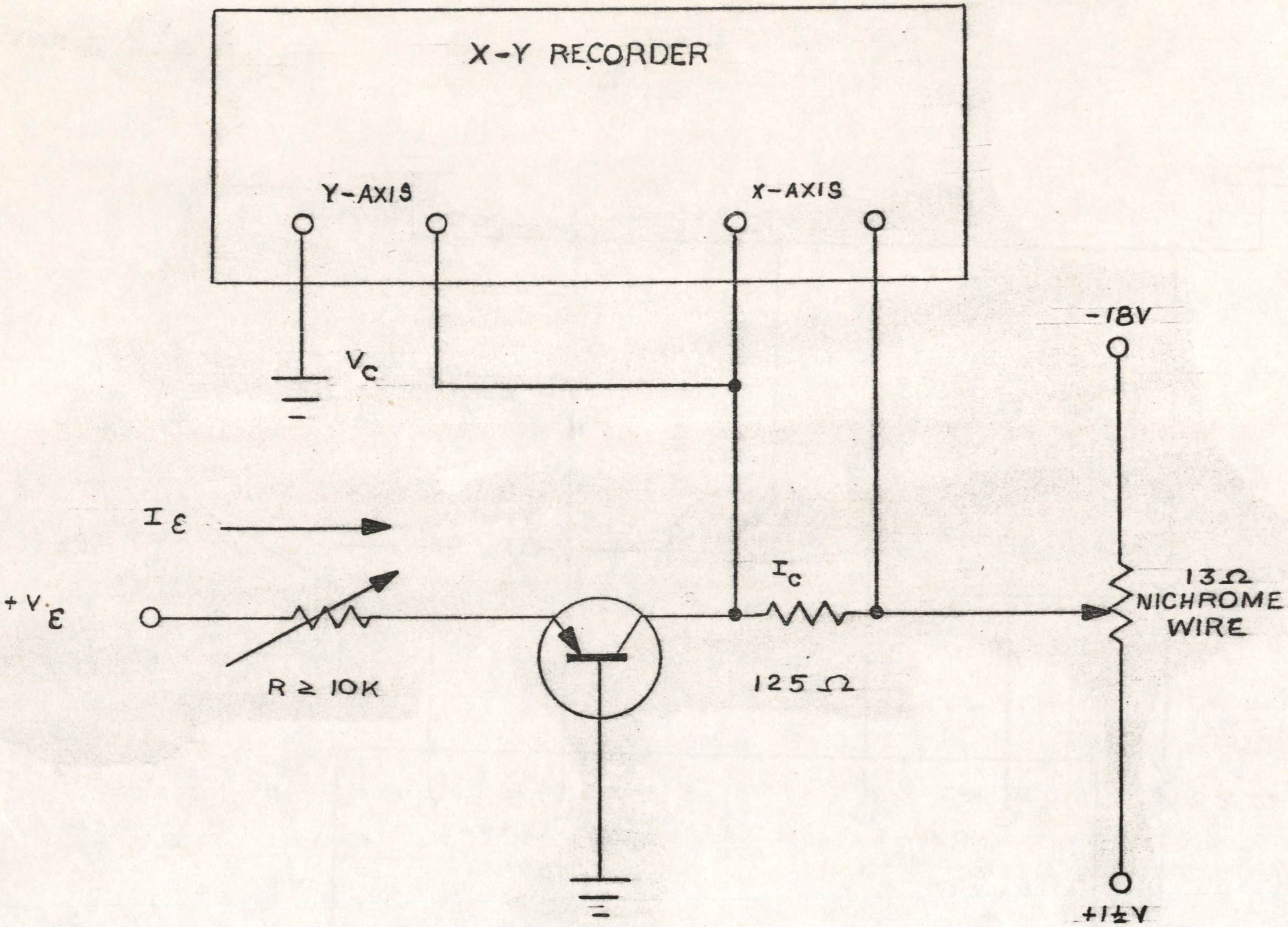


FIG. 32

CIRCUIT USED FOR OBTAINING COLLECTOR CHARACTERISTICS FROM X-Y RECORDER

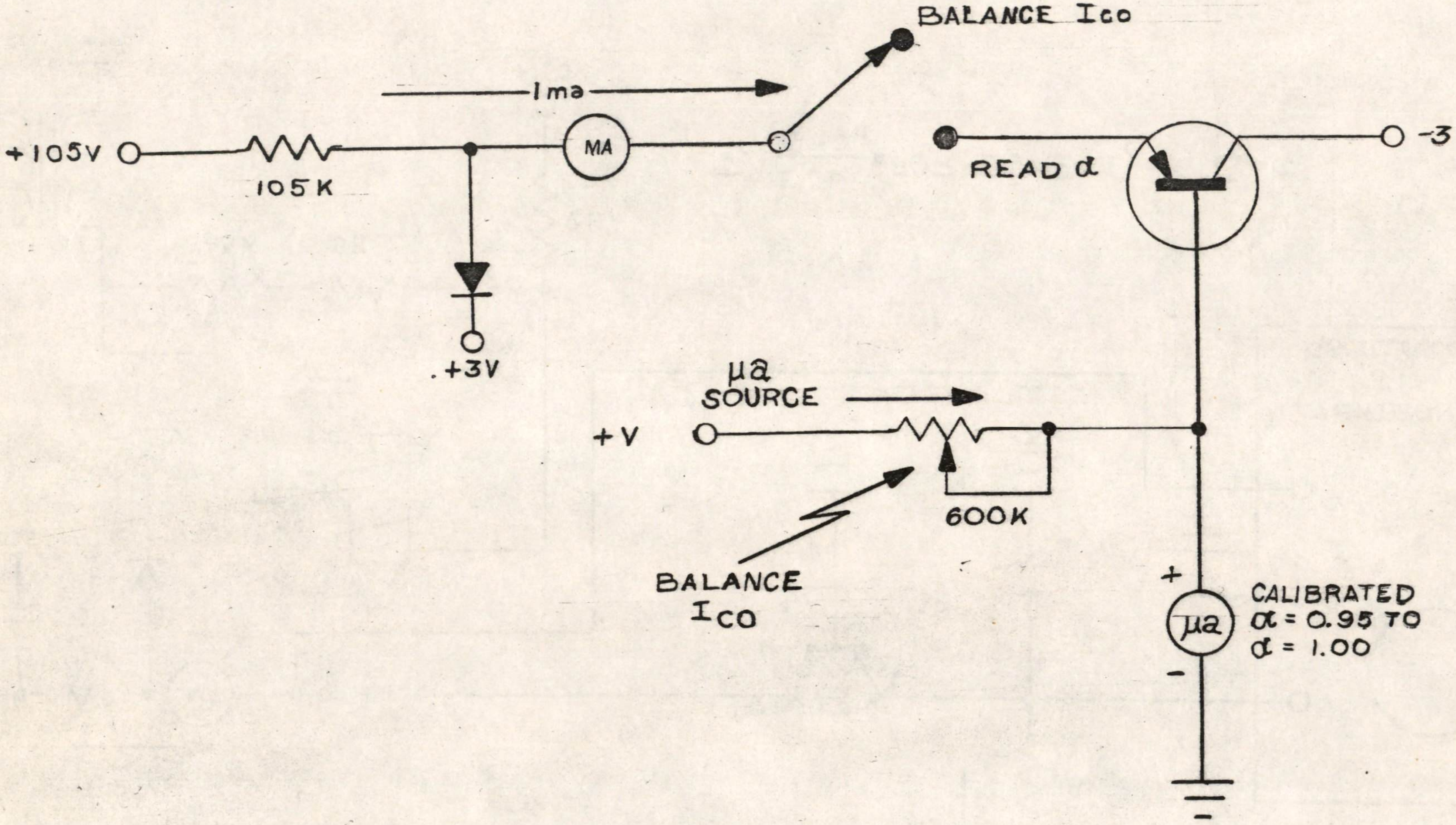
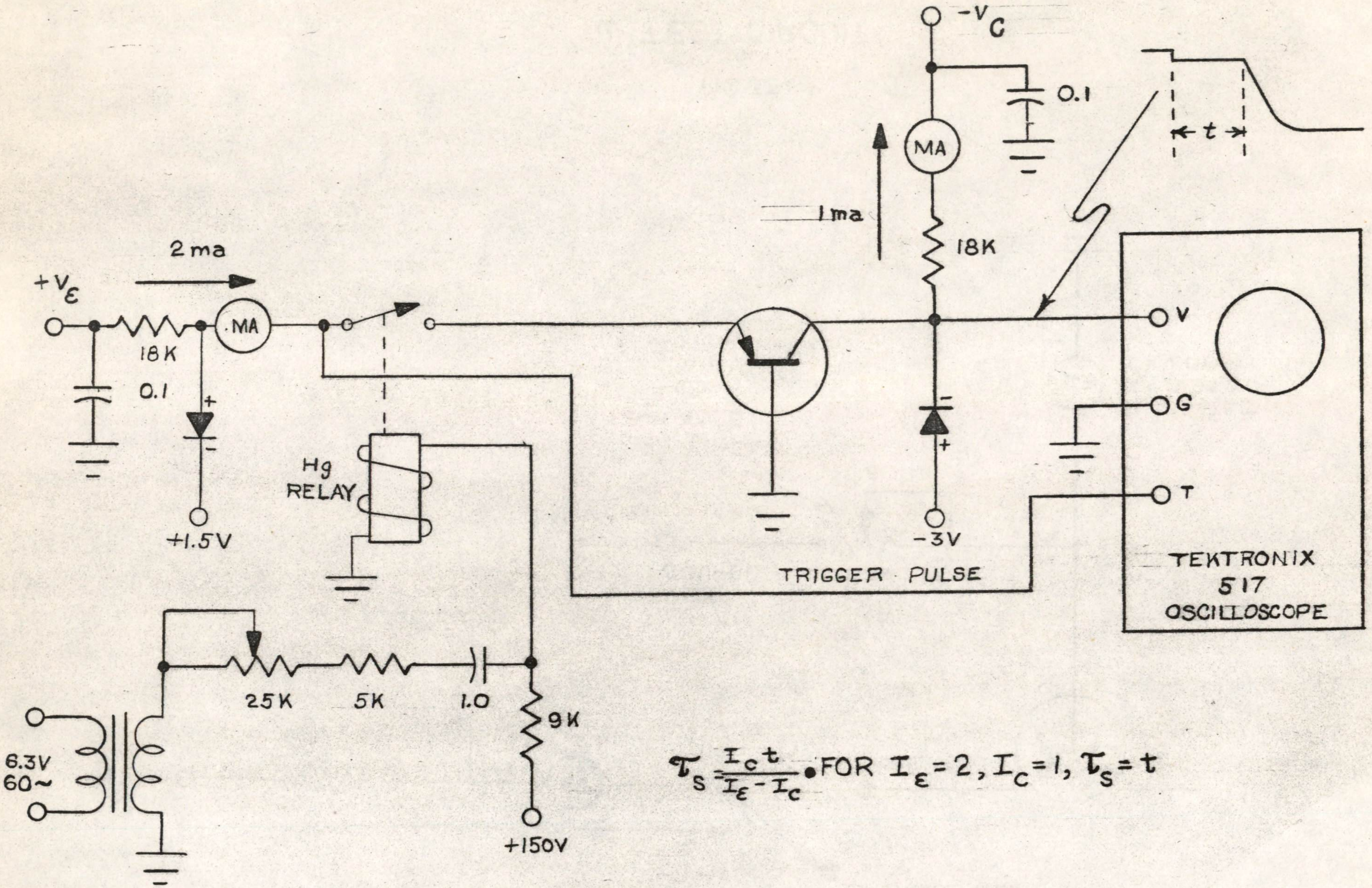


FIG. 33

α TEST CIRCUIT

A-80153



$\tau_S = \frac{I_C t}{I_\varepsilon - I_C}$ • FOR $I_\varepsilon = 2, I_C = 1, \tau_S = t$

FIG. 34

HOLE STORAGE TEST

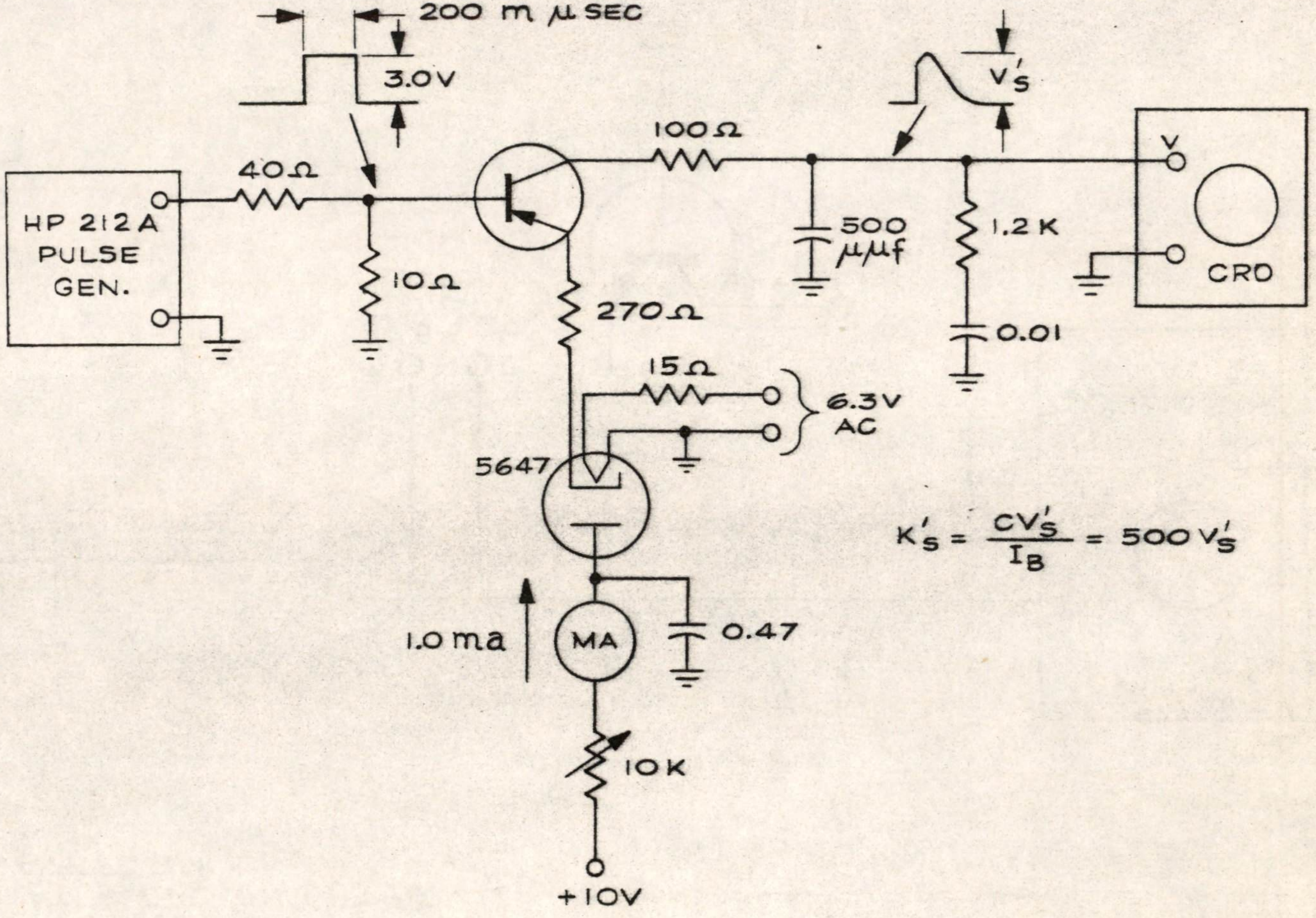


FIG. 35

K'_S TEST CIRCUIT

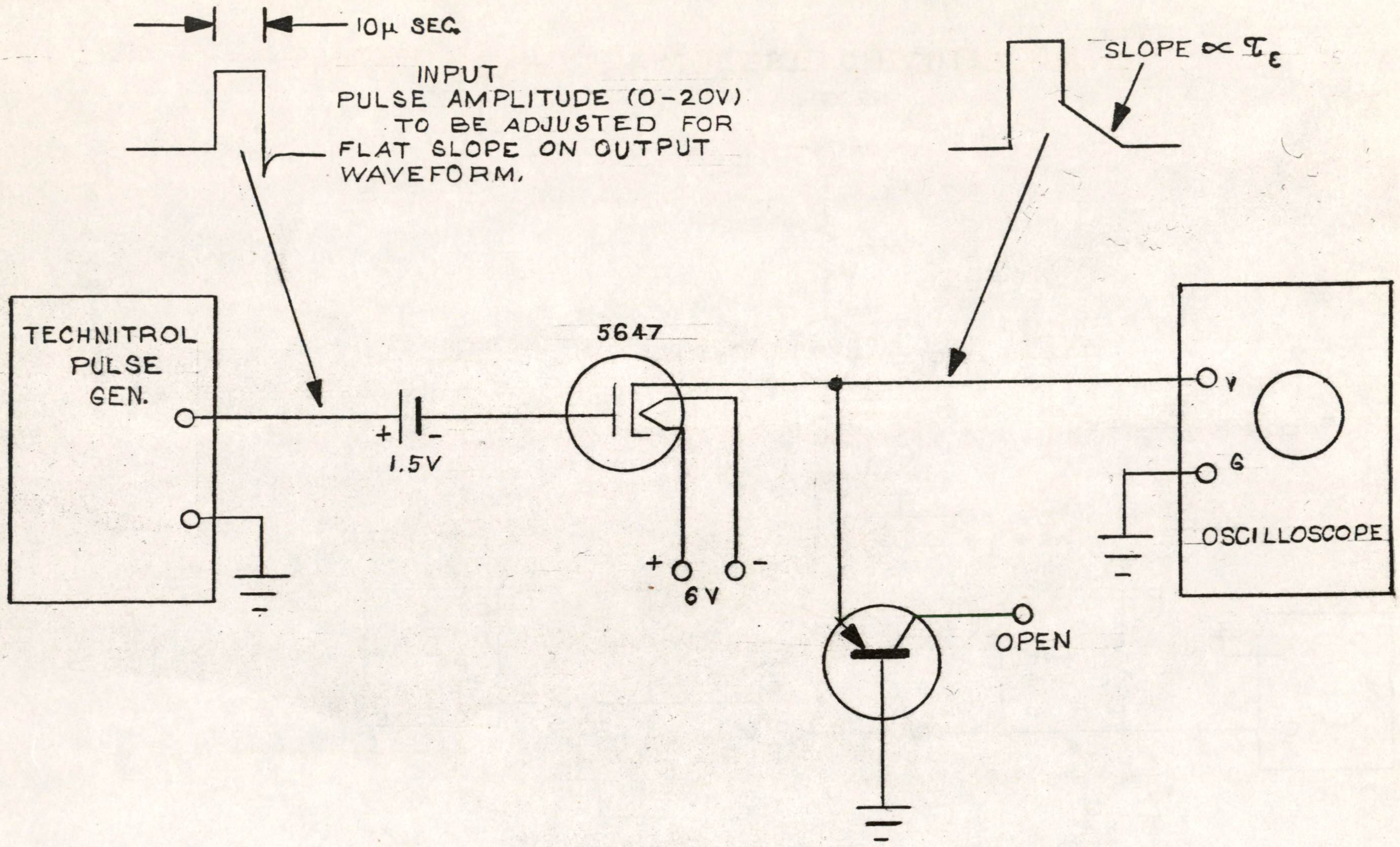


FIG. 36

EFFECTIVE BASE LIFETIME TEST CIRCUIT

(AFTER GIACOLETTO AND LEDERHANDLER)

A-80156

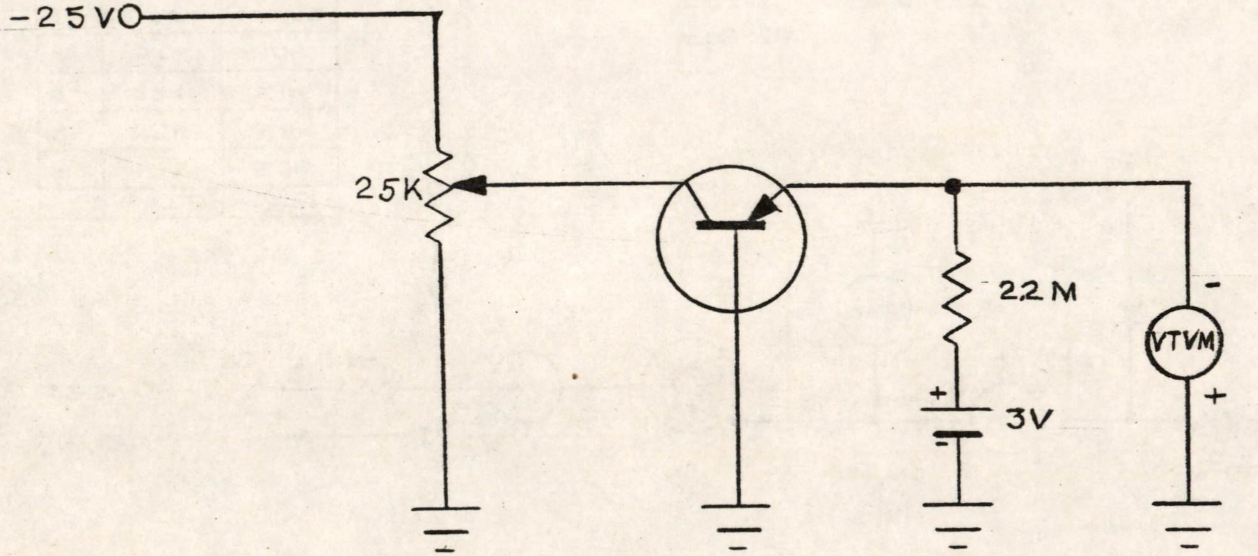
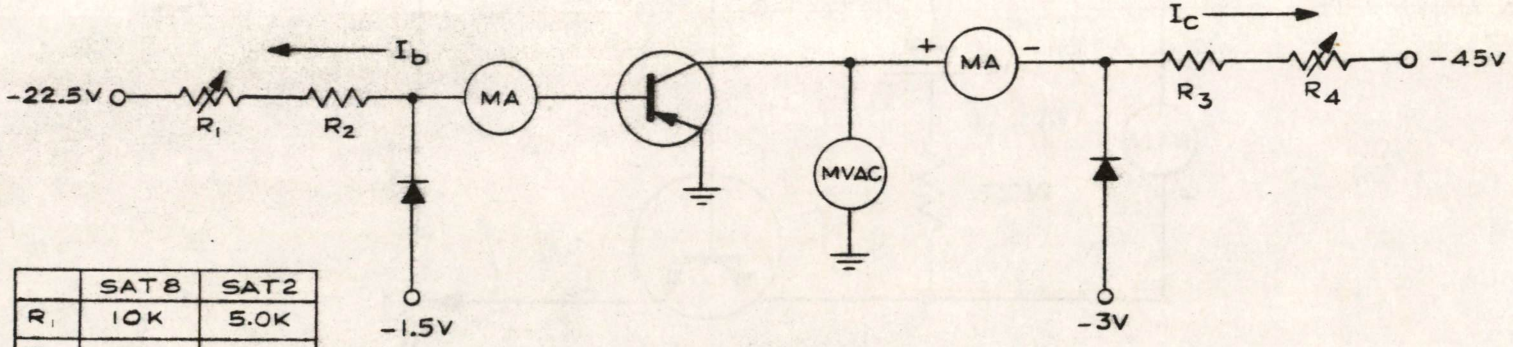


FIG.37

PUNCH-THROUGH VOLTAGE TEST CIRCUIT

B-80157



	SAT 8	SAT 2
R_1	10K	5.0K
R_2	75K	6.8K
R_3	22K	5.1K
R_4	5.0K	1.0K
I_b	2.5 ma	0.3 ma
I_c	8.0 ma	2.0 ma

FIG. 38

V_{CE} SAT $\begin{matrix}8\\2\end{matrix}$ TEST CIRCUIT

A- 80158

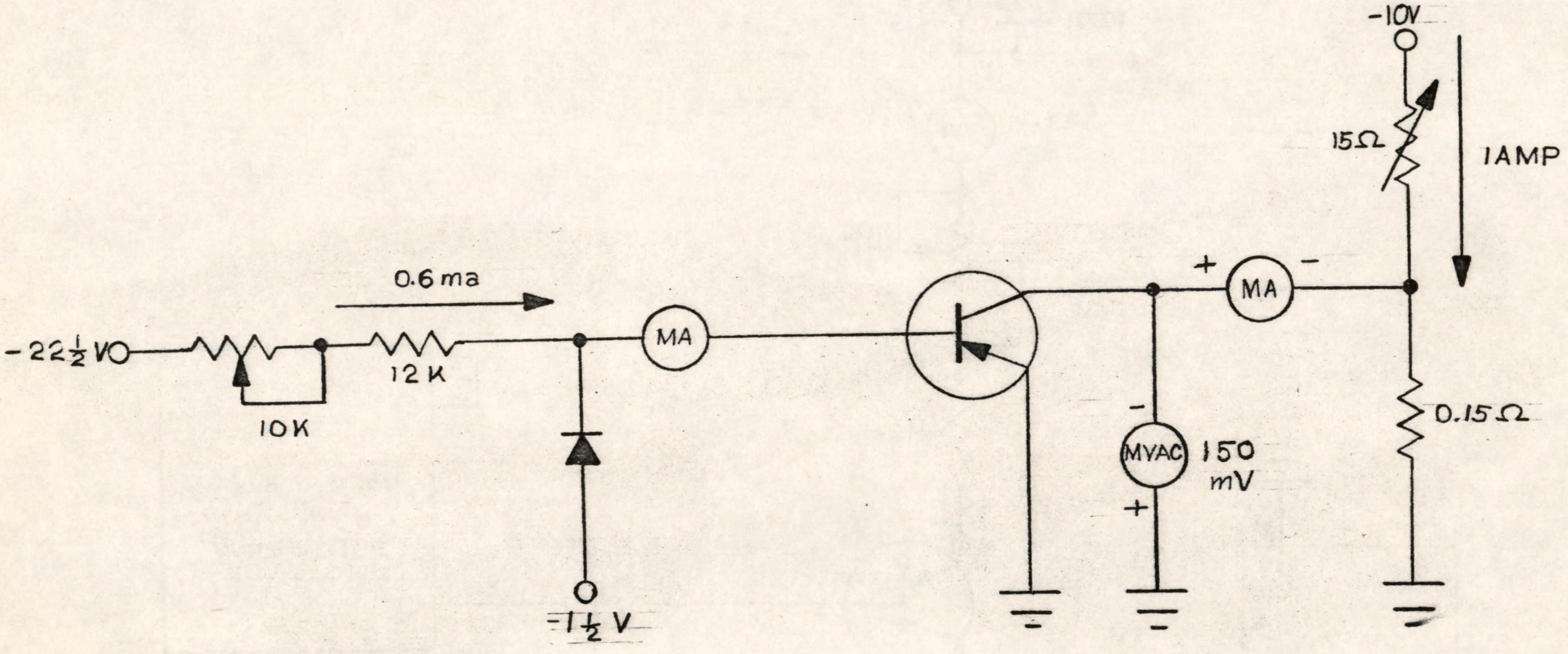


FIG. 39

SATURATION COLLECTOR CURRENT AT 150 mV

TEST CIRCUIT

A-80159

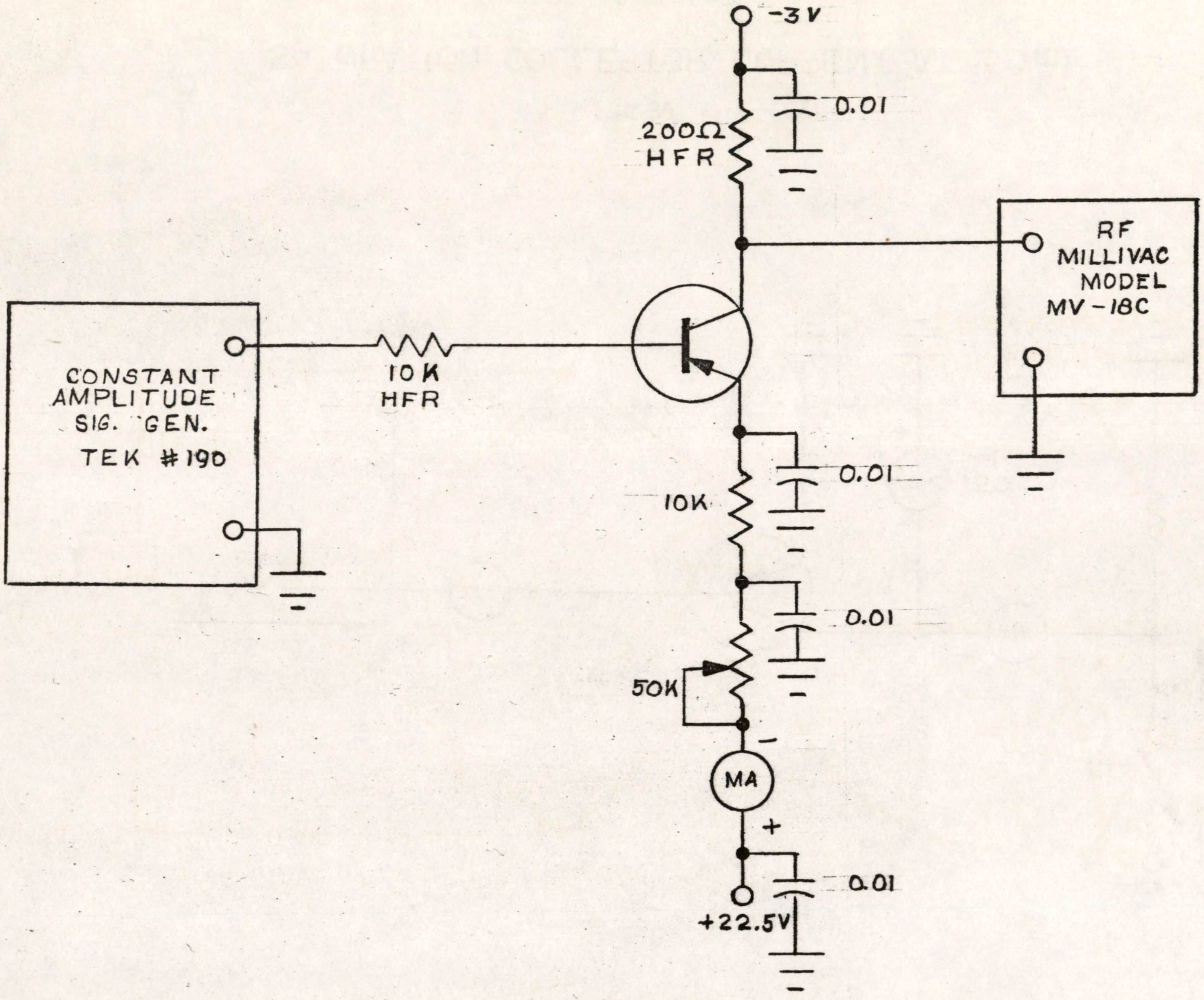


FIG. 40

h_{FE} TEST CIRCUIT

A-80160

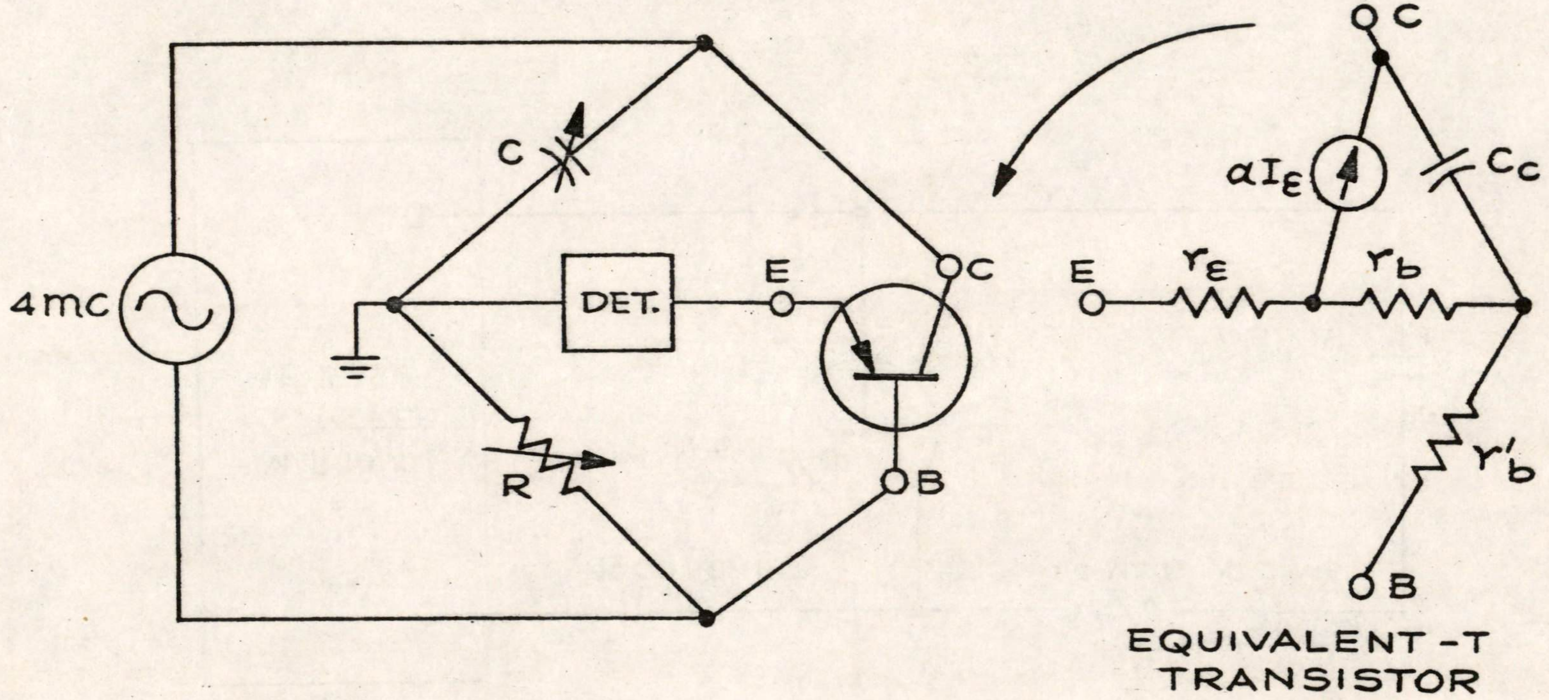


FIG. 41

$r'_b C_c$ BRIDGE

(AC CIRCUIT ONLY)

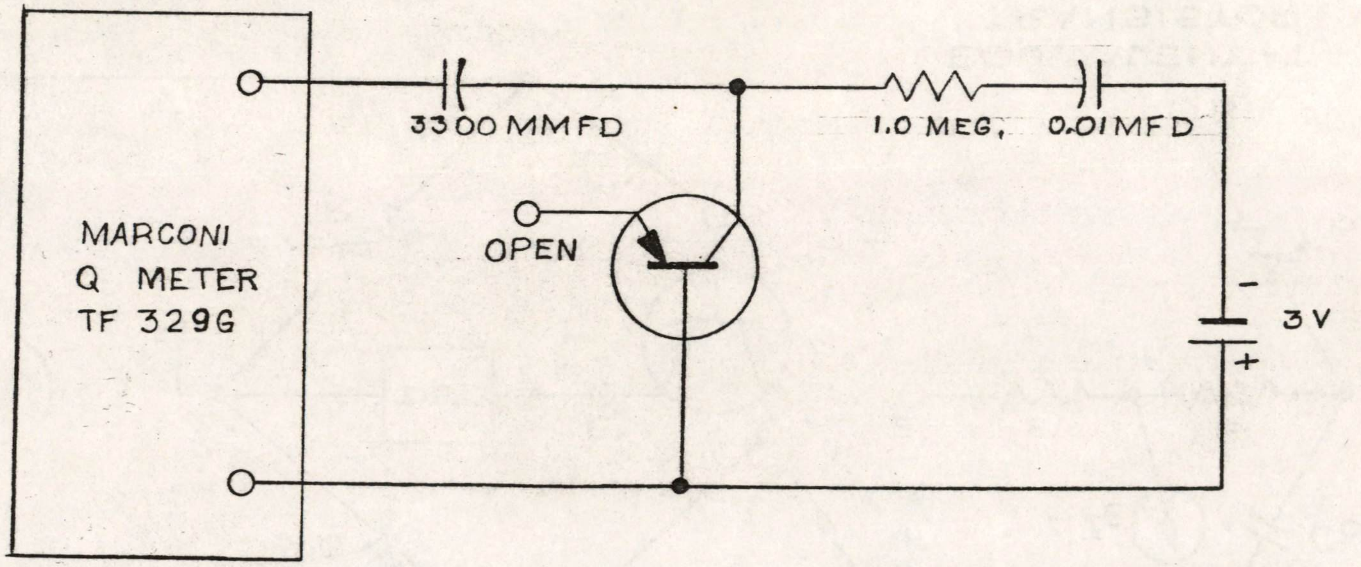


FIG. 42

C_C TEST CIRCUIT

K.H. Olsen
B063

Memorandum 6M-4956
Sheet 1 of 9 Sheets

Division 6 – Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: SELECTIVE MASKING

To: J. C. Proctor

From: G. R. Heidler

Date: 9 April 1957

Approved: ______________________
J. C. Proctor

Abstract: A new approach to the preparation of planar masks for printed wiring memory planes is called "Selective Masking". The new method will produce detailed repetitive patterns with an accuracy of ± .001 or better, in the pattern area, using standard photographic equipment and procedures on a 1:1 scale.

Selective Masking is based on the use of a negative mask containing an accurately spaced series of transparent parallel lines. Another negative containing a single straight transparent line is placed under the first negative so that the single line is at right angles to the mask lines. An exposure on photographic film will produce segments of the single line corresponding to the transparent lines on the mask.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

DISTRIBUTION LIST

Group 60 Staff

Division 2

W. L. Gardner, Dr.
R. J. Horn
E. D. Thomas

Division 6

R. R. Everett
E. A. Guditz
J. L. Mitchell
K. H. Olsen
W. N. Papian

Division 7

P. J. Connolly
L. L. Grant
M. M. Hannosh
H. M. Lane
E. B. Murphy

The preparation of miniaturized, complex, repetitive and accurate patterns, as a photographic negative, for subsequent use in the production of printed wiring memory planes is generally expensive, time consuming and requires special photographic equipment to obtain the necessary accuracy. Photo-reduction is generally used. A technique called Selective Masking, will produce such patterns to actual size without the need of reduction and may be produced quickly with standard photographic equipment.

Selective Masking uses standard photographic techniques, a step and repeat table and an overhead source of light directed through an iris opening. A step and repeat table is a unit which will permit a line(s) or pattern to be reproduced more than once at accurately speced distances. Selective Masking has been used to produce the necessary planar masks for a 4 x 4 printed wiring memory plane, the rear mask pattern of which is shown in Figure No. 1. The mask line pattern contains only vertical and horizontal lines. These have been separated into vertical and horizontal line segments in Figure No. 2.

Figure No. 2 also shows that each segment is a part of a single straight line, e.g., note Line No. 1, Figure No. 3 shows how the unwanted portions of Line No. 1 can be eliminated by masking. Since the pattern is symmetrical about the vertical center line, the mask used to produce the segments of Line No. 1 is also used to produce the segments of Line No. 9. The same mask is also used to produce the segments of Line Nos. 8 and 16. Another mask produces the line segments of Line Nos. 2, 7, 10, and 15. A third mask produces the line segments of Line Nos. 3, 4, 5, 6, 11, 12, 13, and 14.

Two parallel lines are used with the same spacing as that between Line Nos. 1 and 9. By accurately stepping the two parallel lines and selectively masking according to the position of the lines, the vertical pattern is produced as shown in Figure No. 2. A similar procedure produces the horizontal half of the pattern shown in Figure No. 2.

The complete pattern is produced by registration of the two halves. Any discrepancies, overlength lines, or extra line segments may be corrected or removed at this time. A contact print of the registered halves produces the pattern as a negative. Figure No. 6 is a contact print of the front and rear masks for the printed wiring memory plane. Although the front mask is more complicated than the rear, it was produced by a simple extension of the procedures outlined above.

The selective masks are made from the same two parallel lines that are used in the pattern production since the vertical spacing is the same as the horizontal spacing. A base line is added to the mask, as shown in Figure No. 3. An aluminum strip is cemented on the base line as shown in Figure No. 4. The base line is any fixed distance from the pattern from which all measurements for the masks are made. The metal strip provides alignment for the selective mask when it is used with the step and repeat table. After the metal strip is attached, any areas that were not exposed and need opaquing may be taped with opaquing tape. Figure No. 4 shows a completed mask ready for use.

Figure No. 5 shows how the selective mask is placed in the step and repeat table frame at a right angle to the parallel lines. The procedure previously described produces the patterns.

A bleed line exposure is used on portions of the selective masks where 3 or more lines are printed together such that their edges touch. The outer lines are made by conventional exposure. The inner line(s) is made by room light for a short period of time (approximately 10 seconds) such that the line is overexposed. The line will spread or "bleed" into the line on either side as shown in Figure No. 7. This speeds the process of making the selective masks and eliminates the necessity for hand opaquing.

Accuracy of this technique is limited by the accuracy of the step and repeat machine which is calibrated to ± .001", and has been generally producing to a tolerance of ± .0005".

The technique of Selective Masking has been used to make the wiring pattern for a 4 x 4 memory plane. It will be used to produce the masks for a 16 x 16 and possibly for a 64 x 64 memory plane.

Glen R. Heidler

Glen R. Heidler

GRH/jhb

Attachments:

Figure No.	Drawing No.
1	C-69659
2	C-69660
3	A-69661
4	A-69662
5	A-69663
6	A-69651
7	A-69733

C-69659
SN-1771

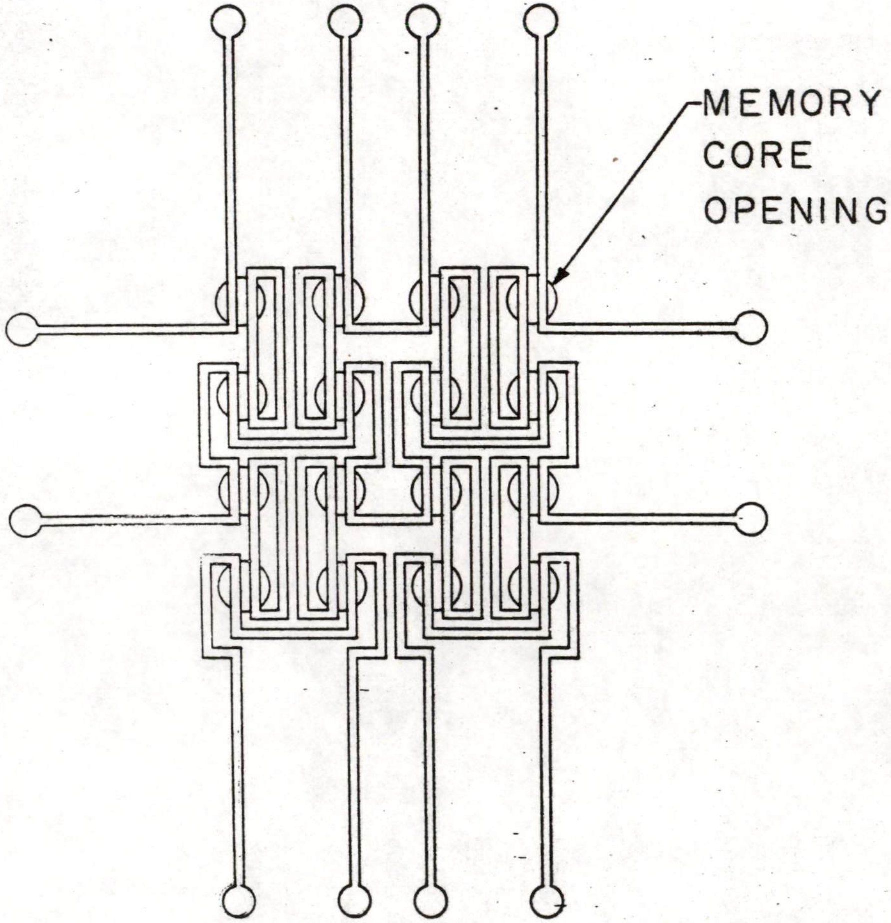
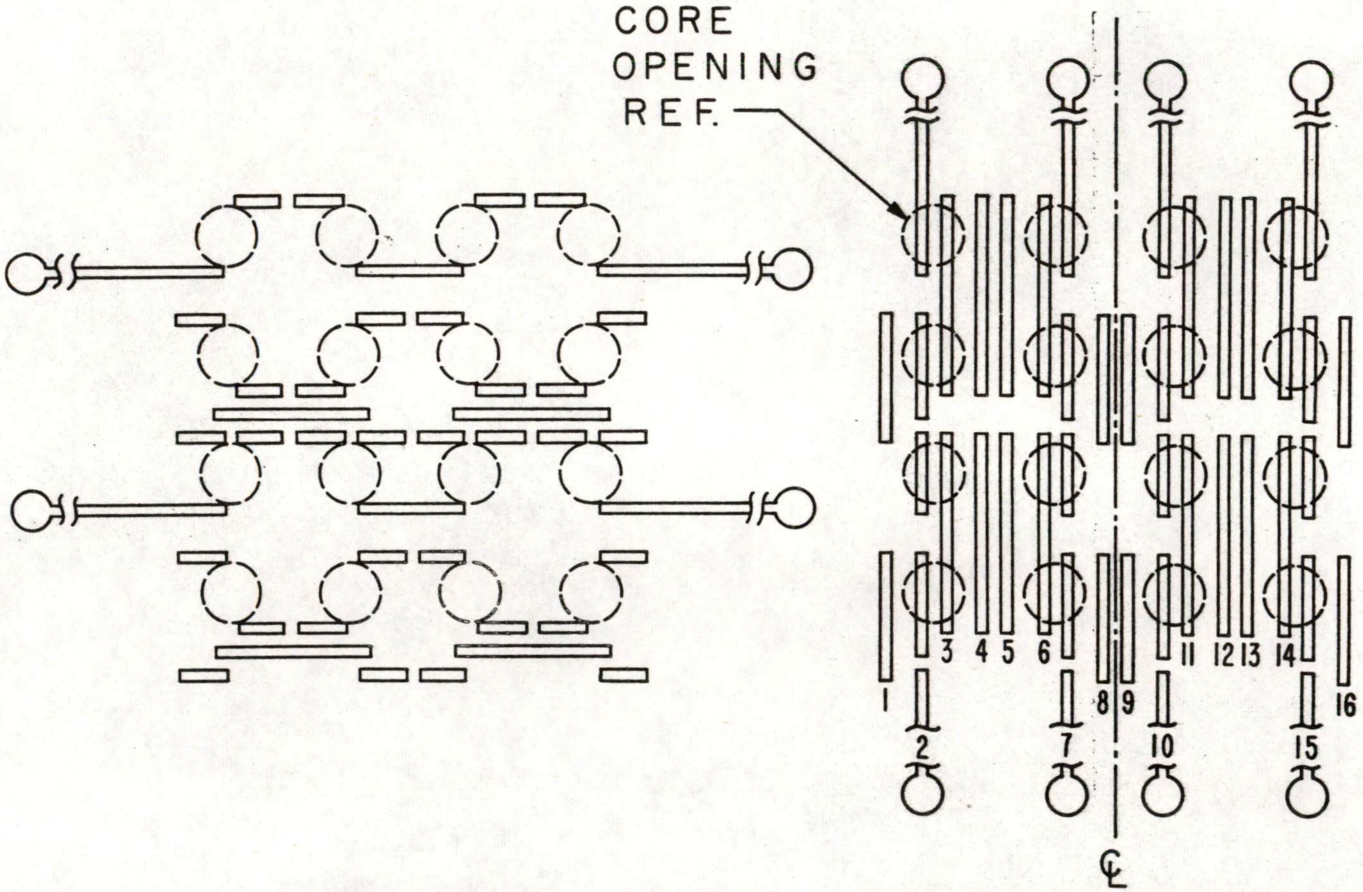


FIG 1
REAR MASK PRINTED MEMORY PLANE PATTERN



HORIZONTAL FIG 2 VERTICAL

PATTERN SEPARATION INTO VERTICAL AND HORIZONTAL LINE SEGMENTS

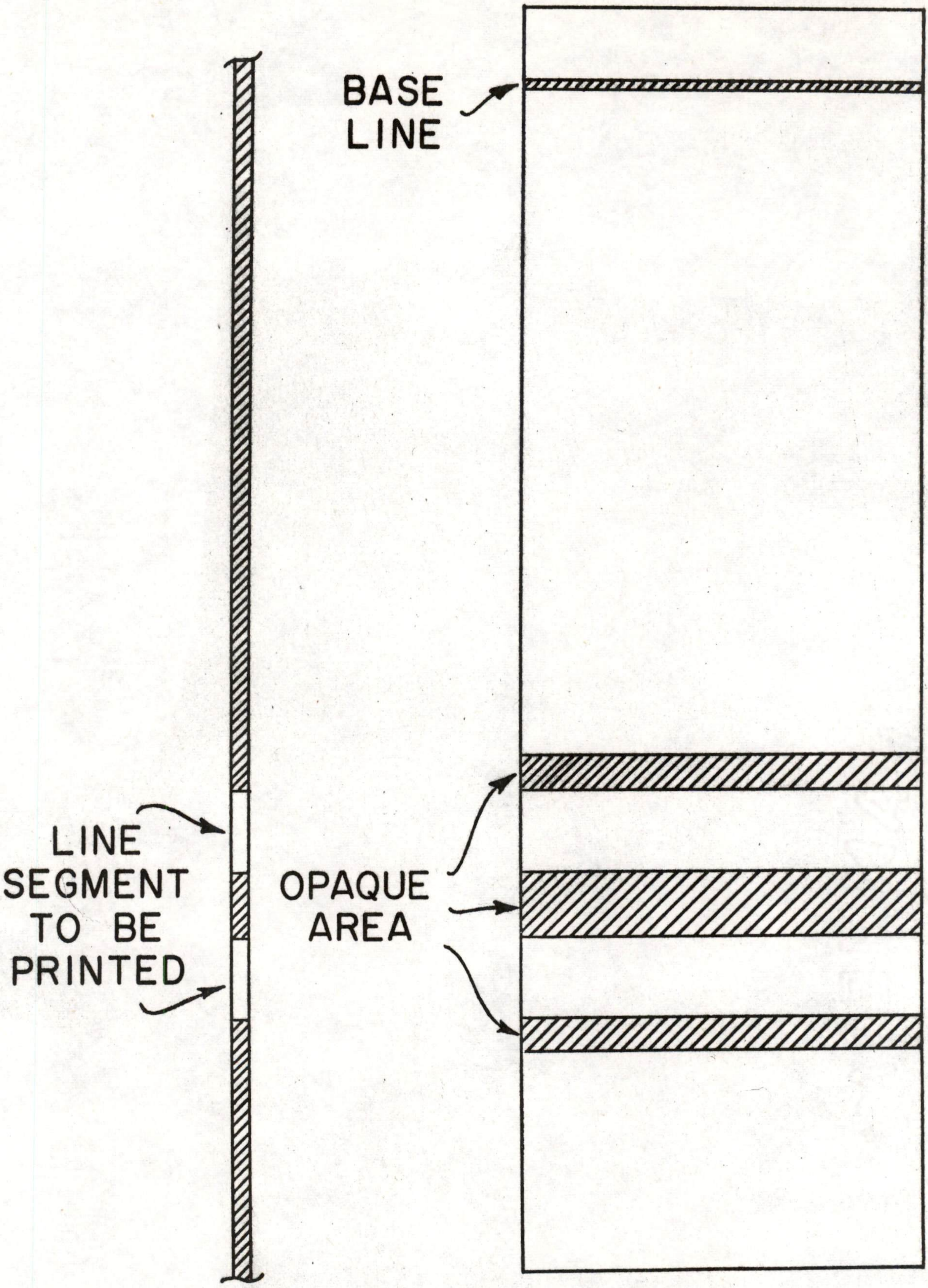


FIG. 3

A-69661
SN-1773

LINE AND SELECTIVE MASK

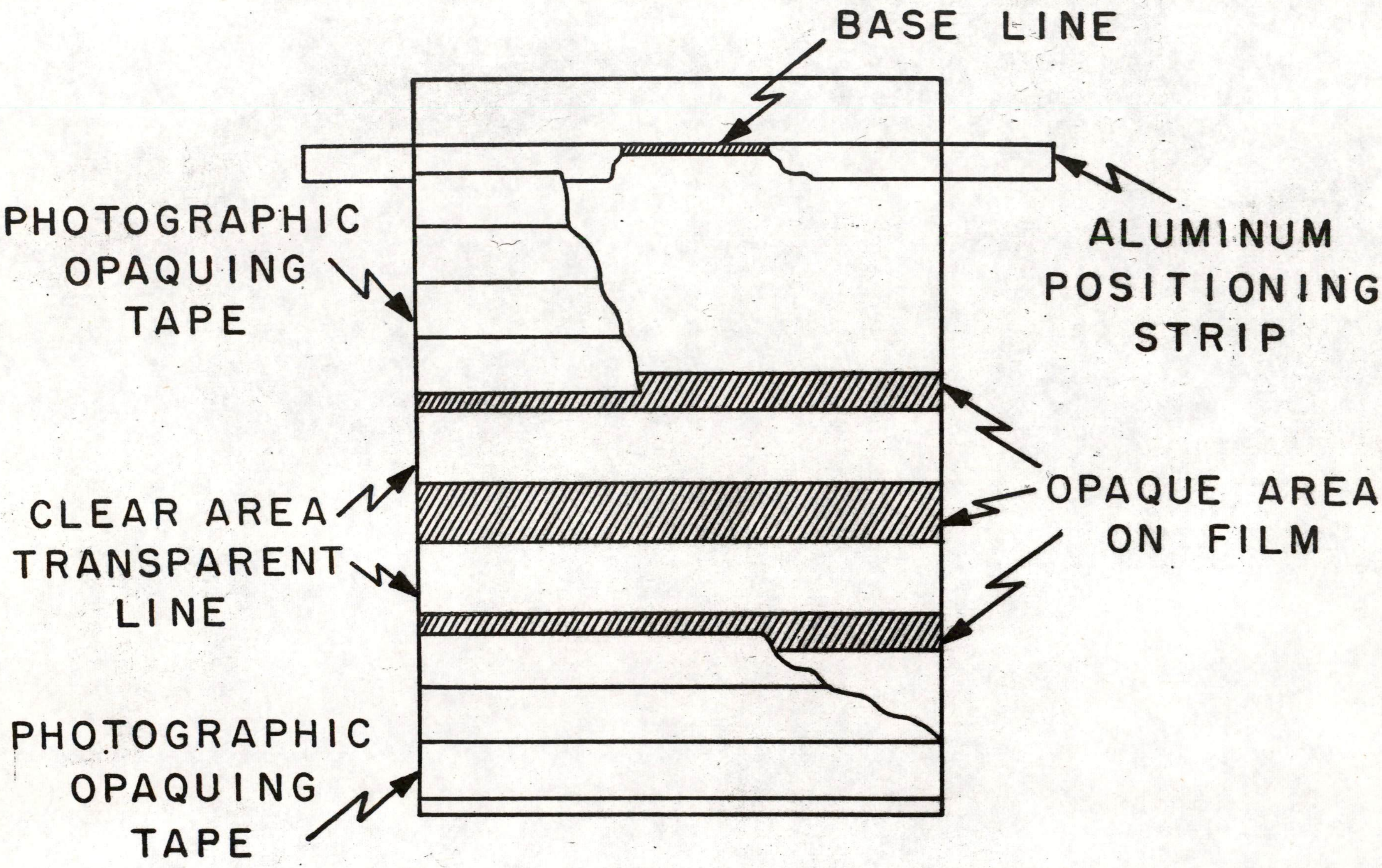
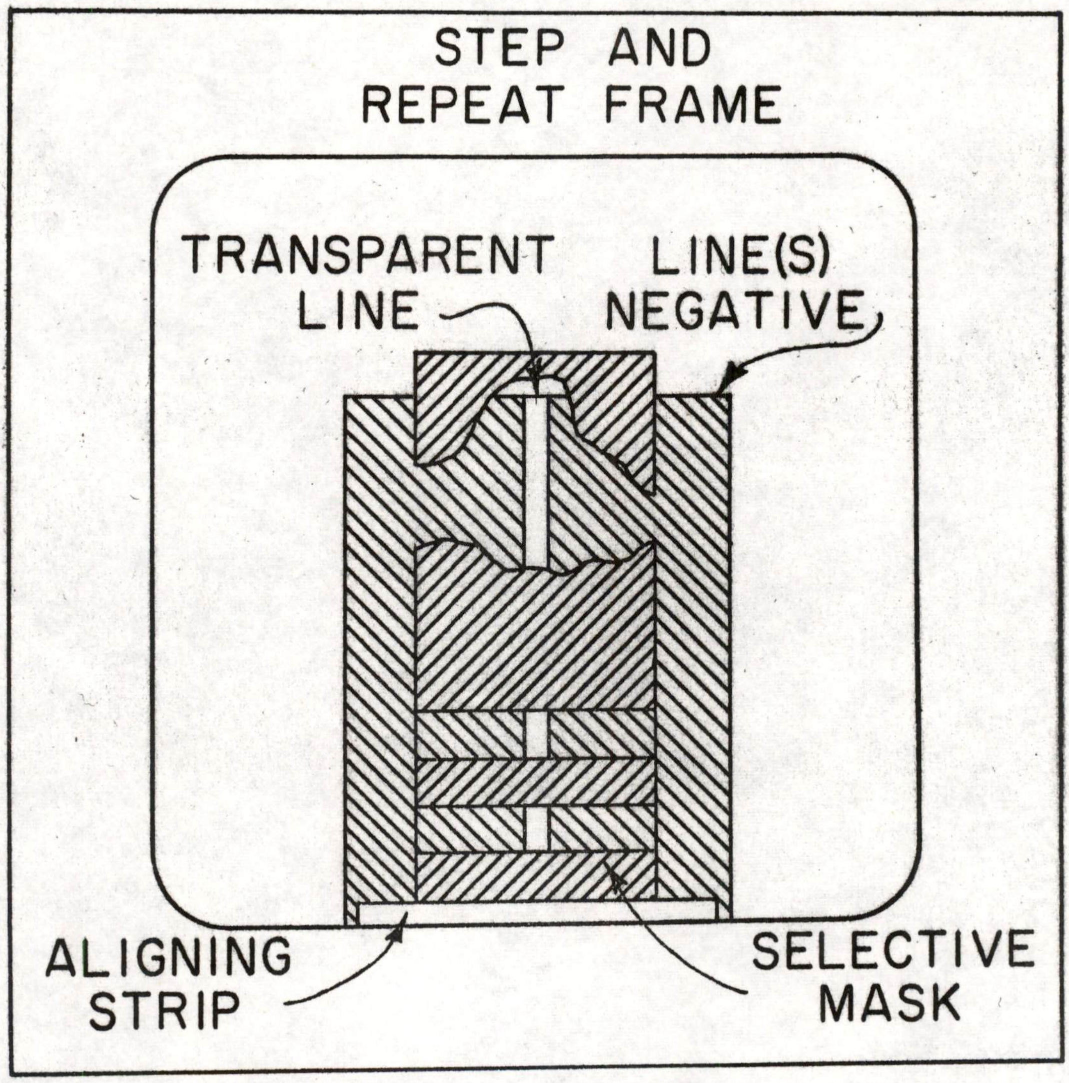

BASE LINE
PHOTOGRAPHIC OPAQUING TAPE
ALUMINUM POSITIONING STRIP
CLEAR AREA
TRANSPARENT LINE
OPAQUE AREA ON FILM
PHOTOGRAPHIC OPAQUING TAPE

FIG. 4

FINISHED SELECTIVE MASK



NOTE: CROSSHATCHING IS OPAQUE PORTION OF FILM

FIG. 5

POSITION FOR EXPOSURE

A-69663
SN-1775

A-69651

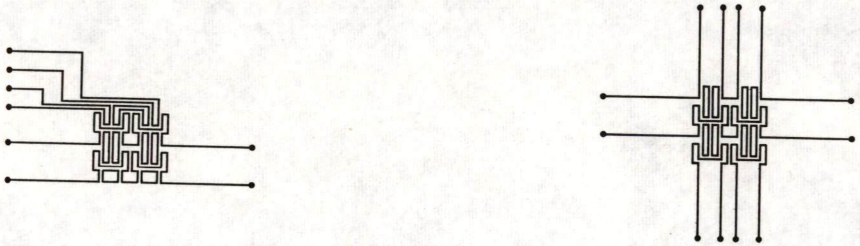


FIG. 6

PATTERN FOR PRINTED MEMORY PLANE AS PRODUCED (ACTUAL SIZE)

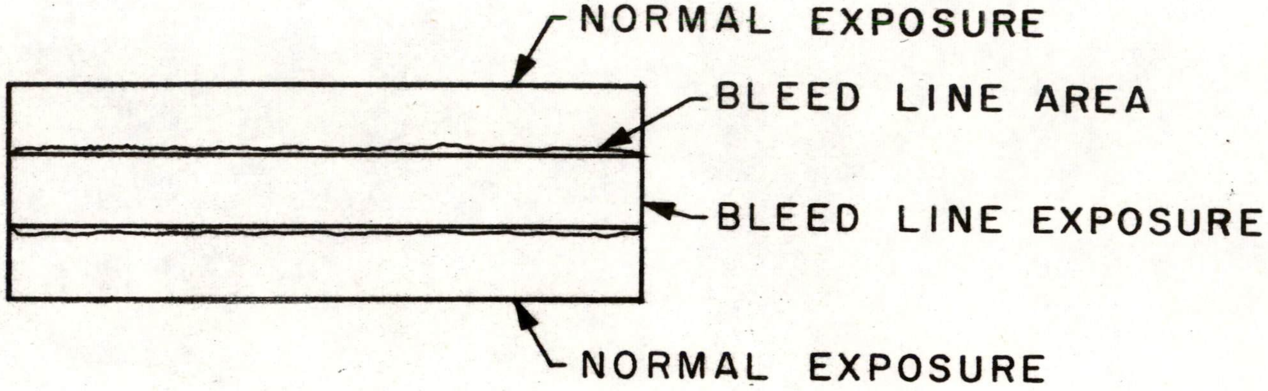


FIG. 7

BLEED LINE EXPOSURE

Memorandum

6M-4789 Page 1 of
Sheet 1 of 34 sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: A FUNCTIONAL DESCRIPTION OF THE TX-O COMPUTER

To: Distribution List

From: John T. Gilmore, Jr. and H. Philip Peterson

Date: November 20, 1956

Approved: Wesley A Clark
Wesley A. Clark

Abstract: The TX-O is an experimental digital computer which was constructed to check transistor circuitry and a 256 x 256 magnetic core memory. The logical design is rather simple since it has only four instructions. Three of these refer to memory in the normal way, but the fourth has the interesting feature of providing the facility to micro-program via time pulses. How useful this is will be determined by the experience gained in programming for TX-O. This memo has been written to give the reader a working knowledge of the computer's logic, usefullness, and capabilities.

Distribution List:

Group 63 Staff
Bagley, P. R.
Heart, F.
Mayer, R.
Dinneen, G. P.
Ziegler, H.
Vanderburgh, A.
Arden, Dean (Barta)
Attridge, W.
Hazel, F.P.
Hosier, W.
Grandy, C.
Buzzard, R.
Holmes, L.
Daggett, N.

Arnow, J.
Zraket, C. A.
Israel, D.
Rich, E.
Bailey, D.
Rising, H. K.
Frachtman, H. E.
Vance, R. R.
Neeb, Donna (RAND)
Thomas, L. M.
Dustin, D. E.
Tritter, A. L.
Briscoe, H.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

TABLE OF CONTENTS

APPENDIX

FIGURES

I INTRODUCTION

The TX-0 computer is a general purpose high-speed machine built primarily of transistors. The computer has one memory device which is a vacuum-tube-driven magnetic-core array capable of storing 1,179,648 bits of information. Each word contains 18 bits for a total of 65,536 or 2^{16} words.[1] The memory cycle time is approximately 6 μsec. The machine performs a complete operation every two memory cycles; the instruction is obtained in the first cycle and the data in the second.[2] Most of the logical and arithmetic operations are executed in the second cycle.

II PRESENT TERMINAL EQUIPMENT

Input

1. A Ferranti photoelectric paper tape reader
   (a) Standard seven-hole flexowriter paper tape
   (b) 200 to 250 lines per second
2. Toggle switch registers
   (a) Toggle switch accmulator called TAC
   (b) Toggle switch buffer register called TBR
   (c) 16 toggle switch registers called toggle switch storage, TSS. These registers can replace the first 16 registers of magnetic core memory by means of a switch on the main console.
3. Flexoprinter input to live register bits 2, 5, 8, 11, 14, 17 setting LR_0 to a one when the key is struck.
4. Provision has been made for a photosensitive device, called the light pen, to control the computer from the display tube.

Output

1. One 12 1/2" cathode ray oscilloscope display tube
   (a) 511 points by 511 points in 7" by 7" array
   (b) a camera will be added in the future
2. Paper tape punch

[1] The TX-2 which is in the process of being constructed will use this memory. The TX-0 will then have a transistor-driven core memory of 2^{13} registers.

[2] Each memory cycle has eight time pulses and the notation we use in refering to them is cycle, time pulse, (i.e., cycle 0, time pulse 8 is written 0.8).

(a) Standard flexowriter tape

3. Standard flexowriter printer

III REGISTERS

1. Memory Buffer Register - (MBR, 18 bits + 1 parity check bit) - receives information from and sends information to the memory. The transfer of information from the memory is checked by means of the parity digit which makes the sum of all 19 digits odd.

2. Accumulator - (AC, 18 bits) - stores the results of numerical operations - is also used as buffer to in-out terminal equipment. The bits of AC are numbered from left to right, 0 to 17.

   One interesting point with regard to the AC is that one may look upon it as strictly a ring adder. If we consider the leftmost digit as a sign, then the largest representable number is $2^{18}-1$ and the smallest is $-2^{-18}+1$. If a one is added to the largest number the result is the smallest and likewise if a one is subtracted from the smallest the result is the largest. There is no overflow alarm. This feature has already been found to be useful in decision techniques.

3. Memory Address Register - (MAR, 16 bits) - selects the information in the memory and has another special feature of selecting operate class commands - (more about this later).

4. Program Counter - (PC, 16 bits) - is used by control and contains the address of the next instruction to be executed.

5. Instruction Register - (IR, 2 bits) - contains the operation part of the instruction which is to be executed.

6. Live Register - (LR, 18 bits) - may be considered as just another storage register which uses flip-flop rather than magnetic cores. It is referred to by means operate class commands which we shall see later.

7. Toggle Switch Buffer Register - (TBR, 18 toggle switches) - used for manual intervention in the normal and test modes.

8. Toggle Switch Accumulator - (TAC, 18 toggle switches) - used for manual intervention in the normal and test modes.

For a description of the flip-flops and logical control, see Figure 6.

IV INSTRUCTIONS AND OPERATING MODES

The first two bits of the 18 bit TX-0 word designate one of four basic instructions. The machine recognizes which one to perform by means of two flip-flops IR_0 and IR_1 called the instruction register. The remaining 16 bits of three of the instructions are used to specify a memory location. The fourth instruction makes use of its remaining 16 bits to designate one or more special commands. These are called operate class commands and are the means by which TX-0 attains its versatility. (As we shall see in section VIII and IX).

TX-0 has three operating modes: Normal, Test, and Read-In. They are specified by two flip-flops, R and T called the mode register. The four instructions are carried out in one of the three modes and for each of the twelve combinations a different function is executed by the machine. The console has a push button to select the Test mode and also one for the Read-In mode. The Normal mode is initiated by instructions in the other two modes. In the Normal mode instruction words are taken from the stored program; in the Test mode, from the TBR; and in the Read-In mode, from the tape being read in.

The mode register (R and T) decodes the modes as follows:

MODE	R	T
Normal	0	0
Test	0	1
Read-In	1	1

V THE NORMAL MODE

The four basic instructions in the Normal mode are interpreted as follows:

IR_0	IR_1	ABBREVIATION	INSTRUCTION
0	0	sto x	Replace the contents of register x with the contents of the AC. Let the AC remain the same.
0	1	add x	Add the word in register x to the contents of the AC and leave the sum in the AC.
1	0	trn x	If the sign digit of the accumulator (AC_0) is negative (i.e. a one) take the next instruction from register x and continue from there. If the sign is positive (i.e. a zero) ignore this instruction and proceed to the next instruction.
1	1	opr x	Execute one of the operate class commands indicated by the number x. (See sections VIII and IX).

VI TEST MODE

The test mode is selected by a push button on the console. Primarily the Test mode was designed into the computer to aid engineers and operators to manually intervene with control and storage for test purposes.

Basically one may consider the test mode as being a one instruction program where the instruction is set in the TBR (Toggle Switch Buffer) and the data to be treated either already in the AC or set in the TAC (Toggle Switch AC). There are two switches on the console which allow a little more versatility to the one instruction. They are called the repeat and step switches. The repeat switch causes the instruction to be repeated over and over again (unless, of course, it is of the transfer control type). The step switch allows the address section of the instruction to be indexed by one each time the instruction is executed.

When the test mode push button on the console is activated (i.e., pushed) the first two digits of the TBR are sent to the IR and the last 16 digits are sent to the MAR. (In the sto x case the AC is reset according to what is set in the TAC). The PC is set to MAR + 1 and the instruction is executed.

Then if:

Repeat Switch	Step Switch	Operation After Execution of the Instruction
Off	Off	The computer will stop
Off	On	The computer will stop but the MAR will be changed to what is in the PC namely, the preceding MAR + 1 and then the PC will be indexed by 1.
On	Off	The computer will continue to perform the same instruction repeatedly at machine speed.
On	On	The MAR will be changed to what is in the PC, namely the preceding MAR + 1. Then the PC will again be indexed by 1 and the instruction will be executed repeatedly with the address section being stepped up by one each time.

The four basic instructions for the test mode are classified as load, examine, test operate, and start.

"Load"	sto x 0 0	The AC is set to what is in the TAC and then the contents of the AC are stored in register x.

"Examine"	add x 0 1	The contents of register x are added to the AC by means of the MBR. Hence x can be examined in the MBR. The AC could have been "anything" before the instruction so all we can say is that the AC will contain "anything" plus the contents of x.
"Test Operate"	opr x 1 1	Any one of the operate class commands is executed. Stepping means nothing in this instruction.
"Start"	trn x 1 0	Change to normal mode and transfer control to instruction in register x. Stepping and repeating mean nothing in this instruction.

VII READ-IN MODE

The Read-In mode is selected by a push button on the console and causes the photoelectric reader to be activated. As each line of tape passes under the read head, the information in tape positions 1, 2, 3, 4, 5, and 6 is transferred to digital positions 3, 6, 9, 12 and 15 of the AC. Once the first line of information is in the AC, the AC is cycled to the right one digital position. The second line is then read in, the AC cycled again one position, and the third line read in. At this point the first three lines are now assembled as a word in the AC. The tapes to be used by the Read-In mode have been made so that each word to be stored follows an instruction word on tape which will perform the storage. In order to transfer control to inner storage all that is required on tape is the transfer instruction itself and it will not be followed by the usual three lines of data as the store instruction is. Getting back to the mechanics of the read-in, the first three lines of information have been read in and assembled in the AC. Since this word will be an instruction in either the storage or the transfer case, the first two digits in the AC are transferred to the IR (Instruction Register) and the last 16 digits to the MAR. At this point the instruction register is examined and if the instruction is of the storage type then the next three lines of tape are read in and assembled in the AC and then the instruction is executed. If when the IR was examined the instruction was of the transfer control type then no more information is read in and the transfer control instruction is executed. Summarizing, we can say that each data word requires six lines of tape; the first three indicating where to put it and the last three the word itself; each transfer control instruction requires only three lines of tape containing the instruction itself. The tape layout canbe seen more clearly in Figure 5.

The four basic instructions of the Read-In mode are separated into two types - storage and transfer control.

Type	First IR	Modified IR	Lines of tape	Symbol	Description
Storage	0 0	0 0	6	sto x	Store the word (which was read in behind this instruction) in register x.
Storage	1 1	0 0	6	opr x	When the two digits for opr x (11) are read into IR(in the Read-In mode) they are complemented and therefore opr x = sto x.
Transfer Control	1 0	1 0	3	trn x	When the two digits of trn (10) are read into IR (in the Read-In mode) the computer stops reading from tape; the computer is changed to the normal mode and control is immediately transferred to register x.

Type	First IR	Modified IR	Lines of tape	Symbol	Description
Transfer Control	0 1	1 0	3	add x	When the two digits of add (01) are read into IR (in the Read-In mode) they are complemented and the computer stops. Upon restarting (by pushing the restart button on the console) the computer performs the instruction trn x. Therefore, add x = stop + trn x.

Note, that with a hand punch, instruction-words on the tape can be modified so that st (00) can become add (01) or transfer (10) and either add or transfer can become operate (11). The flexibility allows changes on the tape without preparing a new one.

In actual practice the Read-In mode is used to read a more efficient Read-In program into storage, since a binary tape with a store instruction following each word would be extremely large and cumbersome. A description of this program will be found in section X. It is called the Input Routine and further describes how data is put into storage and also gives the reader an example of TX-0 programming.

VIII OPERATE CLASS COMMANDS

The following is a list of the operate class commands, the time pulse on which they are executed, the binary form they assume, what they do and the octal notation of the last 16 bits of the operate instruction, opr x.

IR		MAR																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
		2																
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= opr 100,000 (octal)
(0.8) CLL = Clear the left nine digital positions of the AC																		
			3															
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= opr 40,000 (octal)
(0.8) CLR = Clear the right nine digital positions of the AC																		
				4	5													
1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	= opr 20,000 (octal)
(0.8) IOS In-Out Stop = Stop machine so that an In-Out command (specified by digits 6 7 8 of MAR) may be executed.																		
				4	5													
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	= opr 30,000 (octal)
(1.8) Hlt = Halt the computer																		
						6	7	8										
1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	= opr 7,000 (octal)
(0.8) P7H = Punch holes 1-6 in flexo tape specified by AC digital positions 2, 5, 8, 11, 14, and 17. Also punch a 7th hole on tape.																		
						6	7	8										
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	= opr 6,000 (octal)
(0.8) P6H = Same as P7H but no seventh hole																		

```
 IR                              MAR
0 1  2  3 4 5  6 7 8  9 10 11  12 13 14  15 16 17
```

```
                6 7 8
1 1 0  0 0 0    1 0 0   0 0 0   0 0 0   0 0 0 = opr 4,000 (octal)
```

(0.8) PNT = Print one flexowriter character specified by AC digits 2, 5, 8, 11, 14, and 17.

```
                6 7 8
1 1 0  0 0 0    0 0 1   0 0 0   0 0 0   0 0 0 = opr 1,000 (octal)
```

(0.8) R1C = Read one line of flexo tape so that tape positions 1, 2, 3, 4, 5, and 6 will be put in the AC digital positions 0, 3, 6, 9, 12 and 15.

```
                6 7 8
1 1 0  0 0 0    0 1 1   0 0 0   0 0 0   0 0 0 = opr 3,000 (octal)
```

(0.8) R3C = Read one line of flexo tape into AC digits 0, 3, 6, 9, 12, and 15. Then cycle the AC one digital position; read the next line on tape into AC digits 0, 3, 6, 9, 12 and 15, cycle the AC right one digital position and read the third and last line into AC digits 0, 3, 6, 9, 12 and 15. (This command is equal to a triple CYR-R1C.)

```
                6 7 8
1 1 0  0 0 0    0 1 0   0 0 0   0 0 0   0 0 0 = opr 2,000 (octal)
```

(0.8) DIS = Intensify a point on the scope with x and y coordinates where x is specified by AC digits 0-8 with digit 0 being used as the sign and y is specified by AC digits 9-17 with digit 9 being used as the sign for y. The complement system is in effect when the signs are negative.

```
                        9 10
1 1 0  0 0 0    0 0 0   1 0 0   0 0 0   0 0 0 = opr 400 (octal)
```

(1.4) SHR = Shift the AC right one place, i.e. multiply the AC by 2^{-1}

```
                        9 10
1 1 0  0 0 0    0 0 0   1 1 0   0 0 0   0 0 0 = opr 600 (octal)
```

(1.4) CYR = Cycle the AC right one digital position (AC_{17} will become AC_0)

```
                        9 10
1 1 0  0 0 0    0 0 0   0 1 0   0 0 0   0 0 0 = opr 200 (octal)
```

(1.3) MLR = Store the contents of the MBR (memory buffer register) in the live reg.

```
IR                               MAR
0 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17
```

```
                                 11                  15
1 1  0  0  0  0  0  0  0  0  0   1   0   0   0   0   0   0  = opr 100 (octal)
```

(1.1) PEN = Read the light pen flip-flops 1 and 2 into AC_0 and AC_1.

```
                                 11                  15
1 1  0  0  0  0  0  0  0  0  0   0   0   0   0   1   0   0  = opr 4 (octal)
```

(1.1) TAC = Insert a one in each digital position of the AC wherever there is a one in the corresponding digital position of the TAC.

```
                                     12
1 1  0  0  0  0  0  0  0  0  0   0   1   0   0   0   0   0  = opr 40 (octal)
```

(1.2) COM = Complement every digit in the accumulator

```
                                         13
1 1  0  0  0  0  0  0  0  0  0   0   0   1   0   0   0   0  = opr 20 (octal)
```

(1.4) PAD = Partial add AC to MBR, that is, for every digital position of the MBR that contains a one, complement the digit in the corresponding digital position of the AC. This is also called a half add.

```
Example:   AC = 1 0 1 0 1 0 1
          MBR = 0 1 1 1 0 0 0
          -------------------
     New   AC = 1 1 0 1 1 0 1
```

```
 IR                        MAR
 0 1  2  3 4 5  6 7 8  9 10 11  12 13 14  15 16 17
```

```
                                          14
1 1 0  0 0 0  0 0 0  0 0 0  0 0 1  0 0 0 = opr 10 (octal)
```

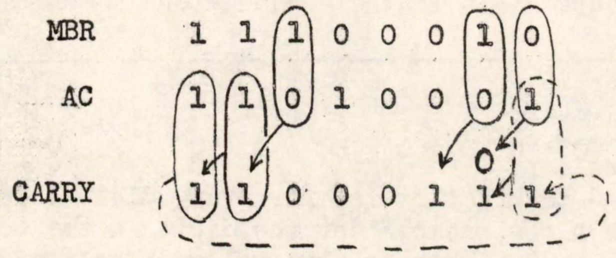
(1.7) CRY = Partial add the 18 digits of the AC to the corresponding 18 digits of the carry.

To determine what the 18 digits of the carry are, use the following rule:

"Grouping the AC and MBR digits into pairs and proceeding from right to left, assign the carry digit of the next pair to a one if in the present pair MBR = 1 and AC = 0 or if in the present pair AC = 1 and carry 1.

(Note: The 0th digit-pair determines the 17th pair's carry digit)

Example:



```
MBR     1 1 1 0 0 0 1 0

AC      1 1 0 1 0 0 0 1
                    0
CARRY   1 1 0 0 0 1 1 1

CARRY   1 1 0 0 0 1 1 1

   AC   1 1 0 1 0 0 0 1
        _______________
New AC  0 0 0 1 0 1 1 0
```

```
                                           16 17
1 1 0  0 0 0  0 0 0  0 0 0  0 0 0.  0 0   1 = opr 1
```

(1.2) AMB = Store the contents of the AC in the MBR.

```
                                           16 17
1 1 0  0 0 0  0 0 0  0 0 0  0 0 0.  0 1   1 = opr 3
```

(1.2) TBR = Store the contents of the TBR in the MBR.

```
                                           16 17
1 1 0  0 0 0  0 0 0  0 0 0  0 0 0.  0 1   0 = opr 2
```

(1.3) LMB = Store the contents of the LR in the MBR.

It should be noticed that the command CYL or cycle left was not listed. The reason for that is:

$$\text{CYL} = \overset{(1.2)}{\text{AMB}},\ \overset{(1.4)}{\text{PAD}},\ \overset{(1.7)}{\text{CRY}}$$

Example

```
                     ⎧    AC = 0  0 1 1 0 0 1 1
After AMB (1.2)      ⎨
                     ⎩   MBR = 0  0 1 1 0 0 1 1

                     ⎧   MBR = 0  0 1 1 0 0 1 1
After PAD (1.4)      ⎨    AC = 0  0 0 0 0 0 0 0
                     ⎩ CARRY = 0  1 1 0 0 1 1 0

After CRY (1.7)           AC = 0  1 1 0 0 1 1 0
```

This is an excellent example of how a programmer can accomplish many things with one operate instruction. Also notice that the AC was cleared by AMB + PAD. Any operate instruction-word is capable of having a large variety of commands within itself as long as the programmer is aware of the time pulse sequence. The preceding list of commands also lists the cycle and time pulse of each command. We have included a TX-0 logical flow chart in this memo with a few side remarks on the chart to assist you in reading it. Whenever there is some question as to what is happening on each time pulse this chart should give you the answer. If you find in experimenting with the operate class commands that a new operate class command would be useful to a programmer, we will be glad to consider your suggestions.

IX COMBINATIONS OF OPERATE CLASS COMMANDS

The following list of combinations has already been found to be useful in programming. A conversion program which will be described in a later memo is capable of assembling most of these combinations using three letter mnemonic symbols (e.g. lac, alr, lad, etc.).

```
0.8   0.8
CLL + CLR = opr 140,000 = clear the AC (CLA)

1.2   1.4   1.7
AMB + PAD + CRY = opr 31 = cycle the AC left one digital position (CYL)

0.8   0.8   1.2
CLL + CLR + COM = opr 140,040 = clear and complement AC (CLC)

0.8   0.8
IOS + DIS = opr 22,000 = Display (this combination was included to remind
                         you that with every in-out command the IOS must be included)
                         (DIS)

0.8   0.8   0.8
IOS + CLL + CLR = opr 160,000 = In out stop with AC cleared.

0.8   0.8   1.4
IOS + P7H + CYR = opr 27,600 = Punch 7 holes and cycle AC right.

0.8   0.8   1.4
IOS + P6H + CYR = opr 26,600 = Punch 6 holes and cycle AC right.

0.8   0.8   0.8   0.8
IOS + CLL + CLR + P6H = opr 166,000 = Clear the AC and punch a blank
                                      space on tape.

0.8   0.8   0.8
IOS + PNT + CYR = opr 24,600 = Print and cycle AC right.

0.8   0.8   1.2   1.4
IOS + P7H + AMB + PAD = opr 27,021 = Punch 7 holes and leave AC cleared.

0.8   0.8   1.2   1.4
IOS + P6H + AMB + PAD = opr 26,021 = Punch 6 holes and leave AC cleared.

0.8   0.8   1.2   1.4
IOS + PNT + AMB + PAD = opr 24,021 = Print and leave AC cleared.

0.8   0.8   0.8
CLL + CLR + RIC = opr 141,000 = Clear AC and start petr running (notice
                                no IOS - which means computer hasn't stopped to wait
                                for information).
```

```
0.8   1.2   1.4   1.7
RIC + AMB + PAD + CRY = opr 1,031 = Start petr running and cycle AC left.

0.8   0.4
RIC + CYR = opr 1,600 = Start petr running and cycle right.

0.8   0.8   0.8   0.8
CLL + CLR + IOS + R3C = opr 163,000 = Clear AC, read 3 lines of tape.

0.8   0.8   0.8   0.8
CLL + CLR + IOS + RIC = opr 161,000 = Clear AC and read one line of tape.

0.8   0.8   0.8   0.8   1.4   1.7
CLR + CLR + IOS + RIC + PAD + CRY = opr 161,031 = Read 1 line of tape and
                                                  cycle AC left.

0.8   0.8   0.8   0.8   1.4
CLL + CLR + IOS + RIC + CYR = opr 161,600 = Read one line of tape and
                                            cycle right.

0.8   0.8   1.1
CLL + CLR + TAC = opr 140,004 = Put contents of TAC in AC.

1.4   1.7
PAD + CRY = opr 30 = Full-add the MBR and AC and leave sum in AC.

0.8   0.8   1.3   1.4
CLL + CLR + LMB + PAD = opr 140,022 = Clear the AC - store LR contents in
                                      memory buffer register - add memory
                                      buffer to AC - i.e. store live reg.
                                      contents in AC. (LAC)

1.2   1.3
AMB + MLR = opr 201 = Store contents of AC in MBR, store contents of MBR
                      in LR, i.e. store contents of AC in LR. (ALR)

1.3   1.4
LMB + PAD = opr 22 = Store contents of LR in MBR, partial add AC and MBR
                     i.e. partial add LR to AC. (LPD)

1.3
MLR = opr 200 = Since MLR alone will have a clear MBR, this is really
                clear LR. (LRO)

1.3   1.4   1.7
LMB + PAD + CRY = opr 32 = Full-add the LR to the AC. (LAD)

0.8   0.8   1.3   1.4
CLL + CLR + TBR + PAD = opr 140,023 = Store contents of TBR in AC.
```

X PROGRAM EXAMPLE

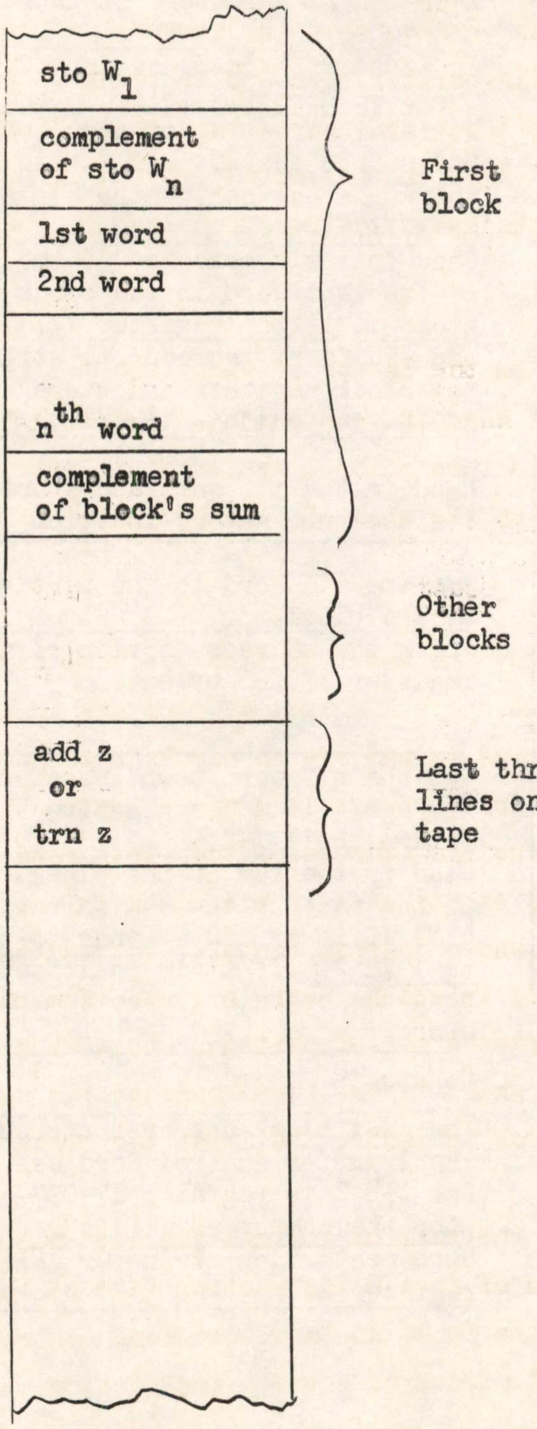
This section was included to give the reader an example of a TX-0 program. The program which was chosen is used to read binary tapes into storage and is called the Input Routine. It was written to avoid the long and cumbersome tapes which would be required by the Read-In mode (a store instruction for each data word). When the conversion program has finished converting a program's flexowriter tape and is ready to punch a binary tape, it first punches the Input Routine on tape in the form that is required by the Read-In mode. Then the converted program is punched out in binary form according to the specifications required by the Input Routine.

By having the Input Routine on the leader of each tape all that is required is the activation of the Read-In push button. The Input Routine is read in by the Read-In mode and then control is immediately transferred to the Input Routine which takes on the task of reading in the rest of the binary tape.

The specifications required by the Input Routine are very simple. The tape channel positions of a word are the same as they are in the Read-In mode. Words are transferred to storage in blocks of sequentially addressed words. The first word in the block is a store instruction word whose address section contains the address of the first word in the block. (Call it sto W_1.) The second word in the block is the complement of a store instruction word whose address contains the address of the last word in the block. (Call it sto W_n where n = no. of words.) The data words follow these two pieces of information. Following the last data word of the block is a word which is the complement of the sum of all the preceding words in the block including the first two control words.

The address of the starting instruction follows the last block of data words. If it is in the form of an add instruction (add z) the computer will be stopped before the Input Routine transfers control to the program. If it is in the form of a transfer control instruction (trn z) then the program will be started immediately after the last block of data words has been read into storage.

TAPE FORMAT REQUIRED BY INPUT ROUTINE



W_1 = Address of 1st word in block

W_n = Address of n^{th} word

trn z = Upon completion of last block transferred to storage take first instruction from register z.

Add z = Stop computer after last block has been transferred to storage. After restart, take first instruction from register z.

INPUT ROUTINE

177741	Temporary storage	Partial sum of block
177742	Add 177773	If the preceding block's sum is correct, go on to next block or transfer control word. If not, go to 177772 and stop computer.
177743	trn 177772	
Enter 177744	opr 163,000 (R3C)	Read in the first word of a block or the transfer control word (add Z or trn Z) and store it in register 177756.
177745	sto 177756	
177746	trn 177756	Is it st W_1, add Z, or trn Z? If trn Z go directly to register 177756.
177747	add 177774	It is either st W_1 or add Z; add 200,000 to the AC. If it was add Z, the AC is now neg. (=trn Z), so go to 177775.
177750	trn 177775	
177751	opr 163,000 (R3C)	Read in the complement of the address of the last word in the block and store it in the register 177777.
177752	sto 177777	
177753	add 177756	Add the first two control words of the block together and store in 177741 to initiate the partial sum.
177754	sto 177741	
177755	opr 163,000 (R3C)	Read in the i^{th} word and store it in its assigned memory location
177756	(sto Wi) (add Z) (Not used) (trn Z)	
177757	add 177741	Add the i^{th} word to the partial sum of the block.
177760	sto 177741	
177761	opr 140,000 (CLA)	Index the address section of the register 177756 by one.
177762	add 177756	
177763	add 177773	
177764	sto 177756	
177765	add 177777	Has the n^{th} word been transferred to storage? If AC is negative - no, return to 177755.
177766	trn 177755	
177767	opr 163,000 (R3C)	Read in the sum of the block. Is it the same as the sum in register 41? If it is, AC = minus zero, go to 177742.
177770	add 177741	
177771	trn 177742	
177772	opr 30,000 (HLT)	If it is positive.... stop the computer. The sum check is wrong.
177773	1	Constants
177774	200,000	
177775	sto 177777	The last block has been stored and the transfer control word was add Z. Put trn Z in register 177777 and stop the computer.
177776	opr 30,000 (HLT)	
177777	trn Z (or the complement of the address of the last word in a block).	Upon restarting, transfer control to register Z.

The operate class commands used in the Input routine were:

opr 160,000 = CLL + CLR + IOS + R3C

Clear AC and read three lines, cycling each time so that they are assembled as an 18 bit word in the AC.

opr 140,000 = CLL + CLR = CLA

Clear both halves of AC.

opr 30,000 = Halt the computer.

It should be noticed that a trn instruction (10) has a one in the sign digital position. In registers 177744, 45, and 46 when the transfer control word "trn Z" is read into the accumulator, the trn 177756 will transfer control to 177756. Since register 177756 will contain trn Z and the AC still contains trn Z, control will immediately be sent to register Z. This is a useful trick. (For example, transfering control to a subroutine with the exit word in the AC).

One other point of interest, if the word add Z is in the AC when instruction trn 177756 in register 177746 is performed, the AC is positive and the next instruction will be add 177774. This will cause the octal number 200,000 to be added to the AC and since the first two bits of the word add Z are 01, the result will be trn Z. This causes the instruction trn 177775 to be executed and 177775 will store the word trn Z in register 177777. The next instruction is the operate class command halt. Since the AC is not disturbed, it will still contain trn Z. If the restart push button is activated, the trn Z in register 177777 will transfer control to register Z.

XI TOGGLE SWITCH STORAGE

The TX-0 has an auxiliary memory system consisting of sixteen toggle-switch registers which we shall refer to as toggle switch storage, TSS. The TSS can be used as a substitute for the first sixteen magnetic core registers 0 through 17. All sixteen registers of TSS can replace core registers 0 through 17 or they can be chosen individually to replace their respective core registers, i.e. TSS_6 can replace register 6 of core memory while the other fifteen can still be core.

The Live Register has been mentioned earlier as an eighteen bit flip-flop register with no address. Up to this point the only way reference could be made to it was by means of the operate class commands. The switches on the TSS panel allow the Live Register to be addressed like any other register. However, its contents can still only be changed by specific operate class commands or by data from the flexo typewriter (if the flexo input switch on the main console is in the on position).

The sixteen registers of TSS are located on the console. (See Figure 4) In addition to the eighteen toggle switches associated with each register there is a toggle switch located to the left of each register which we shall call "cm" and one to the right of each register which we shall call "lr". Also located on the console is a master switch called "core memory select" or CM select. When the CM select switch is on, the first sixteen registers will <u>always</u> be magnetic core. When the CM select switch is <u>off</u> then the first sixteen registers can be either magnetic core, toggle switch storage, or addresses of the live register.

The following is a breakdown of the possible combinations:

CM Select Switch = OFF

Reg.	cm	TSS_x	lr	
x	Off	W	Off	**Case One** Register x is TSS and the word in x is W which is set by the toggle switches.
x	Off	W	On	**Case Two** x is the address of the LR and the word in x will always be the word in the LR and not the toggle sw setting W.
x	On	W	Off	**Case Three** Register x is magnetic core and the toggle switch setting W means nothing.
x	On	W	On	**Case Four** The core switch cm takes precedence over the lr switch and this case becomes the same as case three.

Note that the Live Register may have one, two, three or sixteen different addresses (0 - 17) or none at all if no lr switch is on.

John T. Gilmore Jr
John T. Gilmore, Jr.

H. Philip Peterson
H. Philip Peterson

JTG,HPP:bac

Attachments:
Appendix A
Appendix B
Appendix C

Fig. 1	A-68266
Fig. 2	A-68264
Fig. 3	A-68265
Fig. 4	A-68263
Fig. 5	A-68405
Fig. 6	E-69059
Fig. 7	D-47243

APPENDIX A

TX-0 Console

1. Push Buttons

(a) Stop

(b) Restart

(c) Read-In

(d) Test

(e) Tape Feed

2. Flip-Flop Indicators

(a) IR Two bit instruction register

(b) C Cycle

(c) RT Mode

(d) MR Memory Read

(e) MI Memory Inhibit

(f) PAR Parity

(g) SS Start Stop

(h) PBS Push Button Synchronizer

(i) IOS In Out Stop

(j) CH Chime Alarm

(k) LP Light Pen Flip-Flops 1 and 2.

(l) PETR Photoelectric reader flip-flops 1, 2, 3 and 4

(m) Alarm Indicator

3. Flip-Flop Registers

(a) MAR

(b) PC

(c) MBR

(d) AC

(e) LR

4. Switches

   (a) Suppress Alarm

   (b) Suppress Chime

   (c) Automatic Restart

   (d) Automatic Read-In

   (e) Automatic Test

   (f) Stop on Cycle Zero

   (g) Stop on Cycle One

   (h) Step

   (i) Repeat

   (j) Printer Input

5. Toggle Switch Registers

   (a) TAC - Toggle switch accumulator

   (b) TBR - Toggle switch buffer register

   (c) TSS - Sixteen toggle switch storage registers

APPENDIX B

Operate Class Command Summary

CLL	(0,8)	opr 100,000	Clear left AC
CLR	(0,8)	opr 40,000	Clear right AC
IOS	(0,8)	opr 20,000	In-out stop
HLT	(1,8)	opr 30,000	Halt
P7H	(0,8)	opr 7,000	Punch 7 holes
P6H	(0,8)	opr 6,000	Punch 6 holes
PNT	(0,8)	opr 4,000	Print
R1C	(0,8)	opr 1,000	Read 1 line
R3C	(0,8)	opr 3,000	Read 3 lines
DIS	(0,8)	opr 2,000	Display
SHR	(1,4)	opr 400	Shift right
CYR	(1,4)	opr 600	Cycle right
MLR	(1,3)	opr 200	MBR → LR
PEN	(1,1)	opr 100	Read light pen
TAC	(1,1)	opr 4	TAC ones → AC
COM	(1,2)	opr 40	Complement AC
PAD	(1,4)	opr 20	Partial ADD MBR and AC
CRY	(1,7)	opr 10	Partial ADD carry digits and AC
AMB	(1,2)	opr 1	AC → MBR
TBR	(1,2)	opr 3	TBR → MBR
LMB	(1,3)	opr 2	LR → MBR

APPENDIX C

OPERATE CLASS COMMAND COMBINATION SUMMARY

opr	140,000	=	Clear AC -- Clear AC	(CLA)
opr	31	=	Cycle left	(CYL)
opr	140,040	=	Clear and complement AC	(CLC)
opr	22,000	=	Display	
opr	160,000	=	In out stop and AC cleared	
opr	27,600	=	Punch 7 holes, cycle AC right	
opr	26,600	=	Punch 6 holes, cycle AC right	
opr	166,000	=	Clear AC - Punch blank tape	
opr	24,600	=	Print and cycle AC right	
opr	27,021	=	Punch 7 holes and clear AC	
opr	26,021	=	Punch 6 holes and clear AC	
opr	24,021	=	Print and leave AC cleared	
opr	141,000	=	Clear AC and start PETR running	
opr	1,031	=	Start PETR running and cycle left	
opr	1,600	=	Start PETR running, cycle right	
opr	163,000	=	Clear AC and read 3 lines of tape	
opr	161,000	=	Clear AC and read 1 line of tape	
opr	161,031	=	Read 1 line of tape and cycle left	
opr	161,600	=	Read 1 line of tape and cycle right	
opr	140,004	=	TAC → AC	
opr	30	=	Full add MBR and AC	
opr	140,022	=	LR → AC	(LAC)
opr	201	=	AC → LR	(ALR)
opr	22	=	Partial add LR and AC	(LPD)
opr	200	=	Clear LR	(LRO)
opr	32	=	Full add LR and AC	(LAD)
opr	140,023	=	TBR → AC	(TBR)

A-68266

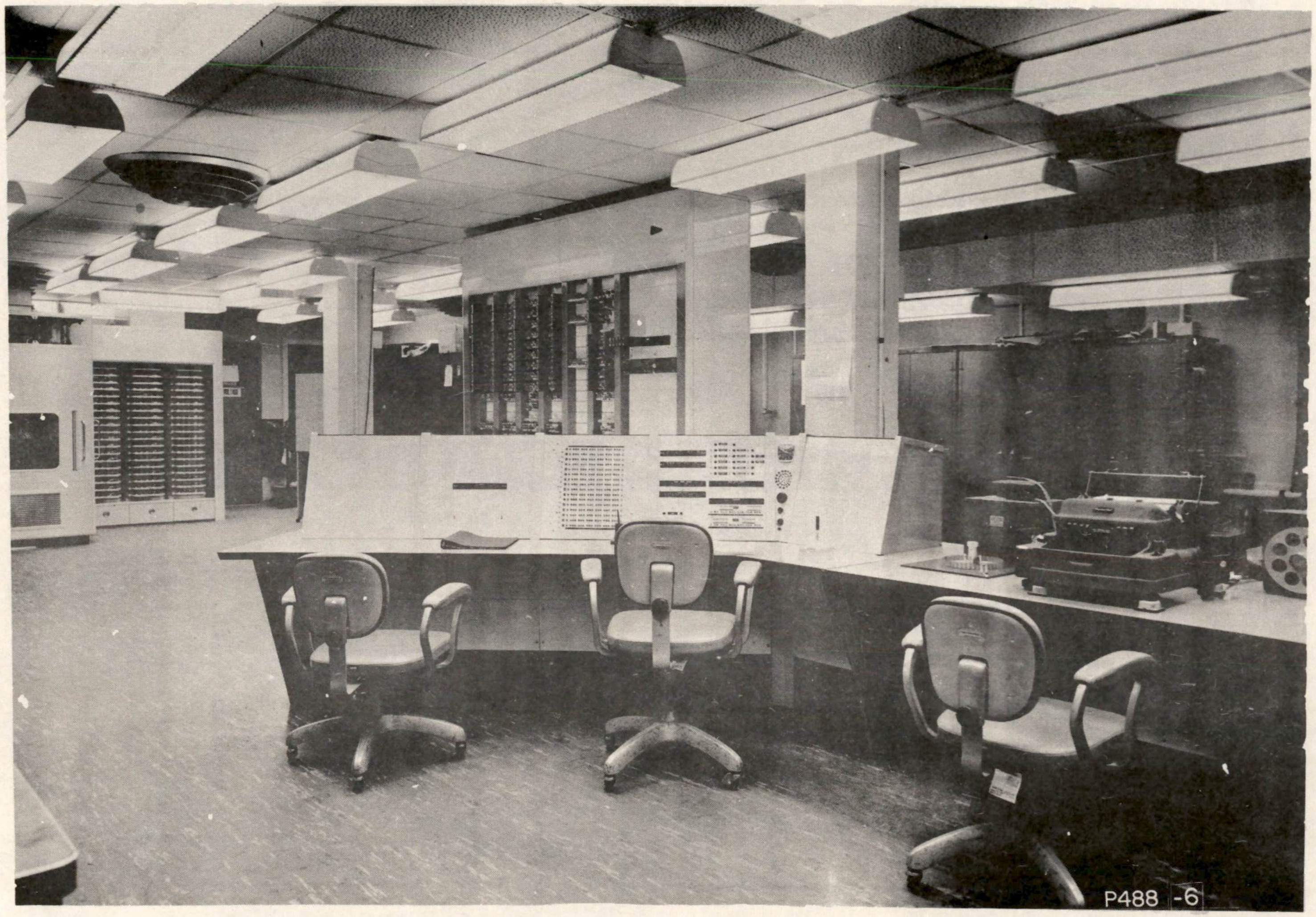


FIG. 1
TX-0 COMPUTER ROOM

A-68264

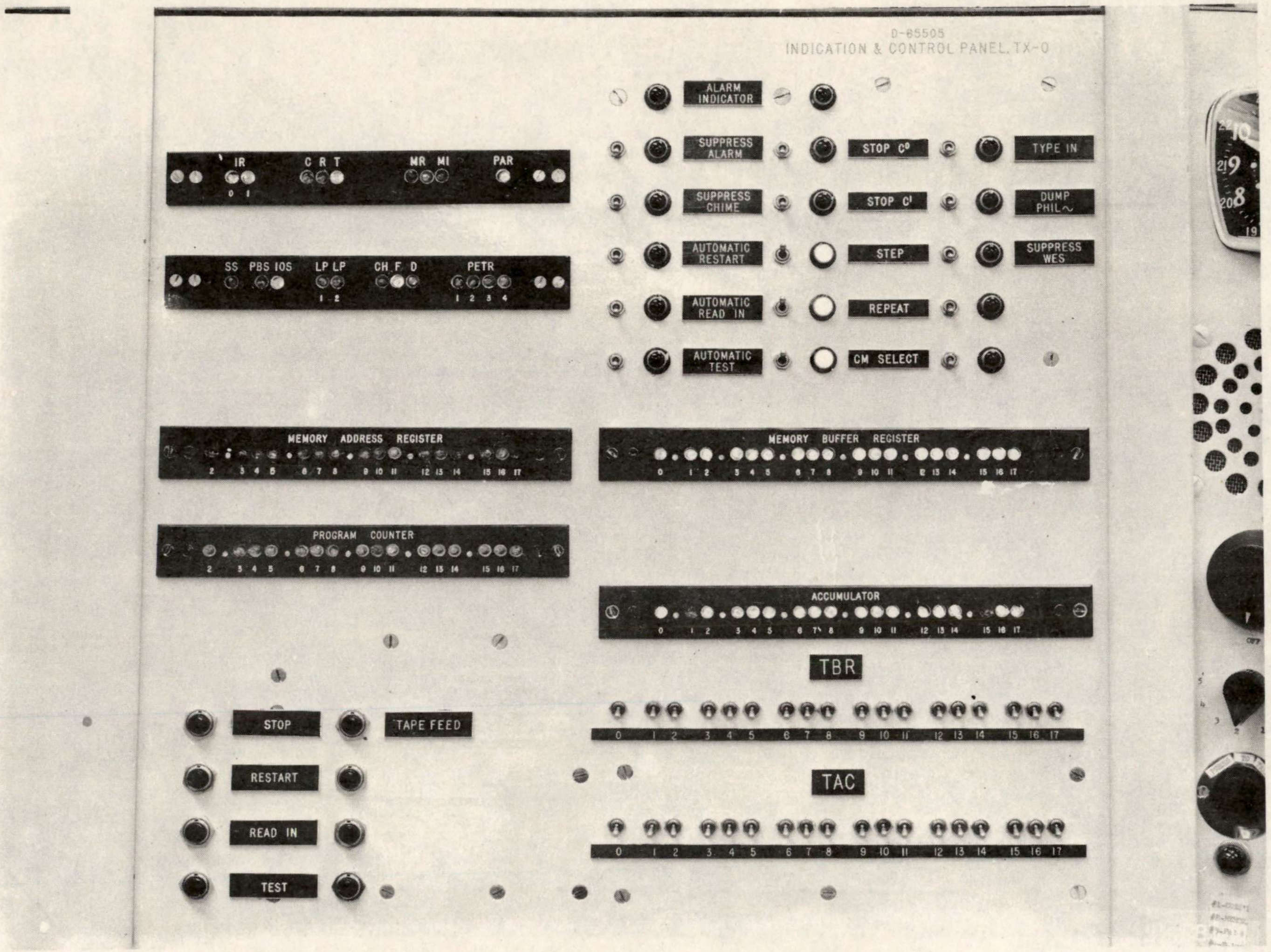


FIG. 3
TX-0 MAIN CONSOLE PANEL

A-68265

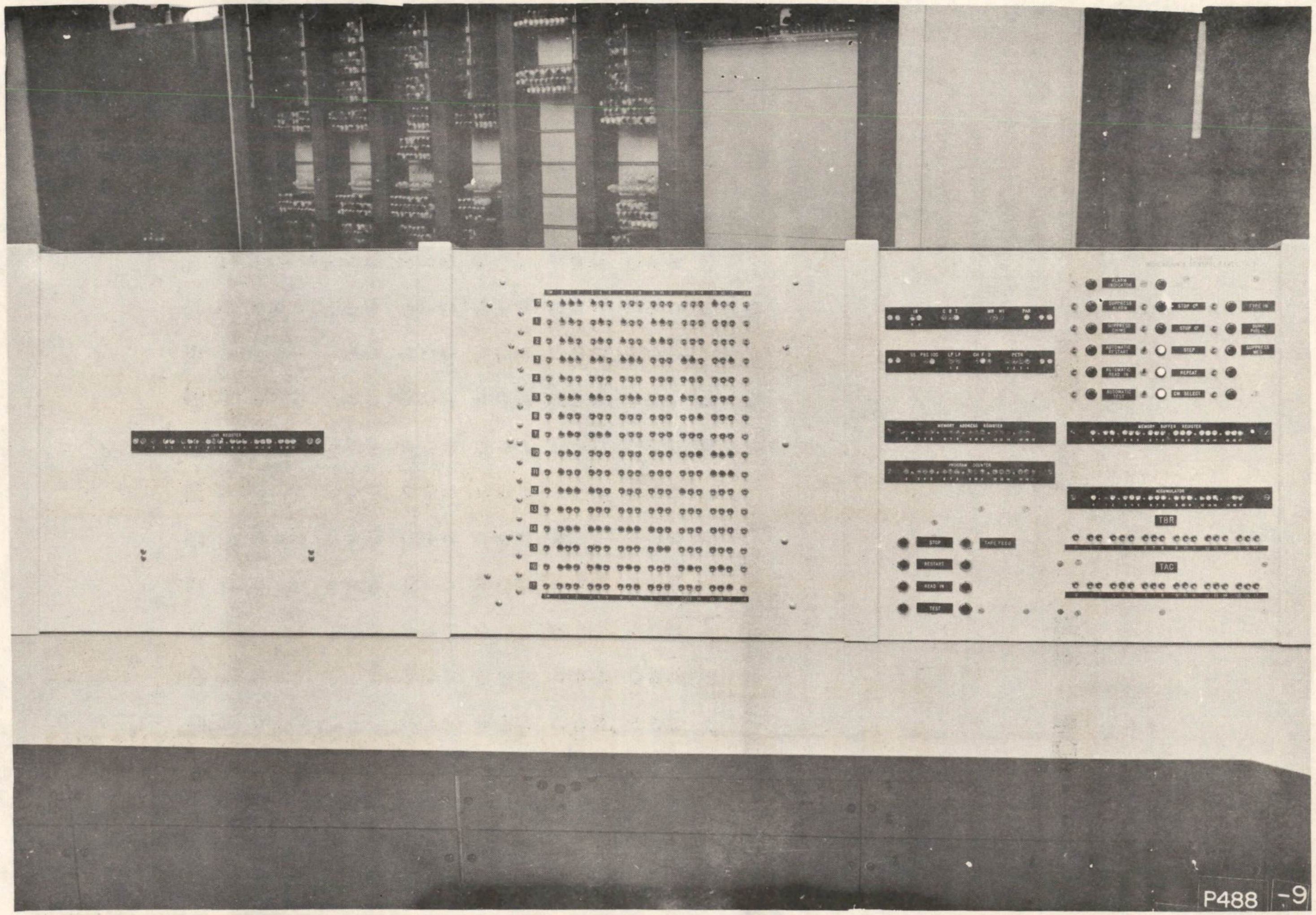


FIG. 2
TX-0 CONSOLE

A-68263

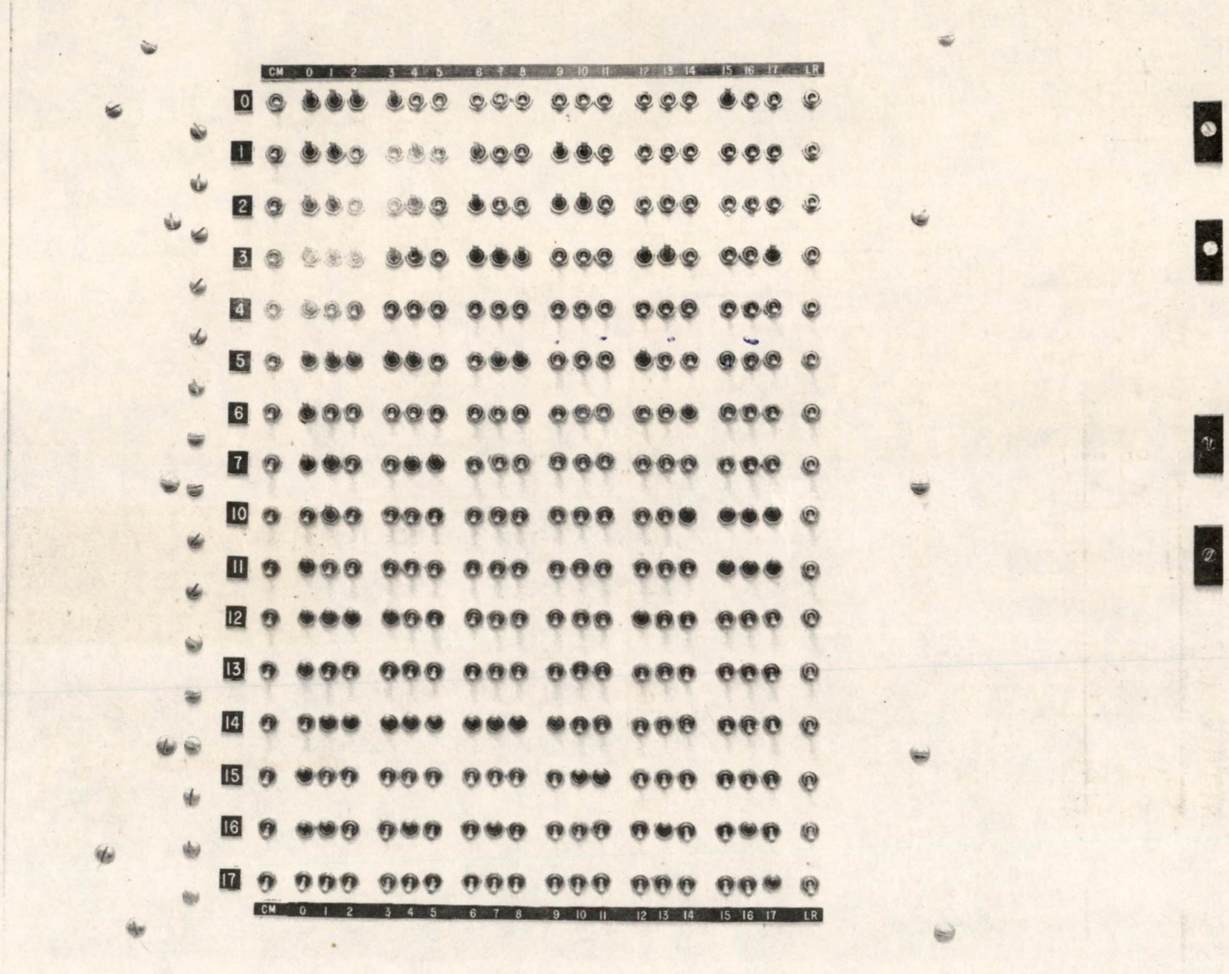
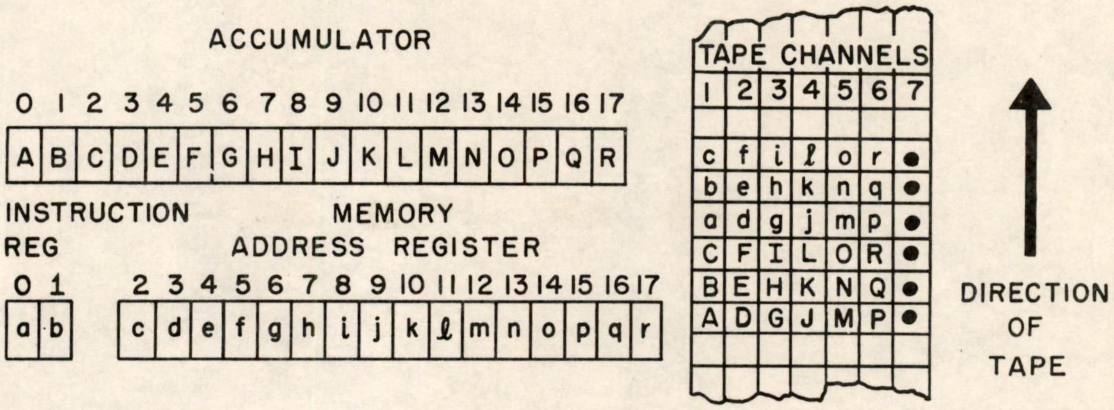


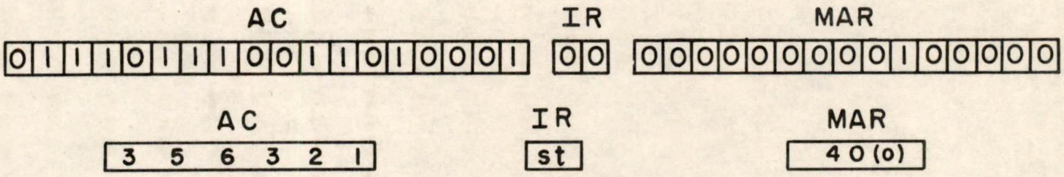
FIG. 4

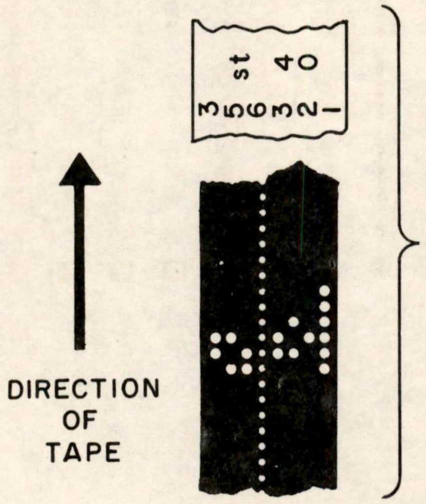
TX-0 TOGGLE SWITCH STORAGE



1	2	3	4	5	6	7
c	f	i	ℓ	o	r	●
b	e	h	k	n	q	●
a	d	g	j	m	p	●
C	F	I	L	O	R	●
B	E	H	K	N	Q	●
A	D	G	J	M	P	●

EXAMPLE: STORE THE OCTAL WORD 356321 IN REGISTER 40 OCTAL



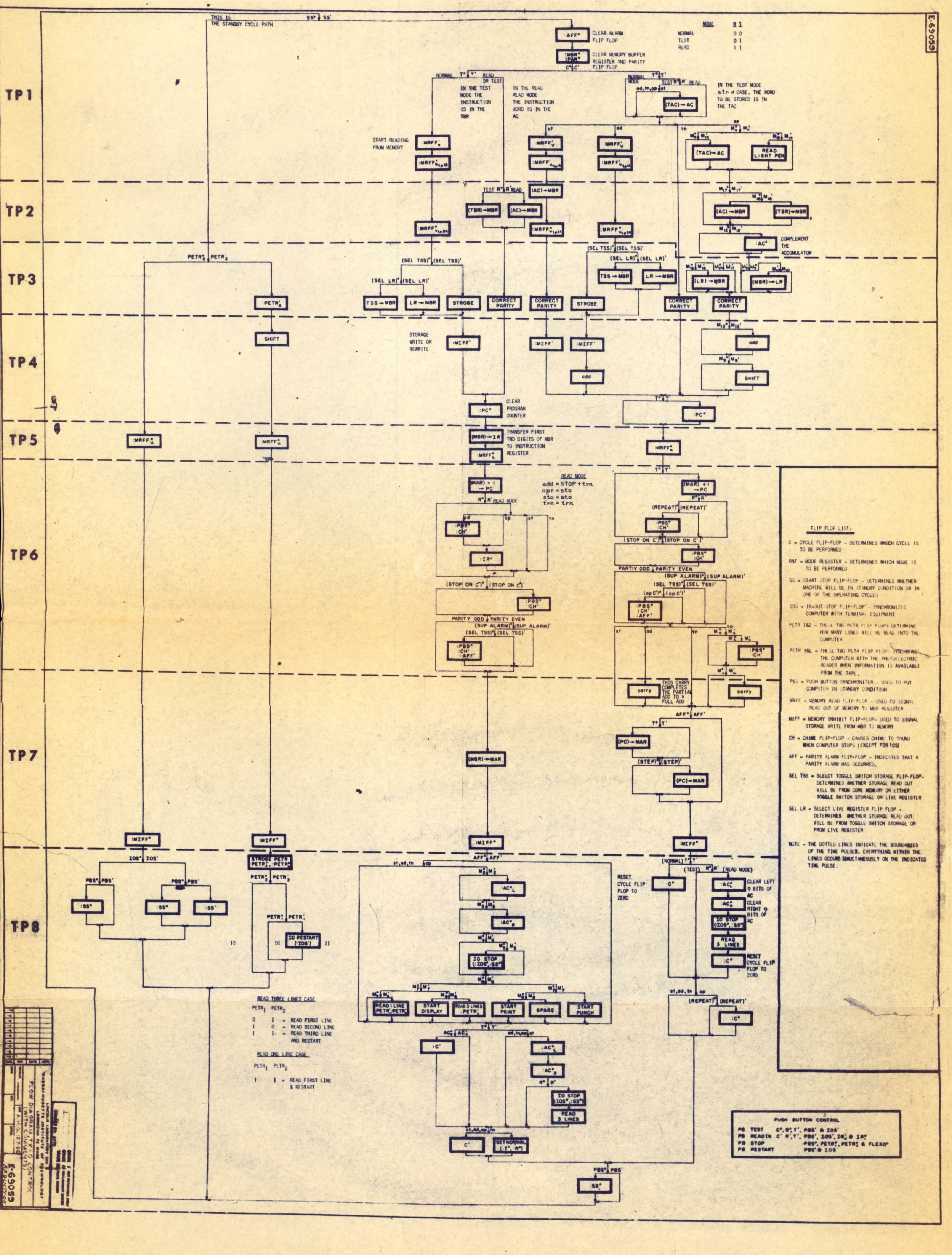


IF THE TAPE IS HELD SO THAT IT IS MOVING FROM LEFT TO RIGHT WITH THE SEVENTH HOLE NEXT TO THE BODY, THE DATA WORD AND STORE INSTRUCTION CAN BE READ OCTALLY IN HORZONTAL FASHION.

FIG. 5

TAPE LAYOUT FOR READ-IN MODE OF TX-0

A-68405



READ THREE LINES CASE

PETR₁	PETR₂	
0	1	= READ FIRST LINE
1	0	= READ SECOND LINE
1	1	= READ THIRD LINE AND RESTART

READ ONE LINE CASE

PETR₁	PETR₂	
1	1	= READ FIRST LINE & RESTART

MODE	R	T
NORMAL	0	0
TEST	0	1
READ	1	1

FLIP FLOP LIST:

C = CYCLE FLIP-FLOP - DETERMINES WHICH CYCLE IS TO BE PERFORMED

R&T = MODE REGISTER - DETERMINES WHICH MODE IS TO BE PERFORMED

SS = START STOP FLIP-FLOP - DETERMINES WHETHER MACHINE WILL BE IN STANDBY CONDITION OR IN ONE OF THE OPERATING CYCLES

IOS = IN-OUT STOP FLIP-FLOP - SYNCHRONIZES COMPUTER WITH TERMINAL EQUIPMENT

PETR 1&2 = THESE TWO PETR FLIP FLOPS DETERMINE HOW MANY LINES WILL BE READ INTO THE COMPUTER

PETR 3&4 = THESE TWO PETR FLIP FLOPS SYNCHRONIZE THE COMPUTER WITH THE PHOTOELECTRIC READER WHEN INFORMATION IS AVAILABLE FROM THE TAPE.

PBS = PUSH BUTTON SYNCHRONIZER - USED TO PUT COMPUTER IN STANDBY CONDITION

MRFF = MEMORY READ FLIP FLOP - USED TO SIGNAL READ OUT OF MEMORY TO MBR REGISTER

MIFF = MEMORY INHIBIT FLIP-FLOP- USED TO SIGNAL STORAGE WRITE FROM MBR TO MEMORY

CH = CHIME FLIP-FLOP - CAUSES CHIME TO SOUND WHEN COMPUTER STOPS (EXCEPT FOR IOS)

AFF = PARITY ALARM FLIP-FLOP - INDICATES THAT A PARITY ALARM HAS OCCURRED.

SEL TSS = SELECT TOGGLE SWITCH STORAGE FLIP-FLOP- DETERMINES WHETHER STORAGE READ OUT WILL BE FROM CORE MEMORY OR EITHER TOGGLE SWITCH STORAGE OR LIVE REGISTER

SEL LR = SELECT LIVE REGISTER FLIP FLOP - DETERMINES WHETHER STORAGE READ OUT WILL BE FROM TOGGLE SWITCH STORAGE OR FROM LIVE REGISTER

NOTE - THE DOTTED LINES INDICATE THE BOUNDARIES OF THE TIME PULSES. EVERYTHING WITHIN THE LINES OCCURS SIMULTANEOUSLY ON THE INDICATED TIME PULSE.

PUSH BUTTON CONTROL

PB	TEST	C°, R°, T', PBS' & IOS'
PB	READIN	C' R', T', PBS', IOS', IR'₀ & IR°₁
PB	STOP	PBS°, PETR°₁, PETR°₂ & FLEXO°
PB	RESTART	PBS' & IOS'

LINCOLN LABORATORY DIV. 6
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON 73 MASS

FLOW DIAGRAM TX-0 CONTROL (WITH COMMENTS)

DR A.M.G. 12-7-56

E-69059

REDUCTION

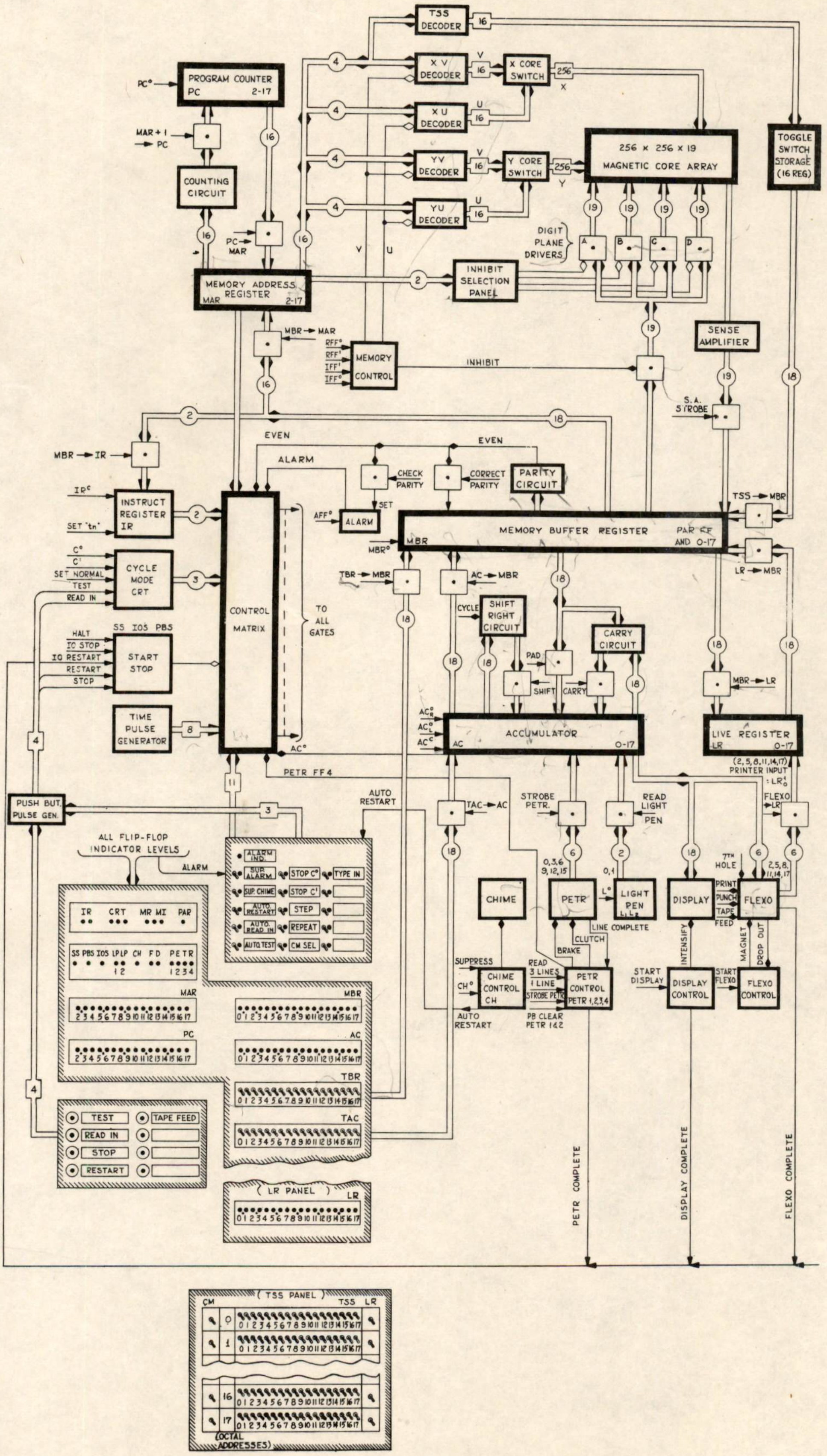


FIG. 7

BLOCK DIAGRAM, TX-0

D-47243

variants. A unique feature of the central computer is its ability to deal with operands in one 36-bit, one 27- and one 9-bit, two 18-bit, or in four 9-bit configurations. These configurations are specified by each instruction -- a feature which permits the 9-bit quarters of the arithmetic element to be connected in various ways to the corresponding quarters of the memory. Control is exercised over the activity of the quarters during the execution of the instruction.

THE LINCOLN TX-2 INPUT-OUTPUT SYSTEM

by James W. Forgie

The Lincoln TX-2 computer design uses the "multiple-sequence program technique" to permit the concurrent operation of a number of input-output devices. A stored program (instruction) counter is associated with each input-output device. The programs referred to by these counters time-share the hardware of the central computer, giving attention to the associated input-output devices as required. A priority system ranks the devices according to speed and type for efficient operation with a minimum of programming restrictions. The multiple-sequence program technique provides an environment in which buffer storage may be considerably reduced at a small cost in machine speed within the limits set by peak and average data rate considerations.

MEMORY UNITS IN THE LINCOLN TX-2

by Richard L. Best

There are three random-access core memories in TX-2 -- all of which may be operated independently and concurrently. Two of these are for conventional storage of data and instructions. The third is used as a file of index registers and program counters. The three types of core memories in TX-2 are described. The largest memory contains 65,536 words 37 digits long, and has a full cycle time of 6 1/2 μsec. The next largest is entirely transistor driven, contains 4,096 words, 37 digits long, and has a 6-μsec cycle time. The smallest and fastest contains 64 words, 19 digits long, and uses external selection and two cores per bit to achieve a "read" cycle time of 1 μsec and a "write" cycle time of 3 μsec. Ferrite cores only 0.050 inches O.D. are used in the two smaller memories.

TX-2 CIRCUITRY

by Kenneth H. Olsen

Only two basic transistor logic circuits are used in TX-2. Surface-barrier transistors in saturated emitter-follower circuits are grouped in parallel for "AND" "OR" operations; groups of inverter circuits are connected in series, parallel, or series-parallel to perform more complicated logic operations. The

Division 6 – Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: The Lincoln TX-2 Computer

To: Distribution List

From: W.A. Clark, J.M. Frankovich, H.P. Peterson, J.W. Forgie, R.L. Best, K.H. Olsen

Date: 1 April 1957

Approved: ______________________
W.N. Papian

Abstract: The information contained in this report was presented at the Western Joint Computer Conference held at Los Angeles, California, in February 1957. The papers given form the sections of this memorandum and are as follows:

THE LINCOLN TX-2 COMPUTER DEVELOPMENT
by Wesley A. Clark

Construction of the TX-2 computer at the Lincoln Laboratory of MIT is part of the Lincoln program for the study and development of large-scale digital computer systems. The Lincoln TX-2 incorporates several new developments in high-speed transistor circuits, large capacity magnetic-core memories, and flexibility in machine organization and is designed to work efficiently with many input-output devices of different types. In the course of development of the TX-2, Lincoln has constructed a small self-checking multiplier system which is on life test, and a complete, though skeletal, general-purpose computer known as the TX-0 which is now in operation.

A FUNCTIONAL DESCRIPTION OF THE LINCOLN TX-2 COMPUTER
by John M. Frankovich and H. Philip Peterson

The Lincoln TX-2 computer is a general-purpose, binary parallel machine with a code of 64 single-address instructions and 64 index registers. The design provides for a random-access memory of 260,000 36-bit words. The instruction code includes the usual arithmetic and logic operations executed at a peak rate of 160,000 36-bit additions per second, with several interesting

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

TX-2 flip-flop is assembled from saturated inverters and emitter-followers and incorporates enough amplification so that it appears to the logic designer as a simple switch over a wide range of loads.

Circuit tolerance to variations in transistor and other component characteristics, in temperature, supply voltages, and noise was studied in detail and designs were adjusted to minimize the effects of them. The study led to the selection of voltage sensitive parameters for indicating the deterioration of components with age and became the basis of the marginal checking system.

ACKNOWLEDGMENT

We are indebted to Mr Frank P. Hazel and Miss June Karlson for their assistance in editing; to Mrs. Barbara Clark, Miss Carole Olson, Miss Elaine Tonra and Miss Ruth McDonald for their typing of the manuscripts; and to Miss Alice Griffin and her associates for the preparation of the many figures.

Distribution List

Division 6

P.R. Bagley
R.L. Best
S. Bradspies
J.H. Burrows
W.A. Clark
L.B. Collins
J.W. Forgie
J.M. Frankovich
S. Goldberg
L.L. Holmes (Barta)
N.T. Jones
G. Marnie
K.H. Olsen
R.B. Paddock
W.N. Papian (3 copies)
H.P. Peterson
E.W. Pughe
A. Rowe (Barta)
W.F. Santelmann
C.A. Zraket

Non-Division 6

A.V. Nedzel
E.D. Thomas
Major H.B. Farmer
R. A. Nelson

Non-Lincoln (inside addresses)

N.E. Trieste (IBM)
A.R. Marshall (RAND)
Major S. Pierce (LRP)

Non-Lincoln (outside addresses)

Miss R. Rita Balogh
IBM, Military Products Div.
Kingston, New York

Carlo Bocciarelli
Philco Research Division
C and Tioga Streets
Philadelphia, Pennsylvania
(2 copies)

Mr. Arnold A. Cohen
Remington Rand UNIVAC
St. Paul 16, Minnesota

Mr. Abraham Katz
Commercial Electronic Products
RCA
Camden, New Jersey

Mr. Steve Levy
Lansdale Tube Co.
Church Rd.
Lansdale, Pennsylvania

Mr. Louis C. Murphy
Marchant Research
717 Los Palos Drive
Lafayette, California

Mr. A.A. Perez
National Cash Register Co.
Electronics Division
1401 East el Segundo Blvd.
Hawthorne, California

Mr. Kenneth M. Rehler, Mgr.
Electronic Control Systems, Inc.
2136 Westwood Blvd.
Los Angeles 25, California

Table of Contents

Table of Contents (Continued)

List of Illustrations

I. THE LINCOLN TX-2 COMPUTER DEVELOPMENT

A. Introduction

The TX-2 is the newest member of a growing family of experimental computers designed and constructed at the Lincoln Laboratory of MIT as part of the Lincoln program for the study and development of large-scale, digital computer systems suitable for control in real time. Although, in general characteristics and design philosophy, it owes a great deal to its predecessors, Whirlwind I and the Memory Test Computer, the Lincoln TX-2 incorporates several new developments in components and circuits, memories, and logical organization. It is the purpose of this paper to summarize these new features and to give some idea of the historical development and general design objectives of the TX-2 program. Fig. 1 shows TX-2 in its present development stage.

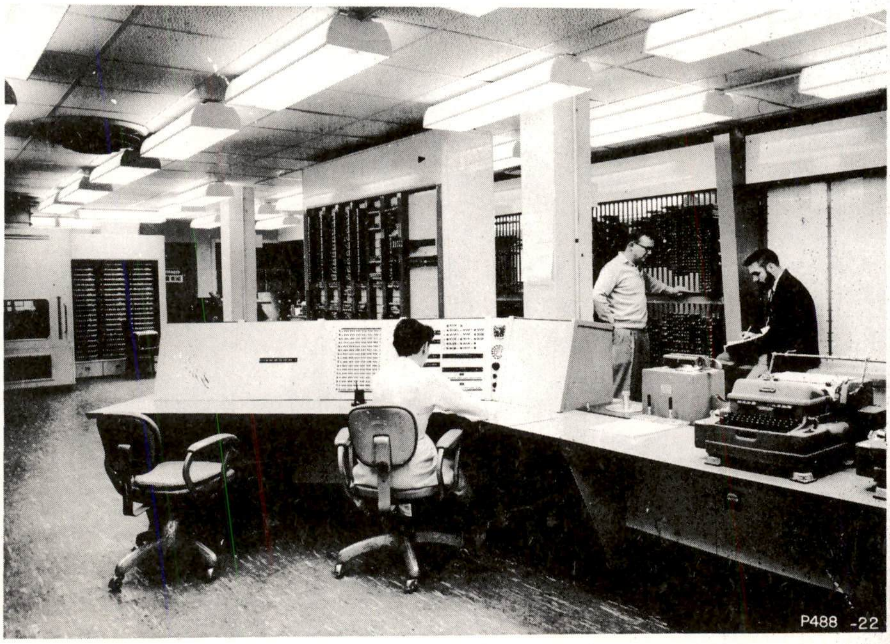


Fig. 1. The Lincoln TX-0 and TX-2 Computers

Foreground: TX-0 console
Middle center: TX-0 central computer frame
Right rear: Partially completed TX-2 frame showing plug-in unit construction
Left rear: The 256 x 256 memory

B. History

With the development by Lincoln and IBM engineers of the SAGE computer for air defense, real-time control computer systems had reached an impressive level of size, sophistication, and complexity. The highly successful 64 x 64 coincident-current, magnetic-core, memory array was in operation in the Memory Test Computer which had given up its earlier 32 x 32 array to Whirlwind. Vacuum tubes abounded in all directions. It was apparent that the further advances in system design which could be made by increasing memory size, eliminating vacuum tubes wherever possible, and organizing input-output buffering, control, and communications into more efficient forms, would be well worthwhile.

The development of a 256 x 256, switch-driven, magnetic-core memory array was begun and the Philco surface-barrier transistor made its appearance. After some very promising bench experiments with flip-flops and logic circuits, it became apparent that this transistor was potentially well-suited to use in large-scale systems and warranted further study. Accordingly, plans were laid for a succession of experimental digital systems of increasing size and complexity which would make possible the development and evaluation of circuits using the surface-barrier transistors, and which would lead to a computer of advanced design that would be capable of making efficient use of the 256 x 256 memory.

A double-rank shift register of eight stages and containing about 100 transistors was constructed and put on life-test in April 1955. It has since been circulating a fixed pattern almost continuously with no known errors and no natural transistor failures.

As the next step, it was decided to build a small, high-speed, error-detecting multiplier and incorporate marginal checking and other system features. The value of a multiplier as a preliminary model had been well demonstrated by the 5-digit system built during Whirlwind's early development. The shift, carry, count, and complement operations, under closely controlled timing conditions, were felt to be representative of all of the operations in the manipulative elements of the type of computer planned. Accordingly, an 8-bit system using 600 transistors was designed and completed in August 1955 and has been in nearly continuous operation since. Operating margins are periodically checked, and in steady state operation, the multiplier's error-rate has been about one every two months or one error per 5 x 10^{11} multiplications at 10^5 multiplications per second. Most of these errors appear to have been caused by cracks in the printed wiring which open intermittently.

During this period, a better idea of the general characteristics of the projected computer began to develop and the engineers who were designing the 256 x 256 memory were encouraged to think in terms of a word of 36 bits. The notion of a logically separate input-output processor was examined and rejected in favor of a minimum buffering scheme in which data is transferred directly to and from the central memory of the computer. The possibility was recognized of programming these transfers by means of additional program sequences and associated program counters, thus taking advantage of the extensive facilities of the central machine itself for processing input-output data.

It was realized that another development step was desirable before attempting such an elaborate 36-bit system. The 8-bit multiplier had produced a certain measure of confidence and familiarity with circuits, packaging, and techniques of logical design, but there remained the problems associated with communicating with memory units and input-output equipment operating at vacuum-tube levels over relatively large distances from a central machine which operated at transistor levels. It appeared that the memory development, which had now entered the construction phase, would also benefit by a preliminary evaluation of the 256 x 256 array and its switching, timing, and noise problems in an operating computer of some kind, possibly with a reduced word length. It was, therefore, decided to design and build next a simple machine - in fact, the simplest reasonable machine - in order to bring about an early intermediate closure of the various efforts within the program.

After some thought about the various possible minimal machines, a design was completed in which the word length would be 18 bits - a graceful half of the projected final form. We began to refer to this computer as the TX-0 and to the projected machine as the TX-2. Because the 256 x 256 memory array required 16 bits for complete addressing, the single-address instruction word of the TX-0 was left with 2 bits in which to encode instructions. The particular set of instructions chosen included three which required a memory address (add, store, and conditional jump) and one which did not. In this last instruction, the remaining 16 bits were used to control certain necessary and useful primitive operations such as clearing and complementing the accumulator, transferring words between registers, and turning on and off input-output equipment.

The TX-0, equipped with a Flexowriter, a paper-tape reader, and a cathode-ray tube display system was completed, except for the memory, in April 1956. Twenty planes of the 256 x 256 memory array were installed the following August and the TX-0, now containing about 3600 transistors and 400 vacuum tubes, began to function as a complete computer. Since that time, it has been used to run a variety of testing and demonstration programs, and a symbolic address compiler and other utility programs have been constructed and are currently in use.

Not only has the TX-0 served the evaluational purposes for which it was built, but it has also demonstrated an effectiveness as a usable computer that is somewhat surprising in view of its simplicity. Its relatively high speed of about 80,000 instructions per second and its 65,536-word memory compensate in large measure for the limitations of its instruction code and logical structure.

With the successful completion of the TX-0, the final steps in the development were undertaken in packaging, circuit refinement, and logical design of the TX-2. A great deal had been learned about the performance of the transistors and memory, the types of logical circuits which are practical, techniques of marginal checking, and the lesser system problems such as color scheme selection and the proper location of pencil sharpeners. As design work progressed, the TX-2 took form as a system of about 22,000 transistors and 600 vacuum tubes. It is an interesting fact that at each step of the development since the shift register, the number of transistors involved was about 6 times the number in the preceding step. This is graphically shown in Fig. 2. At the time of writing approximately 16 million transistor-hours have accumulated in the shift register, multiplier, and TX-0. There have been two natural deaths and a dozen or so violent ones, primarily due to contact shorting with clip leads and probes.

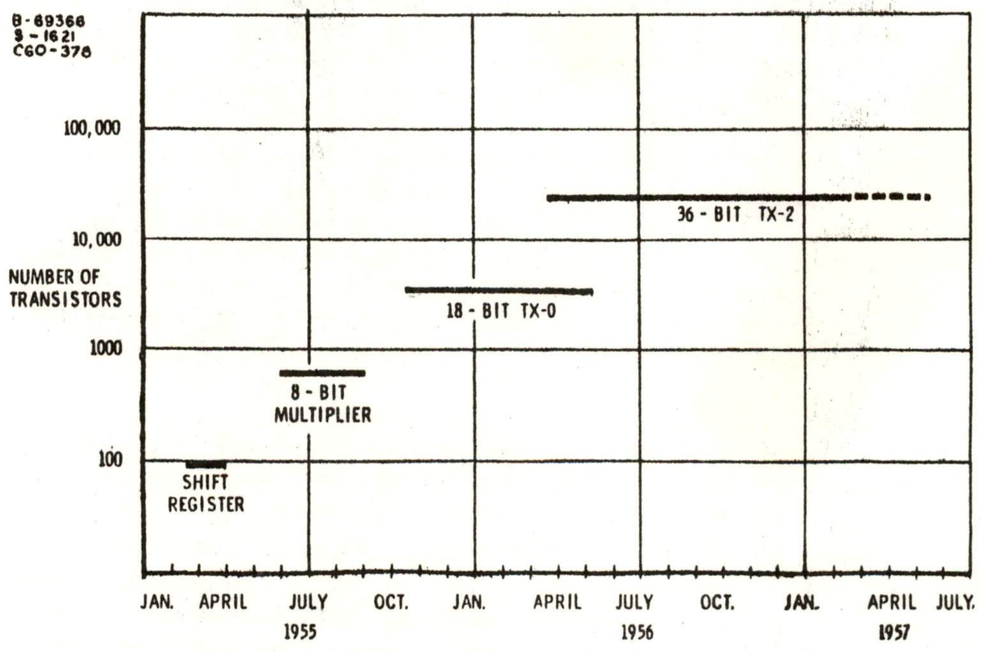


Fig. 2. Steps in the Lincoln TX-2 Development Program

C. Design Objectives

In describing design objectives, it should be pointed out that speed of operation was not the primary consideration to which all other attributes were sacrificed. It would have been possible, at the expense of a few more logic circuits, to increase the speed of multiplication, division, and shift-type operations. Similarly, the operation of the index register system could have been made more efficient at the cost of an additional small, fast memory. The principal objective was rather that of achieving a balance between the factors of speed, reliability, simplicity, flexibility and general virtue.

A key aspect is that of expandability which, in an experimental computer in an active environment, certainly ranks with the foregoing qualities in importance. The address structure in the TX-2 permits an expansion of the memory by about a factor of 4, partly to allow for new memory developments, such as the transistor-driven 64 x 64 array which was begun following the completion of TX-0. New instructions and pieces of terminal equipment will certainly be added during the course of future operation. Extra space and spare plugs have been artfully distributed about in constructing the computer frame. Finally, modular construction will permit a fairly easy physical expansion when required.

The result of all this activity has been a computer of relatively large capability. In addition to incorporating high-speed transistor circuits and a large magnetic-core memory array, the Lincoln TX-2 has two major and distinguishing design characteristics:

1. The structure of the arithmetic element can be altered under program control. Each instruction specifies a particular form of machine in which to operate, ranging from a full 36-bit computer to four 9-bit computers with many variations. Not only is such a scheme able to make more efficient use of the memory in storing data of various word lengths, but it also can be expected to result in greater over-all machine speed because of the increased parallelism of operation.

   Peak operating rates must then be referred to particular configurations. For addition and multiplication, these peak rates are given in the following table:

PEAK OPERATING SPEEDS OF TX-2

Word Lengths (in bits)	Additions per second	Multiplications per second
36	150,000	80,000
18	300,000	240,000
9	600,000	600,000

2. Instead of one instruction counter, the TX-2 has 32 such counters which are assigned separately to different users of the computer, who then compete for operating time from instruction to instruction. A special part of the machine selects a particular user based partly on a predetermined priority schedule and partly on the current needs of that user. This multiple-sequence operation, in which many essentially independent instruction sequences interrupt and interleave one another, is an extension of the breakpoint operation found in DYSEAC of the National Bureau of Standards.

The value of these features will have to be assessed during the course of future machine operation. The features themselves are discussed in more detail in the following sections of this report.

II. A FUNCTIONAL DESCRIPTION OF THE LINCOLN TX-2 COMPUTER

A. Introduction

TX-2 is a large-scale digital computer, designed and built at the MIT Lincoln Laboratory, which uses new memory and circuit components and some new logical design concepts. The computer will be a research tool in scientific computations, data-handling, and real-time problems. The design of the computer reflects not only the characteristics of the components available, but also the nature of the intended applications. This section describes the functions and organization of the computer that are important from the user's point of view.

B. General Structure of TX-2

TX-2 is a parallel, binary computer with words 36 digits long. The internal memory is random-access and will initially consist of 69,632 registers of parity-checked, magnetic-core memory and about 24 additional toggle-switch and flip-flop registers. About 150,000 instructions can be executed per second. Instructions are indexed, single-address type and a fixed-point, signed-fraction, ONE's - complement number system is used.

Several unusual ideas incorporated in the organization of the system reduce the amount of information unnecessarily manipulated during program sequences. Furthermore, the organization facilitates the execution of several operations simultaneously, thereby increasing the effective speed of the computer.

The principal registers and information paths in the computer are illustrated schematically in Fig. 3. A,B,C,D,E,F,M, and N are the 36-bit flip-flop registers in the machine. M and N are memory buffer registers, each of which has a parity flip-flop and associated circuitry used to check the parity of memory words. P,Q, and X are 18-digit registers; X also has a parity digit which is used to check the parity of words in the X-memory. Control flip-flops are not shown in Fig. 3.

Instructions are full memory words and are placed in the control element during the instruction memory cycle. During the operand memory cycle, an operand is usually transmitted between the memory element and some other element--always through the exchange element. The 36-digit configuration of the memory is not, however, maintained throughout the computer during operation timing. A programmer can, in effect, simultaneously control several independent, with operands of shorter word lengths during the execution of each

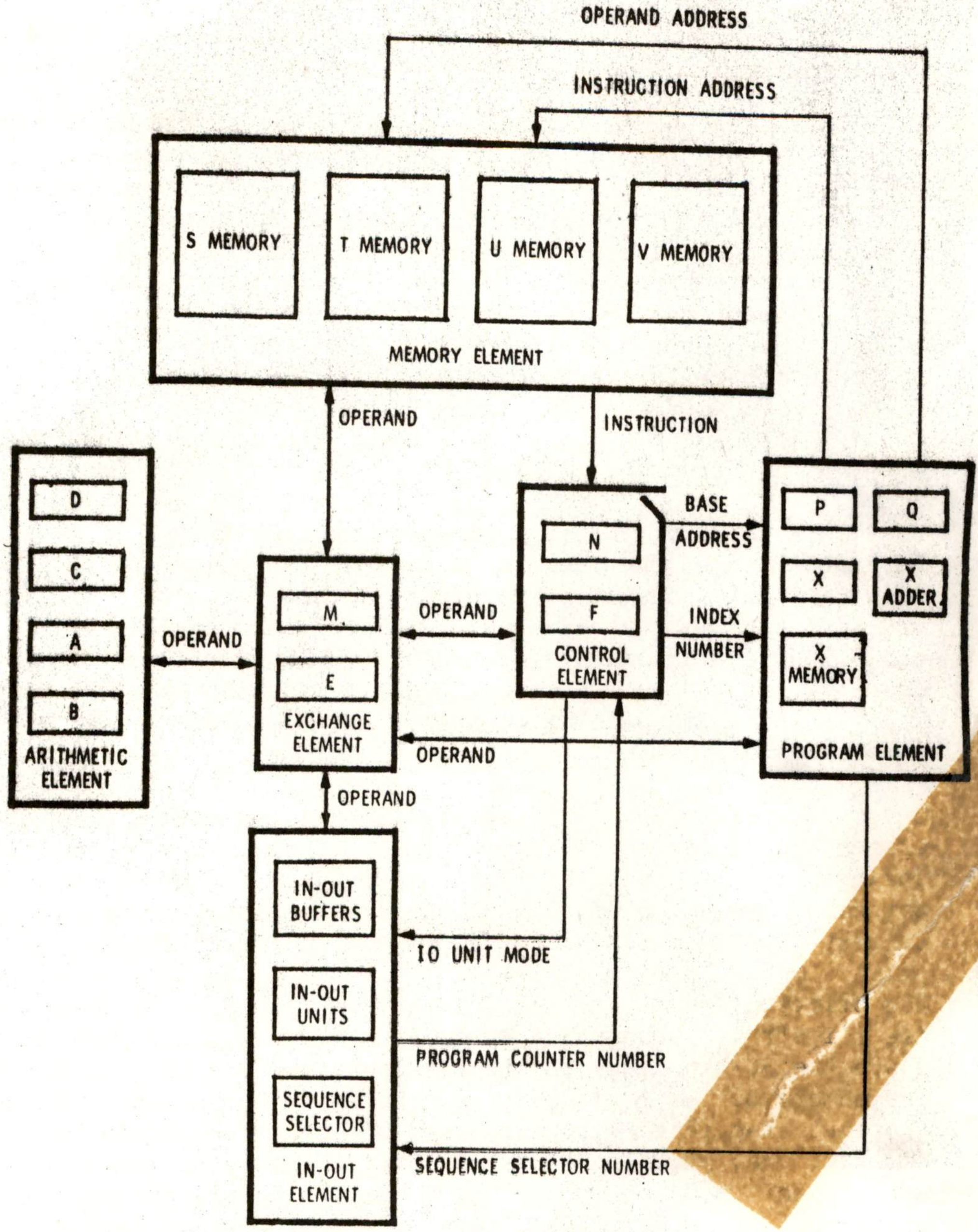


Fig. 3 TX-2 System Schematic

instruction. This flexibility is realized by specifying a particular system configuration with each instruction.

The computer communicates with the outside world through units in the in-out element, several of which can be simultaneously operated. Whenever any piece of data (input or output) is ready to be transferred between an in-out unit and a register in the memory element, signals to the program element from the in-out element automatically call into operation the instruction sequence associated with the in-out unit. This process is referred to as multiple-sequencing and will be described in Section III of this report.

C. Memory Element

The availability of a large, fast, core memory for TX-2 permitted an emphasis on the design of a machine with a completely random-access memory with a design capacity of 262,144 words. The homogeneous aspect of such a large memory system simplifies the programmer's coding problems and permits continued high-speed operation regardless of where the program is located in the internal memory.

The TX-2 memory element (Fig. 4) is divided into four, independently operating memory systems, each containing up to 65,536-digit words. The operating speed of TX-2 is determined by the cycle time for the memories: The 65,536-word S-memory is expected to have a cycle time of between 6.0 and 7.0 µsecs, and the 4,096 T-memory, a cycle time between 5.0 and 6.0 µsecs. Both memories have parity checks.

Although the U-memory is not presently specified, it may contain a 4,096-word core memory in the initial system. The V-memory consists of 8 flip-flop registers in the central machine and 16 toggle-switch registers which contain the program sequence executed whenever the START button on the operator's console is pushed. The contents of the toggle-switch registers can be used as instructions or operands, but cannot be altered by a program. The six 36-bit registers A,B,C,D,E, and F are part of the V-memory, but their contents can be used only as operands during the execution of an instruction. The programmer has, in a limited sense, a two-address instruction machine when he refers to these registers in load and store type instructions. The other two flip-flop registers in the V-memory are a 60-cps clock and a random-number register.

When an instruction calls for the storing of an operand in memory, the operand memory cycle can be extended up to 2.0 µsecs. The extension occurs between the time that the memory register is read and the time that it is rewritten. During this extension, the contents of memory-registers in the central computer are transferred, the parity of the word read from memory is checked, and the parity of the new memory word is computed. Because the extended cycle is less than

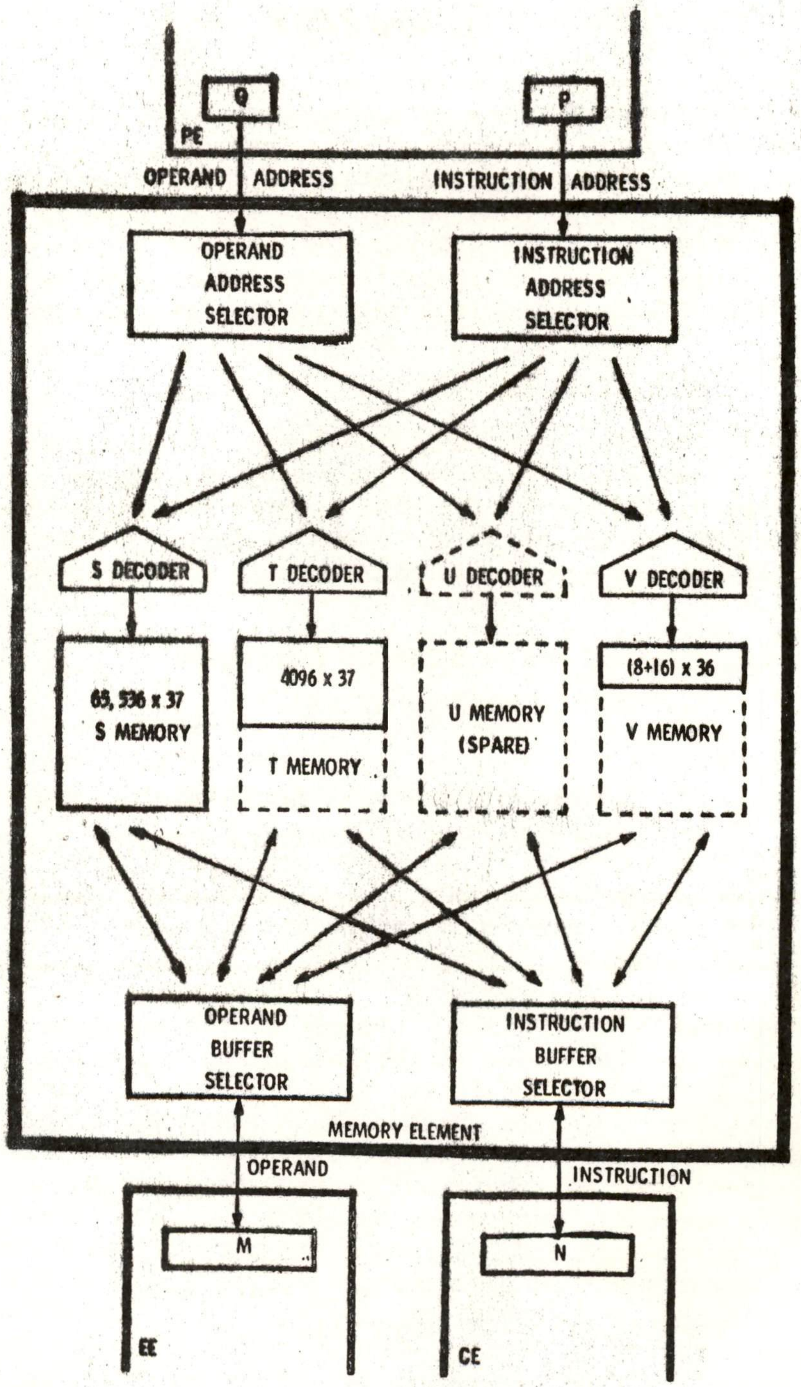


Fig. 4. TX-2 Memory Element - Showing method of using two address registers and two buffer registers to simultaneously operate two out of four memories

the two complete cycles usually used for instructions that modify words, computing efficiency is increased.

The P-register in the program element specifies the location of an instruction in memory and the N-register in the control element holds the instruction after it has been read from memory. The first 2 digits (reading left to right) of P select the memory system from which the instruction word is to be obtained; the remaining 16 digits address the word within the memory. Similarly, the Q-register locates the operand in one of the memory systems, and the operand is placed in the M-register.

D. Control and Indexing

An instruction word read into N has the structure shown in Fig. 5. The first 2 digits of the word specify information to the in-out element, and the 4 cf digits specify the computer configuration. The interpretation of the b and n digits is discussed in Section III. The cf digits are discussed in subsection H, p.17 .

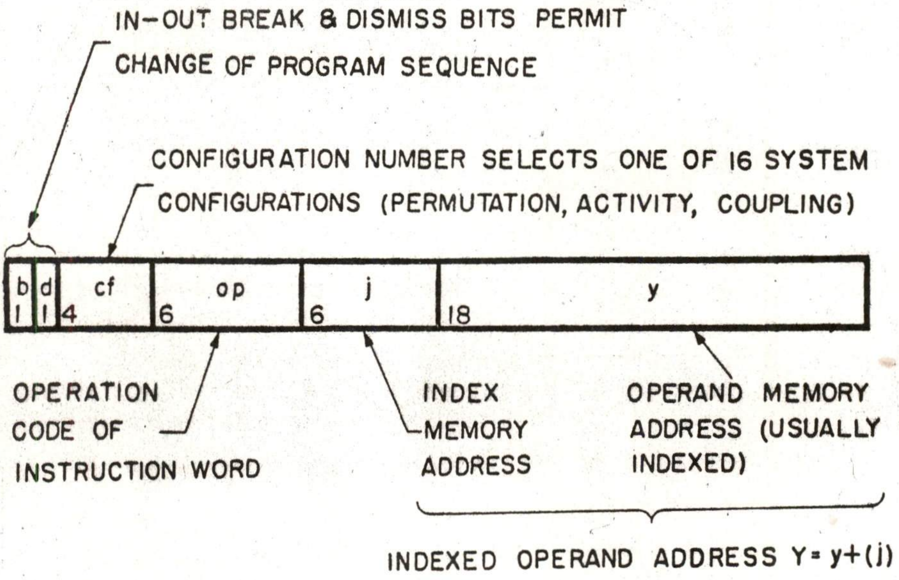

IN-OUT BREAK & DISMISS BITS PERMIT
CHANGE OF PROGRAM SEQUENCE
CONFIGURATION NUMBER SELECTS ONE OF 16 SYSTEM
CONFIGURATIONS (PERMUTATION, ACTIVITY, COUPLING)
b
1
d
1
cf
4
op
6
j
6
y
18
OPERATION
CODE OF
INSTRUCTION WORD
INDEX
MEMORY
ADDRESS
OPERAND MEMORY
ADDRESS (USUALLY
INDEXED)
INDEXED OPERAND ADDRESS Y= y+(j)

Fig. 5. TX-2 Instruction Word Structure

The operation code for the instruction is specified by the 6 op digits. On simple load and store type instructions, these

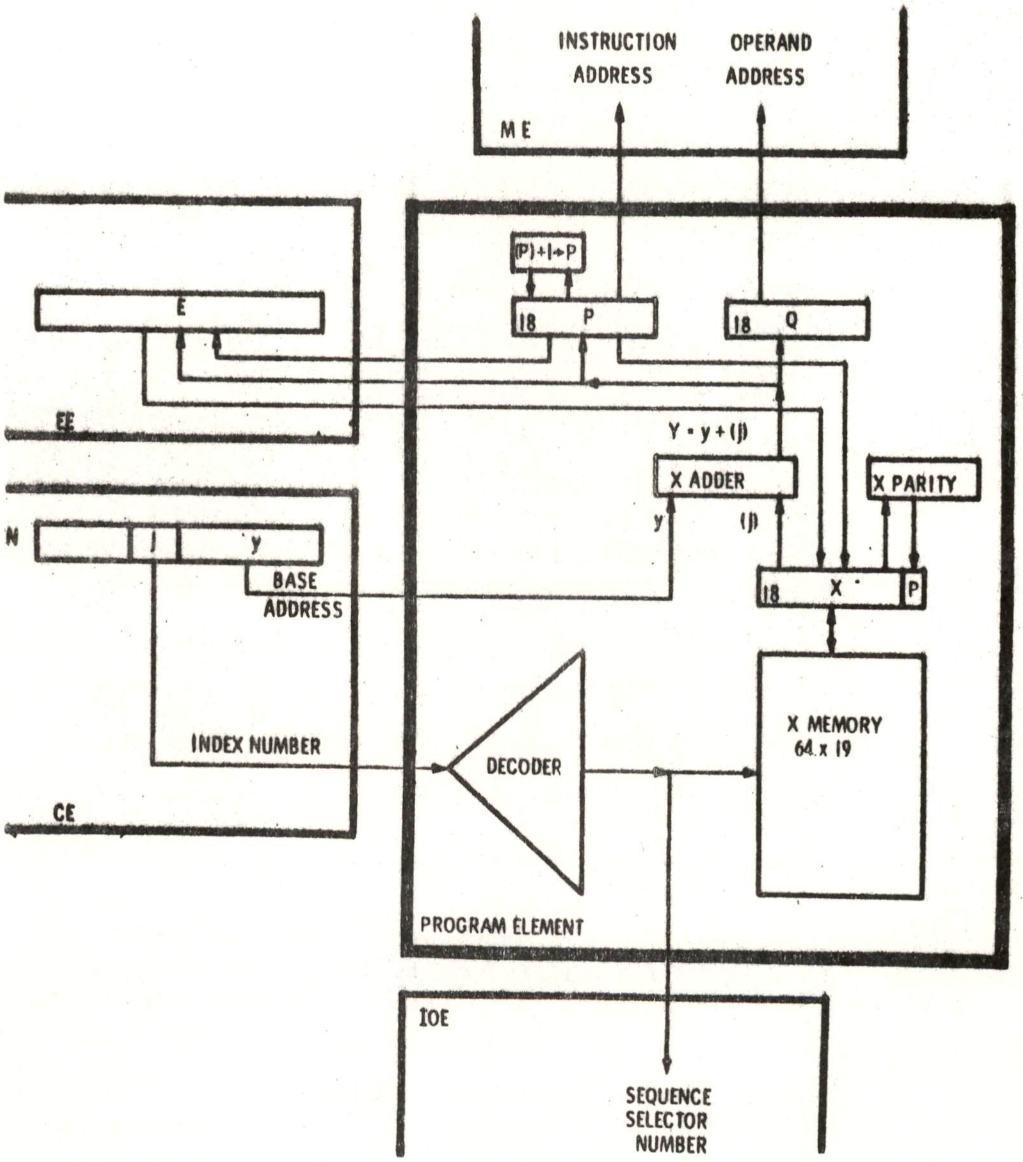


Fig. 6. Program Element Showing paths enabling index adding and storing and loading program counter from the index memory and the exchange element

6 digits are further subdivided into two groups of three. The first group determines the operation and the second specifies the register in the central computer that is being loaded, or whose contents are being stored.

The base address for the operand, formed by the 18 y-digits, is usually modified by the contents of the index register selected by the 6 j-digits. The index-registers form a unique, 64-register, parity-checked, core memory which has an access time of 1.0 μsec. The contents of the specified index register is read into the X-register of the program element via the paths indicated in Fig. 6. The base address and the index are fed into a full adder circuit which produces the sum, X = y + (j), in about 1.0 μsec. The over-all complexity of the program element was reduced by having the adder produce both the sum, Y, and the unmodified base address, y: either of these quantities can be directed to the operand memory address register, Q. Whenever the FIRST (j = 0) index register is chosen, the adder produces only the unmodified base address. The effect is the same as having that index register contain ZERO and the programmer can avoid index modification altogether.

The P-register which holds the memory address of instruction words, normally is indexed by one as each instruction is executed, but the output of the index adder is sometimes directed to P when jump instructions are executed. The adder also provides a communication path for index jump instructions from the X-memory to the memory element by way of the exchange element.

E. Arithmetic Element (AE)

Fig. 7 shows the registers and sufficient basic operations in the arithmetic element for adding, multiplying, dividing, shifting, and executing various logic instructions. Operation timing for most of the TX-2 instructions is also performed in the arithmetic element.

The design of the AE reflects the desire to attain high-speed operation for TX-2, even when the AE is carrying out lengthy instructions. The only instructions which take longer than one memory cycle for execution are those which involve shifting. These are, for example, multiplication, division, arithmetic and cyclic shifting, and normalization. Therefore, the AE contains a sufficient number of storage registers to permit these instructions to be carried out in the AE while the remainder of TX-2 is free to execute other instructions.

The four registers in the AE can each communicate with the E-register in the exchange element and thus with the Memory Element. As mentioned earlier, these registers are addressable

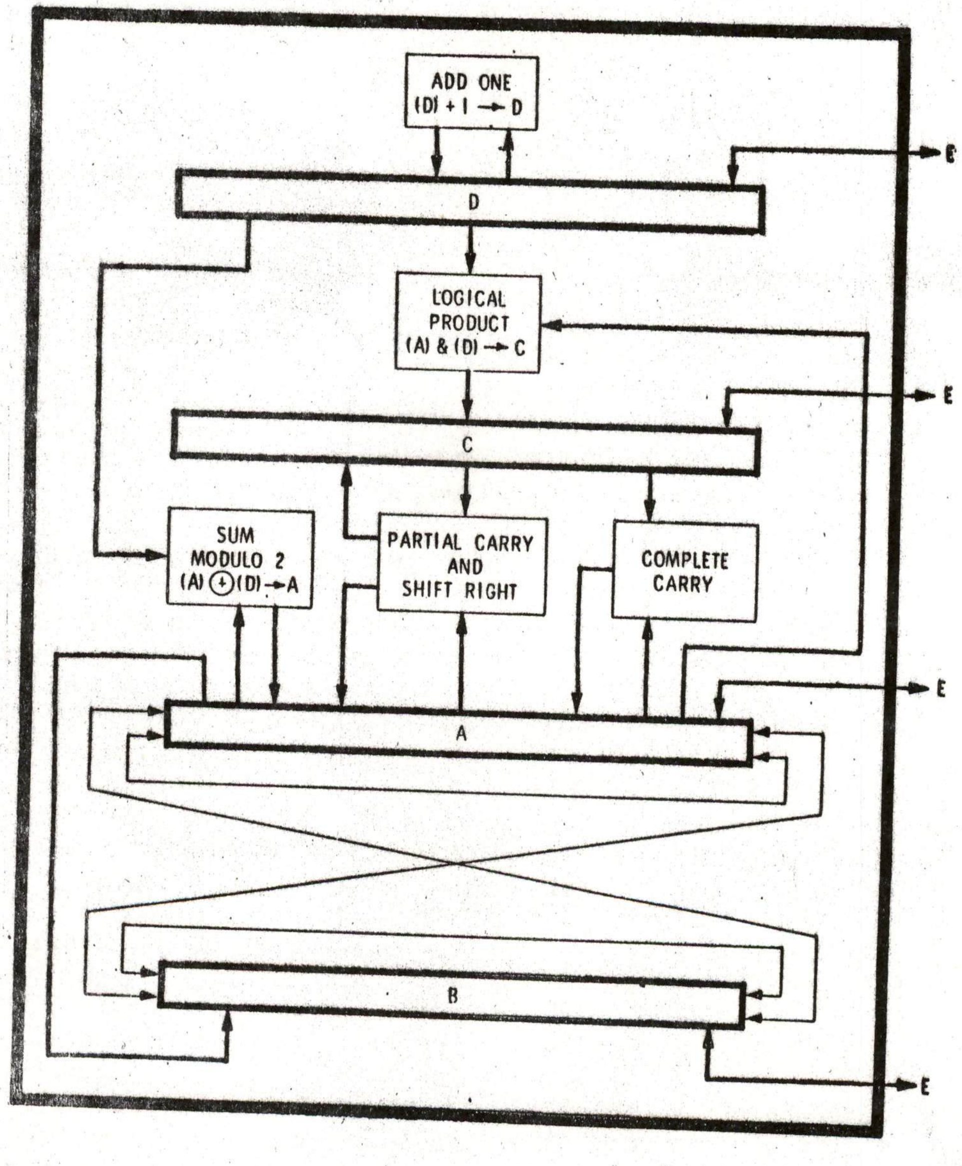


Fig. 7. Circuits and Transfer Paths (General) Of Any TX-2 Arithmetic Element Forms

as part of the V-memory system. Therefore, programmers have access to the results in any register of an AE computation.

The AE registers, designated by A,B,C, and D, are described below:

The A-Register accumulates the results of all the arithmetic operations, except division, for which it holds the remainder. It holds one of the operands and accumulates the results of the three logic operations (AND, INCLUSIVE OR, EXCLUSIVE OR) which, it should be noted, are bit-wise operations. The information in the A-register can also be shifted (i.e. multiplied by some positive or negative power of two) or cycled (i.e. shifted, without preserving the special significance of the sign bit, as in a closed ring).

The B-Register serves as an extension of A during multiplication, certain shifts and cycles, and, in a sense, during division when the least significant digits of the double-length dividend are stored in B. The resulting quotient then appears in B. Moreover, the information in B can be shifted or cycled independently of A. In multiplication, the multiplier originally in A is transferred via parallel paths directly into B (where the least significant digit then controls the operation).

The C-Register stores the partial carries during arithmetic operations, which is most important during multiplication, as described later. Since these partial carries are actually bit-wise logic products AND, C is also used to accumulate logic products.

The D-Register holds the multiplicands, divisors, addends and one of the operands for the logic operations. It also holds the numbers which control the shifting and cycling of A and B, namely the number of places, up to 62, and the direction, right or left. The ability of D to count is used also to accumulate the results of normalizing A and counting ONES in A.

Each of the AE registers can also be complemented, thus allowing subtractions to be performed.

F. AE Circuits

There are four add-one circuits on D, so that different parts of A and B can be controlled separately and simultaneously. For simplicity, just one add-one circuit is shown in Fig. 7. These add-one circuits use the simultaneous carry principle, permitting one count every 0.4 μsec; each can count up to 127.

The logical-product (AND) circuit of A and D into C and the sum-modulo

2 (EXCLUSIVE OR) circuit of A and D into A when used at the same time are called a partial add. When the complete-carry circuit is activated after a partial add, the result is a full addition of D and A into A. The complete-carry circuit uses the high-speed carry principle and takes about 1.5 μsecs for 36 bits.

The partial-carry and shift-right circuit is also known as "multiply step" and was, we believe, first used on Whirlwind I. As it is used in multiplication, this circuit makes a full addition unnecessary for each ONE in the multiplier. Carries are extended only one stage during each step, except the last, when a complete carry is executed. The complete prcess takes about 16 μsecs at most for a full, 36-digit multiplication. The process for division, on the other hand, requires a complete addition at each step and consequently takes about 72 μsecs in the worst case.

Two features of the AE control should be mentioned here. A 7-bit step counter, like the add-one circuit on D, controls multiplication and division and limits the shifting in normalizing and the cycling in counting ONE's. A flip-flop, which signifies overflow during addition and division, is also used to remember the sign of the product during multiplication and the sign of the quotient during division. If division causes an overflow, the sign is replaced by the overflow state and the quotient is lost.

Control of the arithmetic element is independent of the rest of the machine. Thus, non-AE instructions can be executed while the AE continues to perform a long shift operation or a division.

G. System Timing

The high speed of TX-2 is attained in part by overlapping the operation of as many components as is logically possible without incorporating large amounts of circuitry. The time-consuming cyclic operations in an indexed, single-address computer are the instruction memory cycle, the index memory cycle, the index addition time, the operand memory cycle, and the operation timing. These cycles occur in the mentioned sequence during the execution of ordinary instructions. Several asynchronous "clocks," which use a 5-megacycle pulse-source, control the cycles. The instruction and operand memory cycles can be overlapped if they take place in different memory systems.

The overlap of these cycle times for a sequence of load-type instructions is illustrated in Fig. 8 (a). Here different instruction and operand memory systems are assumed to have roughly equal cycle times. If a sequence of store-type instructions is executed which requires extended memory cycles for the operands, then the situation is as shown in Fig. 8 (b). Fig. 8 (c) shows the time used when both the instruction and the operand are in the same memory.

"Peak" operating speed for the computer is attained only when the situation is as shown in Fig. 8 (a); in the situation shown in Figs. 8 (b) and 8 (c) the operating speed could be increased by adding circuitry, but only at considerable cost. It is interesting to note that if the computer is to run at peak speeds, the address of the operand used by the present instruction must be available before the earliest moment at which the next instruction memory cycle could begin. If the total accumulated time, from the beginning of an instruction memory cycle till the time that the address of the operand is known, is greater than the instruction memory cycle time, then the computer cannot run in the ideal manner shown in Fig. 8 (a). This means that the access time of memories and the index-add time must be kept as short as possible.

Fig. 8 (d) shows the timing of events when the in-out element causes a change in program sequence by changing the contents of the P-register. The additional X-memory cycle, which must be performed, produces a timing situation similar to that of the X-memory load and store instructions.

The operation timing for an instruction is executed when the operand is available from memory. Only the arithmetic element step-counter instructions (multiply, divide, shift, etc.) require an operating timing cycle longer than a memory cycle. Since only the arithmetic element is tied up by these instructions, the control element permits any non-arithmetic element instruction to be executed while the AE is busy. Division takes up to 75 μsecs, so the programmer can write as many as 14 non-AE instructions following a divide, all of which can be executed before the division is completed.

H. Configuration Control

The design of a general-purpose computer must necessarily reflect the contradictory demands for both short and long words, floating and fixed-point arithmetic operations, and a multitude of logic and decision instructions. The computer should be able to process information at an optimum rate, in a variety of problems, without the need for intricately coded programs. This ability should be achieved without excessively complex and costly circuitry.

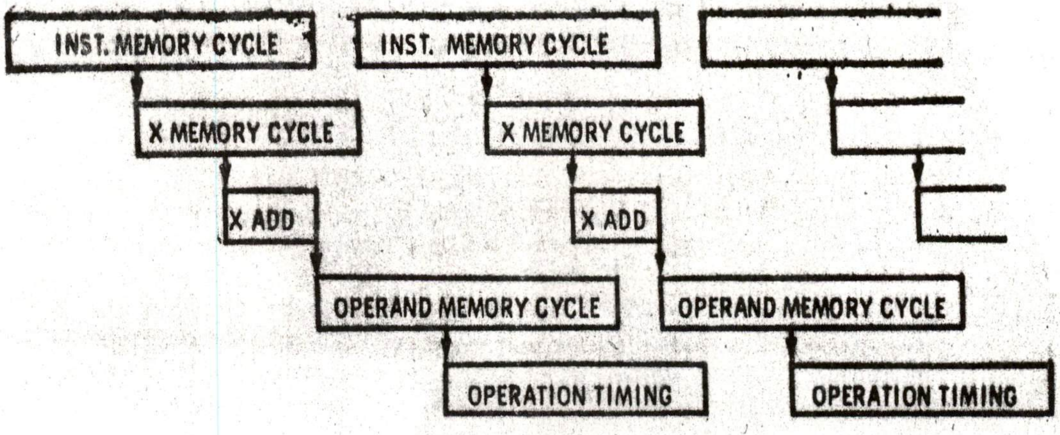


Fig. 8 (a) Consecutive Load Type Instructions
Instructions and Operands in Different Memories

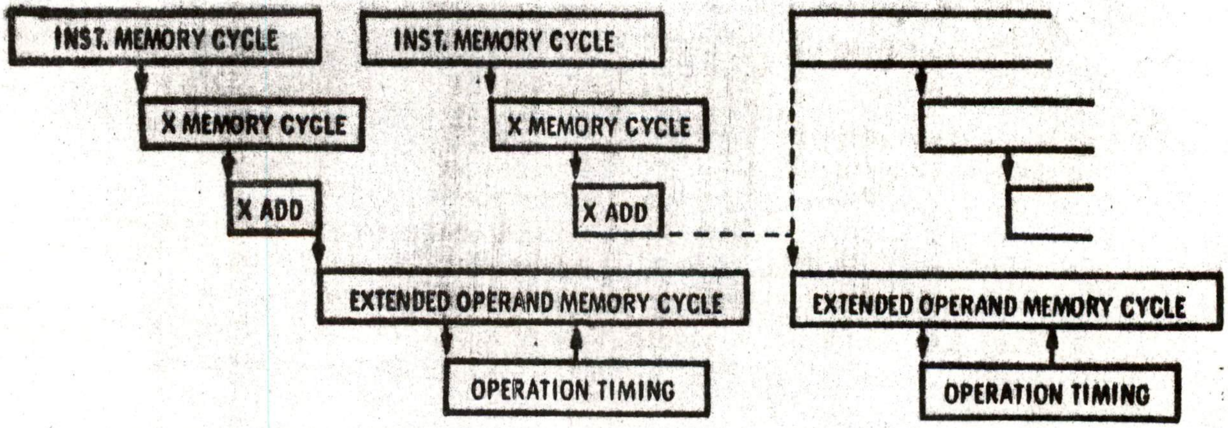


Fig. 8 (b) Consecutive Store Type Instructions

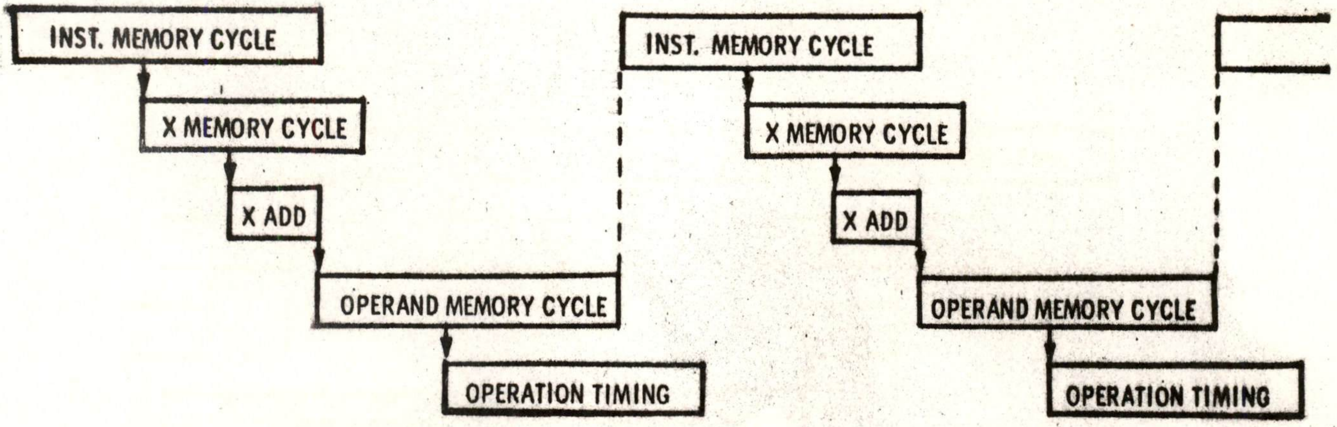


Fig. 8 (c) Instruction and Operand In Same Memory

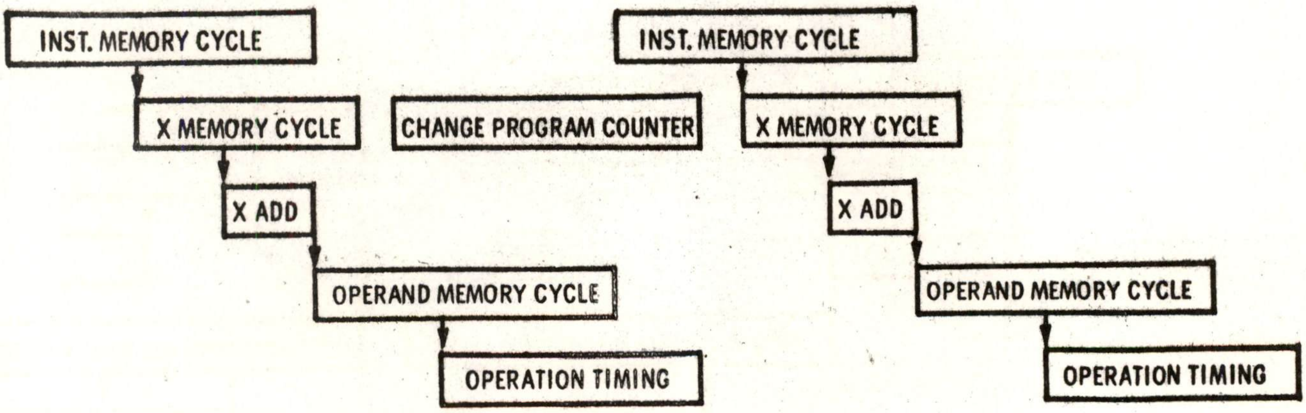


Fig. 8 (d) Change Sequence

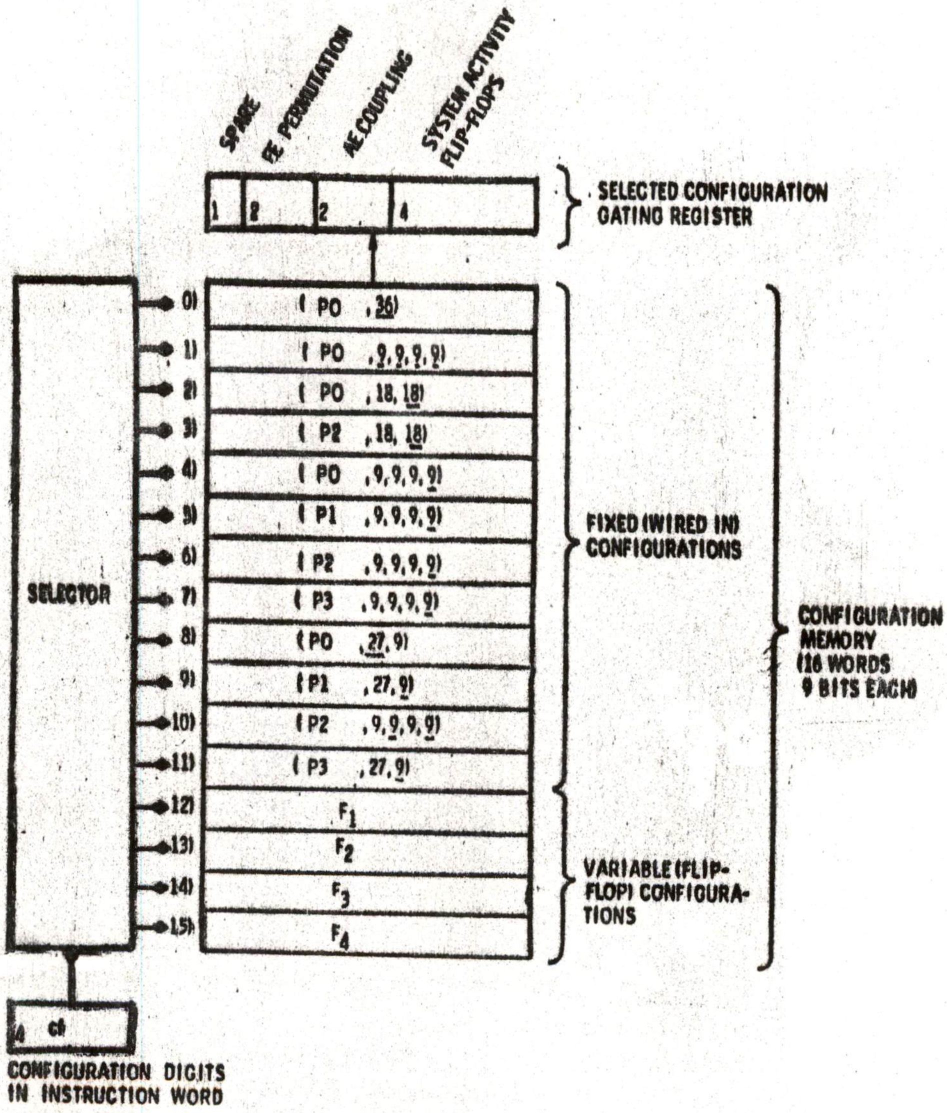


Fig. 9. TX-2 Configuration Selection

The 36-digit word of TX-2 represents a reasonable length for operands in some numerical computations, notably scientific and engineering computations. Though floating-point arithmetic operations are not included in the instruction code, both they and multiple-precision operations can be easily synthesized by means of the existing instructions. The logic instructions in the code facilitate operations on individual digits. Also, a configuration which the programmer specifies with each instruction permits him to perform arithmetic operations on operands that are less than 36 digits long. When operands are less than 36 digits, several can be manipulated simultaneously.

The four cf digits in an instruction word (see Fig. 5) are decoded as shown schematically in Fig. 9, selecting one of 16 configurations. The selected configuration, represented by a 9-digit word, is placed in a flip-flop register whose output levels determine a static configuration for the entire computer while the instruction is executed. The notation used to represent the contents of the first 12 registers will be clarified in the following discussion.

The full 36-digit word length is always maintained for instruction words, but during operation timing, every 36-digit register in the memory, exchange and arithmetic elements is considered as broken into 9-digit quarters, numbered from 1 to 4, right to left as in Fig. 10 (a). While the instruction is being executed, these quarters are recombined on the basis of the configuration.

Information is usually moved about in the machine by parallel transfers between registers (see Fig. 9). The EE permutation digits select one of the four permutations P0, P1, P2, P3, shown in Fig. 10 (b). The chosen permutation effects the corresponding cross-communication paths between the quarters of the E- and M-registers of the exchange element. While operands are transmitted through the EE, the quarters follow the set of paths determined by the selected permutation. The result is that the operand is shifted 9n places to the left as it moves from M to E or 9n places to the right as it moves from E to M, n = 0,1, 2 or 3. Thus, the programmer can have any quarter of the AE communicate with any quarter of the ME.

This ability to communicate is focused more sharply by having the configuration specify a system activity (see Fig. 9). All operation timing events, in a given quarter of the AE and EE and the quarter of the ME connected via the selected permutation path in the EE, are controlled by the activity flip-flop on that quarter. If the activity flip-flop of a given quarter is a ONE, as specified by the configuration, the operation timing

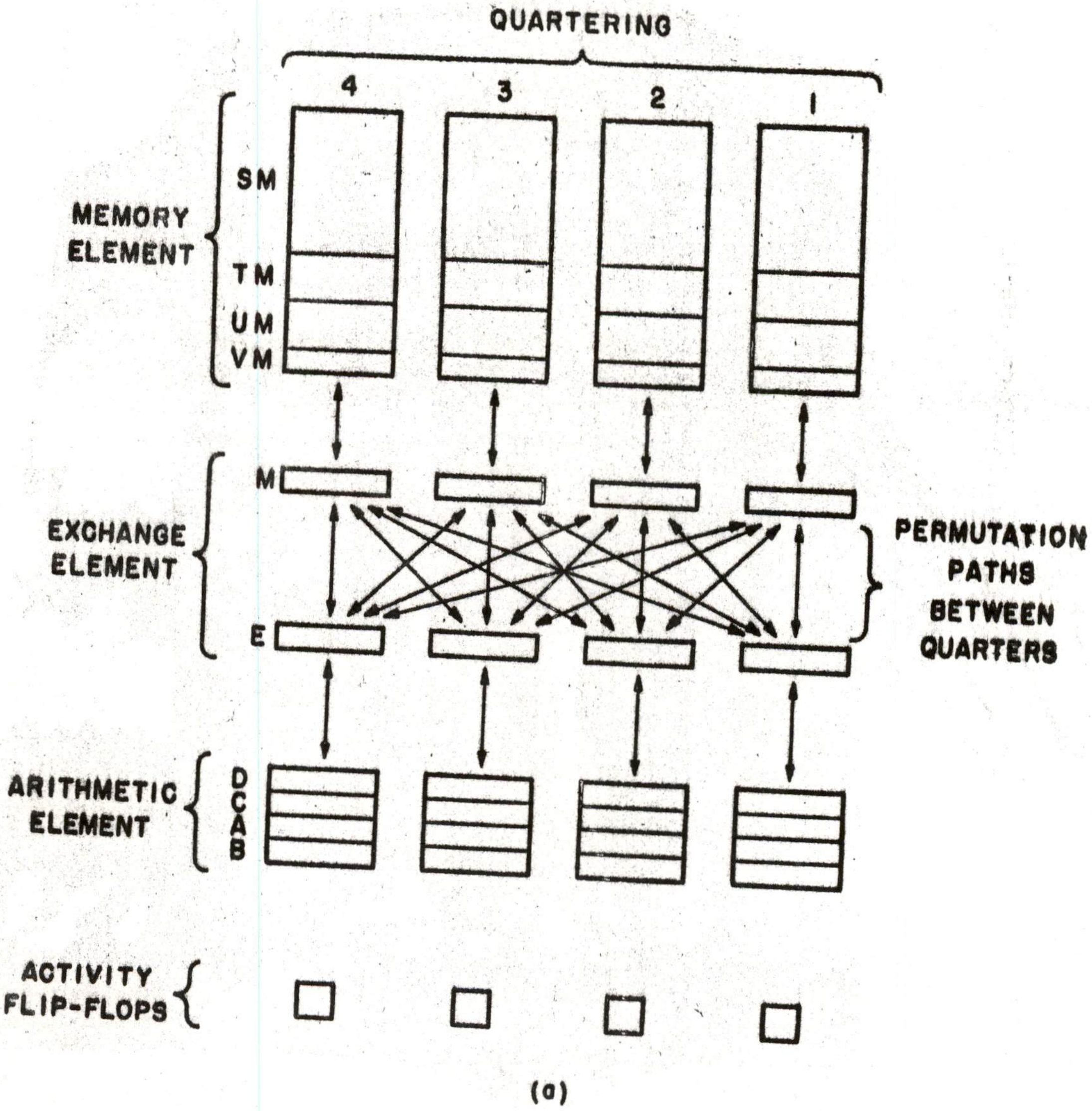


Fig. 10 (a) Quartering Permutation Paths and Activity Flip-Flops Shown

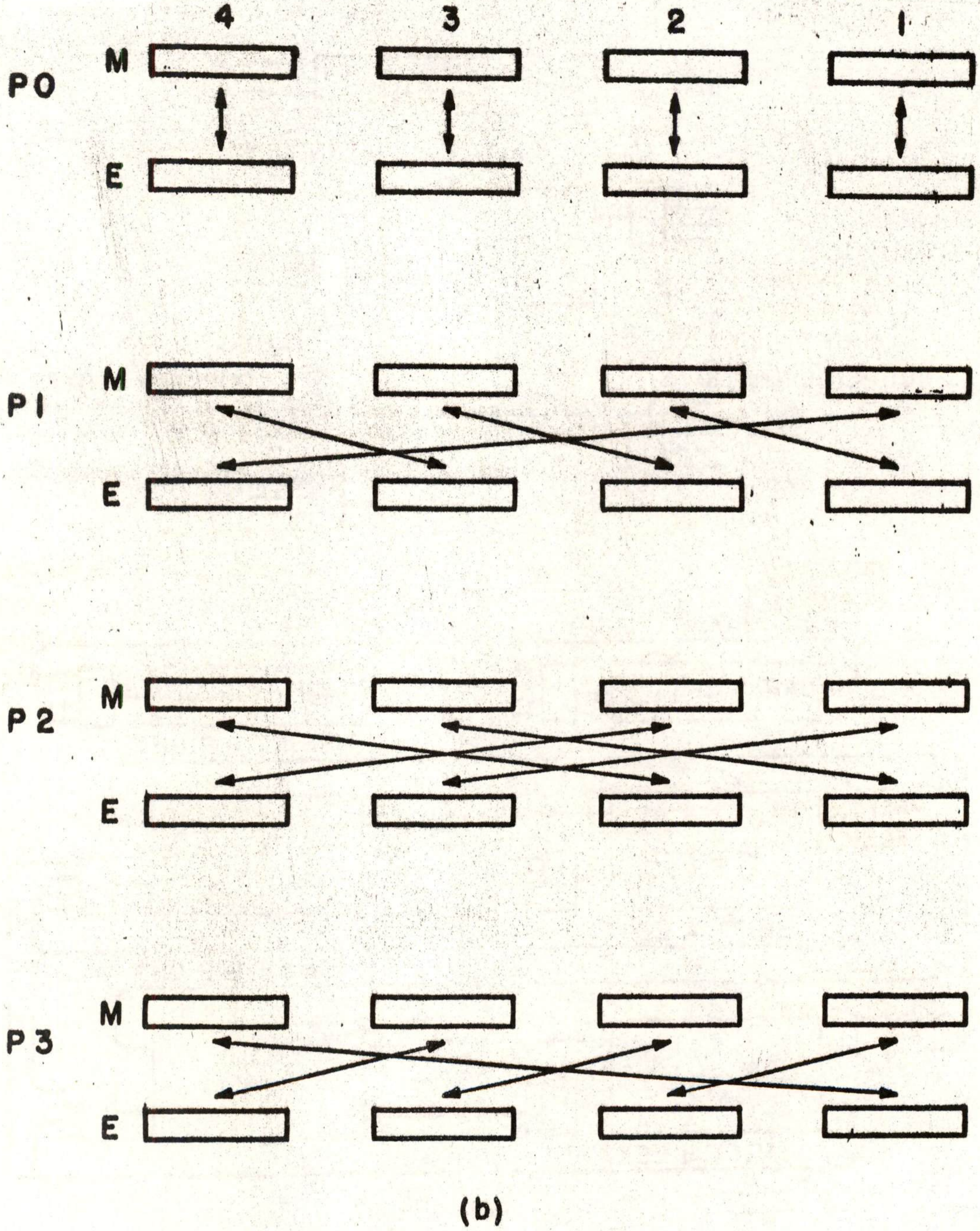
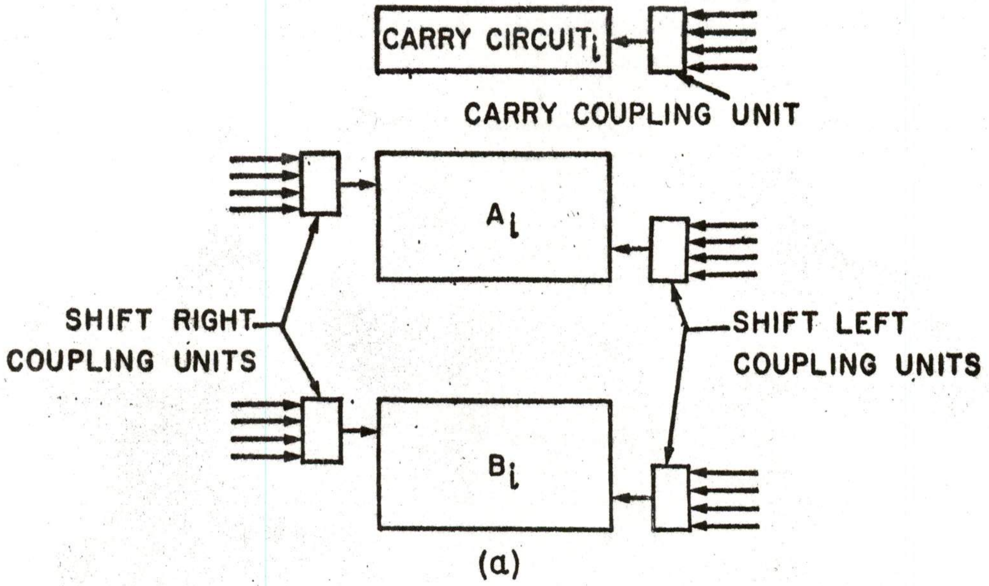


Fig. 10 (b) Paths in Exchange Element



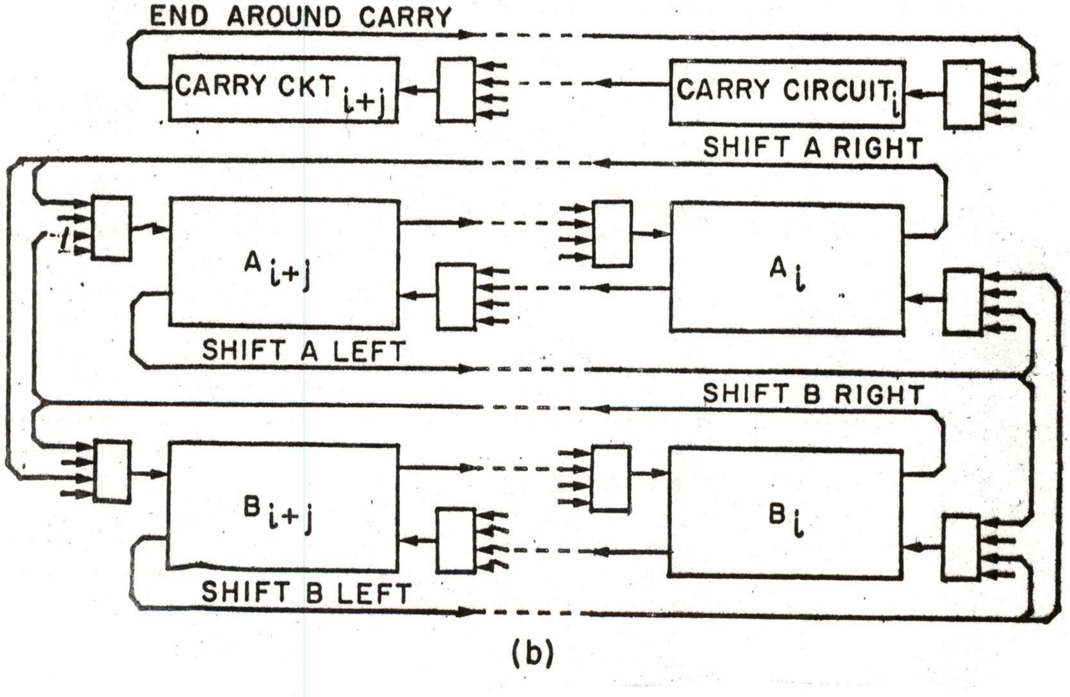


Fig. 11 (a) ith Quarter Coupling Units
(b) Coupling Unit Connections

events of the instruction occur in that quarter. If the activity flip-flop is a ZERO, nothing happens.

During the execution of arithmetic operations, the AE coupling digits (see Fig. 9) further specify the connections of the lateral information paths between quarters in the AE. Information flows laterally only through the shift and the carry circuits, and the connection of these circuits alone determines the word lengths of the numerical quantities manipulated in the AE.

As shown in Fig. 11 (a), every quarter of the AE has coupling units at each end which receive the shift and carry information entering the quarter. The general type of connections between several quarters is shown in Fig. 11 (b). The digit length of operands, during add and shift operations, is determined by the number of quarters coupled together. In TX-2, from 1 to 4 quarters can be coupled together to permit arithmetic operations on 9-, 18-, 27-, or 36-digit operands. The various combinations of coupling-unit connections actually chosen by the AE coupling are symbolized in Fig. 11 (c). Since A-register, B-register, and AB-register shifts are permitted in the arithmetic element, the programmer can obtain 18-, 36-, 54-, or 72-digit shifts. All the possible shift (and cycle) configurations are shown in Fig. 11 (d).

Only those inputs to the coupling units which would yield useful arithmetic element structures are realized by the AE coupling. It should be emphasized that the programmer can realize several arithmetic elements simultaneously. The coupling, (36), gives only one 36-bit arithmetic element, but the coupling, (18,18), gives two complete, independent, 18-bit arithmetic elements which are separately, but simultaneously, controlled by the instruction being executed. Two arithmetic elements are again available with the coupling, (27,9), one 27 bits and the other 9 bits long, and the (9,9,9,9) coupling gives four, 9-bit arithmetic elements. The permutation paths in the exchange element permit each arithmetic element to communicate with any quarter of a memory word and the activity flip-flops can specify just which of the realized arithmetic elements actually will be active and will communicate with the connected part of memory.

In Fig. 12, several examples are given of the different configurations which can be realized in TX-2. The most straightforward has one, 36-digit arithmetic element and communicates directly with memory. The notation, (PO, $\underline{36}$) signifies the permutation (no shift) and the form of the arithmetic element (one 36-digit). The underlining indicates that the whole system is active. Slightly more varied is the (PO, $9,\underline{9},9,\underline{9}$) configuration which specifies four, 9-digit arithmetic elements communicating directly with memory, but with only two of them active. The (P2, $9,\underline{9},9,\underline{9}$) configuration

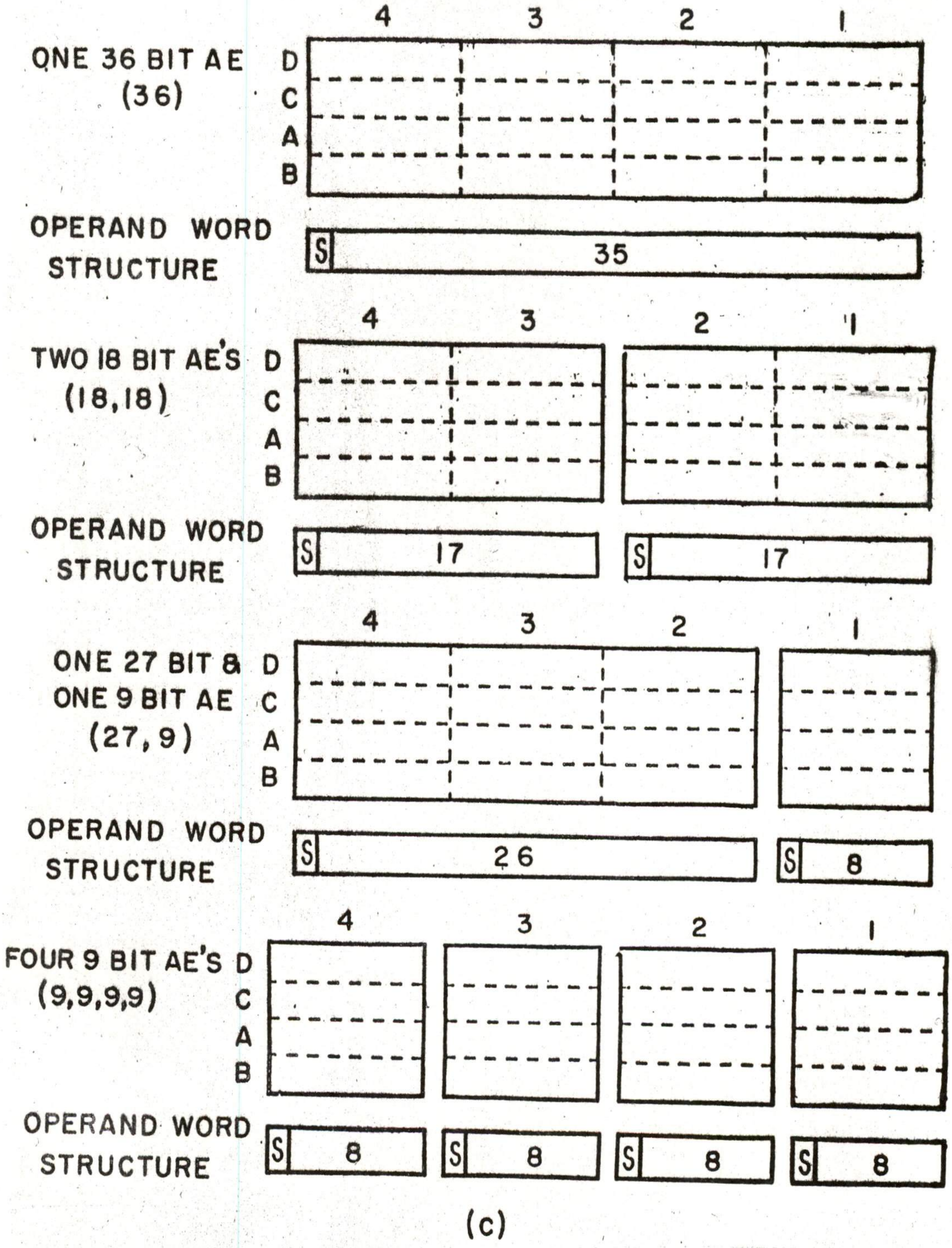


Fig. 11 (c) TX-2 Configuration - Arithmetic Elements And Operand Word Structures

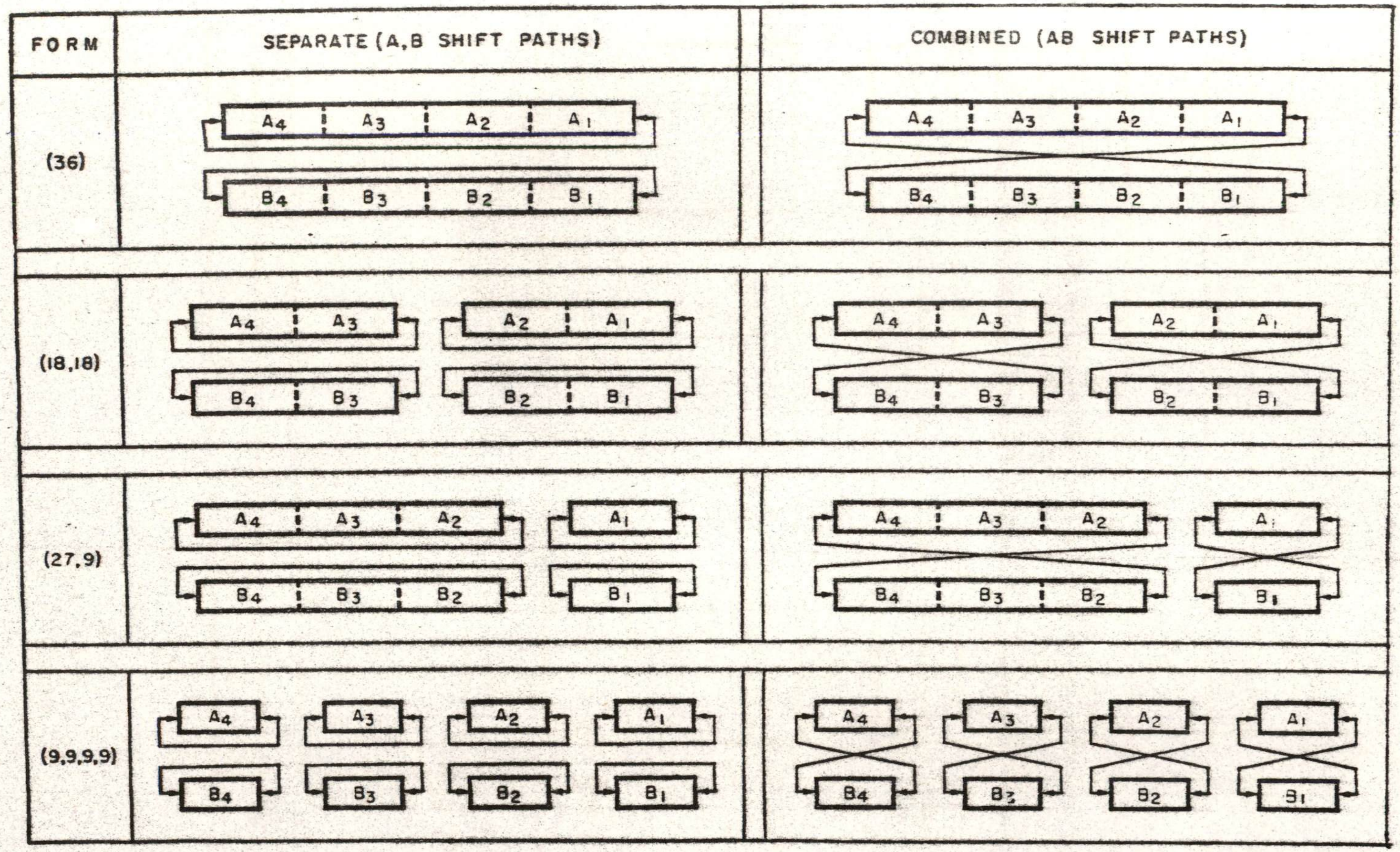
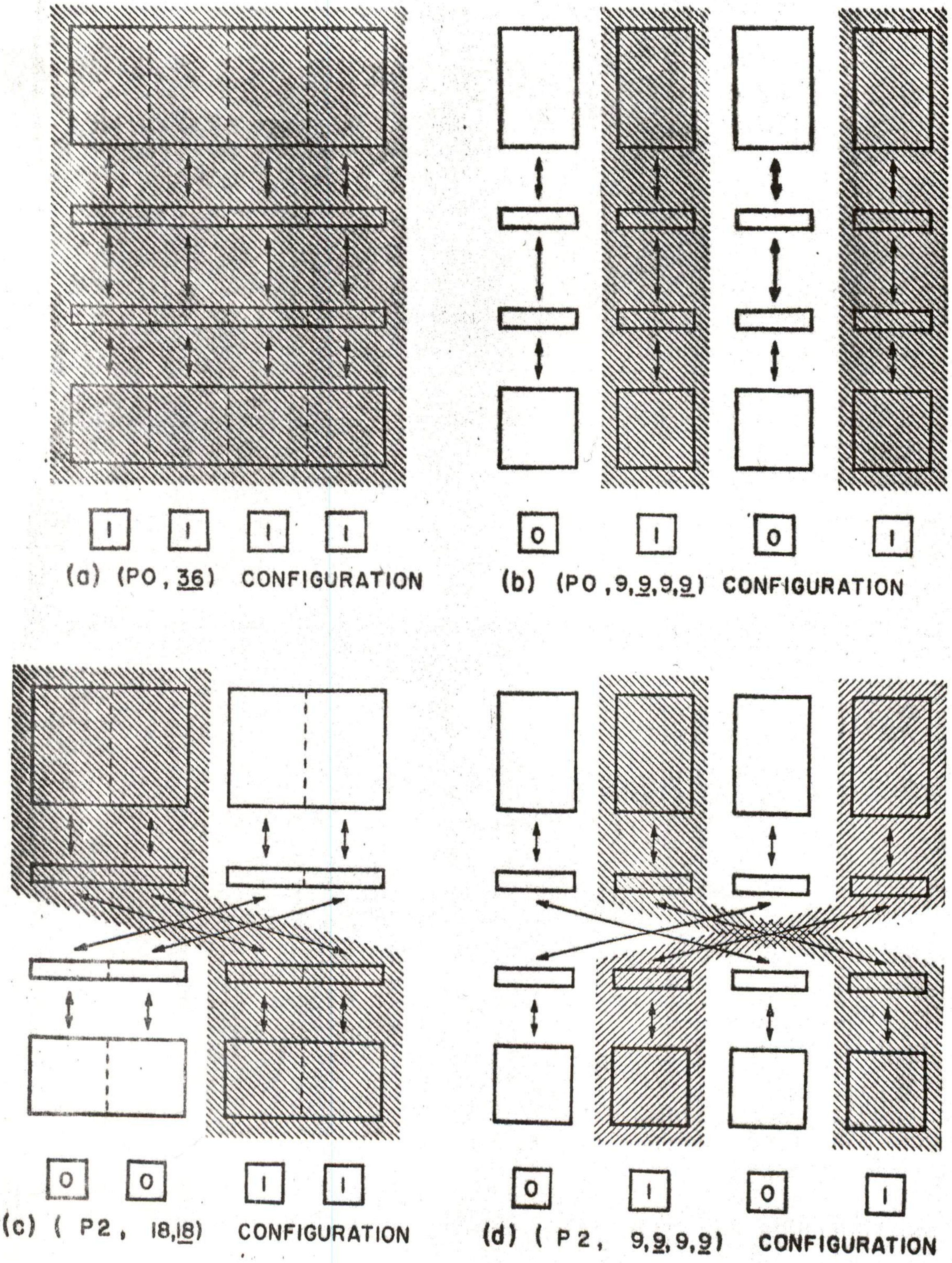


Fig. 11 (d) TX-2 Shift Path Arrangements



(a) (PO, $\underline{36}$) CONFIGURATION

(b) (PO, 9,$\underline{9}$,9,$\underline{9}$) CONFIGURATION

(c) (P2, 18,$\underline{18}$) CONFIGURATION

(d) (P2, 9,$\underline{9}$,9,$\underline{9}$) CONFIGURATION

Fig. 12. TX-2 Configurations
Areas Of Activity During Execution Of Instruction Shown Shaded. Effect Of AE Couplings Illustrated By Juxtaposition Of Quarters

has the same arithmetic elements but with the associated memories interchanged. The (P2, 18,18) configuration illustrates an 18-digit arithmetic element which uses the "other" half of memory.

One of the digits in the 9-digit configuration is at the moment unused, but will probably be used to control the extension of the sign of numbers as they pass through the EE on the way from the ME to the AE. The scheme presently under consideration would permit programmers, for example, to add a 9-digit memory operand to an 18-digit arithmetic element. This scheme would permit closer packing of operands in memory and significantly increase the speed of solving some real-time problems, where the sign of short pieces of data must be extended so that higher precision can be maintained during computations. The working details of the scheme have yet to be fixed.

The memory from which the programmer chooses a configuration for use with each instruction was shown in Fig. 9. Twelve of the memory registers are fixed circuitry whose contents cannot be changed without changing the wiring of the computer. The configurations represented by the contents of these registers are assumed to be useful to most programmers. The last four registers in the memory consist of the 36 digits of the F-register. As will be seen, the programmer can quite simply alter the contents of this register and thereby obtain any of the possible configurations. The total number of distinct possible configurations is less than 2^9.

I. Operation Code

Only 51 of the 64 possible operation codes are currently decoded to define instructions. In Table I the effect of each instruction is described. If the selected configuration defines several computers, the operation takes place in all of them simultaneously and independently. The notation used in the definition of the operation is described in Table II.

The instructions are grouped according to type. Load and store instructions transfer an operand between a selected register and memory. The load-complement instructions are variants which load the ONES complement into the selected registers. The Exchange instruction interchanges the contents of A and of the addressed memory register. The insert instruction allows any set of bits in A, as specified by the bits in B, to be stored in memory. The j-bits of the load and store instructions, which refer to the index memory, select the index register involved so that the operand address is not modified.

All of the add and step-counter instructions can also be classed as load type instructions, in so far as the operand memory cycle is concerned. The multiply instruction forms the full product in the A and B registers. Division is the inverse of multiplication, the double-length divident in A and B being divided by the memory operand. The remainder is left in A and the quotient in B. Normalize instructions shift the contents of A and B left until the magnitude of the number in A is between one-half and one. The number of shifts to do this (the normalizing factor) is subtracted from the memory operand in D. Since more than 18 bits are required to specify all the possible shifts for the (9,9,9,9) configuration, the shift and cycle instructions use a memory operand, rather than the address section of the instruction to specify the number of places to shift. The count ones instruction adds the number of ONE's bits in A to the memory operand in D, thus providing a simple means for determining bit density in areas of storage, since the ONE's count for several words can be accumulated in D.

The two replace-add instructions, using the index memory, facilitate instruction and index modification. Both require two memory-cycle times for execution.

The two in-out read instructions transmit information between the memory and the selected in-out unit. The details of these instructions and the in-out select instruction are given in Section III.

Single bits in memory can be manipulated with the three bit-setting instructions. The bit-sensing instruction facilitates the use of single bits in memory as operands. The variety of available jump instructions simplifies the coding of, logic decision functions. The two index-jump instructions permit indexed program loops to refer successively in either the forward or backward direction to operands in a data block. The unconditional-jump instruction uses the cf digits to specify whether the selected index register will be used to remember the previous contents of P. These contents are always transmitted to the E-register whenever a jump occurs.

Arithmetic overflows can be caused by addition, subtraction, and division instructions. Such overflows as do occur are remembered in overflow flip-flops in the arithmetic element. The overflow condition can be detected by a jump instruction, or by the in-out element in a manner described in Section III. If an overflow is anticipated, however, it can be shifted into the A register by executing a normalize instruction. A normalize usually shifts AB left. However, if an overflow exists, AB is shifted right one place and the overflow placed in the most significant digit position of A to the right of the sign digit. The memory operand is also increased by one in the D register, rather than decreased. This interpretation of an overflow permits floating-point operations to be programmed very simply in the arithmetic element. The in-out select and operate instructions differ from all the others in the sense that the y-digits are used to specify different operations. In-out select chooses the mode in which an in-out unit will run. The operate instruction will control individual useful commands, e.g., round-off.

J. Instruction Times

The average execution time for instructions depends upon whether one memory or two different, overlapped memory systems are used for instructions and operands. If one memory is used, the execution time is the sum of the instruction and operand cycle times; if two memories are used, the execution time is the longer of the two memory cycle time. It should be remembered that any instruction which involves storing an operand in memory has the normal operand memory cycle time extended by from 1.0 to 2.0 μsecs. Instructions which alter or transfer the contents of index memory registers require approximately two memory cycles, even when instruction and operand memory cycles are overlapped.

Successive step-counter instructions require a time which depends upon the length of the longest active arithmetic element. In the case of multiply, divide, and count ONE's, this time is a function of the length of the operand word only, but the shift, cycle, and normalize times depend upon the number of places actually shifted. Divide requires about 2.0 μsecs per digit and all other step-counter instructions 0.4 μsecs per digit. These shift times become significant only when they exceed the one or two memory cycles already required. In the worst 36-digit case, about 75 μsecs are required for division and 19 μsecs for multiplication. A 72-place shift would take 32 μsecs. These are the times required for these instructions when they are written in sequence. If the operand word is shorter, then these times become proportionally less, down to the minimum memory times required.

K. Summary

The organization of TX-2 permits a programmer, who pays considerable attention to coding details, to receive a worthwhile reward in the form of increased efficiency of operation. The operating speed can be doubled when instruction and operands are stored in different memories. Higher speeds result from sequencing instructions so that non-AE instructions are executed concurrently with AE step-counter instructions. And the ability to choose a configuration with each instruction means not only that some instructions take less time, but also that many of them can be eliminated from a program altogether.

However, this versatility and efficiency is not accompanied by a disastrous loss in simplicity. The system organization is such that details can be easily ignored by the naive programmer, without the details having even subtly unfavorable effects. If all the digits in an instruction word are ZERO, except for the operation code and the base address, then TX-2 appears as a simple single-address computer with operand words 36-bits long and with a single, uniformly addressed, 70,000-word memory.

If the j-bits are used, then the machine is enlarged to become an indexed, single-address, computer, with operand words 36 bits long, for which the entire instruction code is meaningful. When the b and d bits are used, the programmer can control the manner in which several in-out units, running concurrently, can cause the computer to change program sequences. And by selecting various configurations, the programmer can perform more operations simultaneously with each instruction.

The different facilities for indexing, memory overlap, instruction overlap, multiple-sequencing, and configuration can be ignored or used as the programmer desires. Ignoring them would seem to permit straight-forward coding; using them actually permits much shorter and faster codes for a given function. Each facility is represented by a clear concept of what the facility permits, the only real difficulty being the number of simultaneous actions possible with each instruction. However, higher speeds and greater system capacity are obtained by shorter cycle times, increased bit storage and greater simultaneousness of events. In TX-2, all three aspects are emphasized.

TABLE I

Type	Mnemonic Code	Operation	Name
Load	lda	$(Y) \rightarrow \left\{\begin{matrix}A\\B\\C\\D\\E\\E\\F\end{matrix}\right.$	Load into A
	ldb		Load into B
	ldc		Load into C
	ldd		Load into D
	lde		Load into E
	lde		Load into E
	lde		Load into F
	lca	$\overline{(Y)} \rightarrow \left\{\begin{matrix}A\\B\end{matrix}\right.$	Load complement into A
	lcb		Load complement into B
	ldx	$(y) \rightarrow j$	Load into index
Store	sta	$\left.\begin{matrix}(A)\\(B)\\(C)\\(D)\\(F)\end{matrix}\right\} \rightarrow Y$	Store A
	stb		Store B
	stc		Store C
	std		Store D
	stf		Store F
	exa	$\left\{\begin{matrix}(Y) \rightarrow A\\(A) \rightarrow Y\end{matrix}\right.$	Exchange A
	ins	$(B) \& (A) \vee (B) \& (Y) \rightarrow Y$	Insert digits of A
	stx	$(j) \rightarrow y$	Store index
Add	add	$(A) + (Y) \rightarrow A$	Add
	sub	$(A) + \overline{(Y)} \rightarrow A$	Subtract
	dma	$\|(A)\| + \overline{\|(Y)\|} \rightarrow A$	Difference of magnitude
	and	$(A) \& (Y) \rightarrow A$	Logical and
	ori	$(A) \vee (Y) \rightarrow A$	Logical or - inclusive
	ore	$\left\{\begin{matrix}(A) \oplus (Y) \rightarrow A\\(A) \& (Y) \vee (C) \rightarrow C\end{matrix}\right\}$	Logical or - exclusive (and accumulate product)
	axm	$(j) \quad (y) \rightarrow y$	Add index to memory
	amx	$(j) \quad (y) \rightarrow j$	Add memory to index
Set bit	sbo	$1 \rightarrow Y_j$	Set j-th bit one
	sbz	$0 \rightarrow Y_j$	Set j-th bit zero
	sbc	$(Y_j) \rightarrow Y_j$	Set j-th bit complement

TABLE I (Continued)

Type	Mnemonic Code	Operation	Name
Step-Count	mul	$(A) \times (Y) \rightarrow AB$	Multiply
	div	$(AB) \div (Y) \rightarrow \begin{cases} A \text{ (remainder)} \\ B \text{ (quotient)} \end{cases}$	Divide
	sha sab shb	$\left.\begin{matrix}(A)\\(AB)\\(B)\end{matrix}\right\} \times 2^{(Y)} \rightarrow \left\{\begin{matrix}A\\AB\\B\end{matrix}\right.$	Shift A Shift AB together Shift B
	cya cab cyb	$\begin{matrix}(A)\\(AB)\\(B)\end{matrix} \text{ cyc } (Y) \rightarrow \left\{\begin{matrix}A\\AB\\B\end{matrix}\right.$	Cycle A Cycle B together Cycle B
	nab	$\begin{cases}(AB) \times 2^{nf} \longrightarrow AB \\ (Y) - nf \longrightarrow D\end{cases}$	Normalize AB
	coa	$(Y) + no \longrightarrow D$	Count ones in A
In-out	rds	$\left\{\begin{matrix}(Y) \rightarrow 10\\(10) \rightarrow Y\end{matrix}\right\}$	Read and shift
	rdn	$\left\{\begin{matrix}(Y) \rightarrow 10\\(10) \rightarrow Y\end{matrix}\right\}$	Read without shift
Jump	jpe	If $(E_j) = 1$, then $y \rightarrow P$	Jump if j-th bit of E is a one
	jpp	If any $(A) > 0$ then $Y \rightarrow P$	Jump if the contents of any A is positive
	jpn	If any $(A) \leq 0$	Jump if the contents of any A is ngeative
	jpz	If any $(A) = 0$	Jump if the contents of any A is zero
	jpo	If any (A) overflowed	Jump if the contents of any A has overflowed
	jxp	If $(j) \geq 0$, then $(j) - cf \rightarrow j$, $y \rightarrow P$	Jump if index positive and decrease index
	jxn	If $(j) < 0$, then $(j) + cf \rightarrow j$, $y \rightarrow P$	Jump if index negative and increase index
	jpu	$\left\{\begin{matrix}\text{If cf} = 1,3, \text{ then } (P)+1 \rightarrow j\\ \text{If cf} = 0,1, \text{ then } y \rightarrow P\\ \text{If cf} = 2,3, \text{ then } Y \rightarrow P\end{matrix}\right\}$	Jump unconditionally
Misc.	ios opr		In-out select Operate

TABLE II

Notation	Meaning
$\longrightarrow$	goes into
(x)	contents of x
$Y = y + (j)$	indexed memory address
$\lvert (x) \rvert$	magnitude of (x)
$\overline{(x)}$	one's complement of (x)
&	logical and operation
$\vee$	inclusive or operation
$\oplus$	exclusive or operation
$+$	one's complement addition
nf	number of shifts to normalize
no	number of ones
Y_j	j-th digit of register Y

III. THE LINCOLN TX-2 INPUT-OUTPUT SYSTEM

A. Introduction

The input-output system of the Lincoln TX-2 computer contains a variety of input-output devices suitable for general research and control applications. The system is designed in such a way that several input-output devices may be operated simultaneously. Since the computer is experimental in nature, and changes in the complement of input-output devices are anticipated, the modular scheme used will facilitate expansion and modification. The experimental nature of the computer also requires that the input-output system provide a maximum of flexibility in operating and programming for its input-output devices.

The input-output devices, currently scheduled for connection to TX-2, include magnetic-tape units for auxiliary storage; photoelectric paper-tape readers for program input; a high-speed printer, cathode-ray-tube displays, and Flexowriters for direct output; analog-to-digital conversion equipment; data links with other computers; and miscellaneous special-purpose equipment. This section will not be concerned with the details of these devices, but will limit itself to a discussion of the logical incorporation of them into the system.

In describing the TX-2 input-output system, occasional reference will be made to certain aspects of the design of other parts of the TX-2 as set forth in Section II.

B. The Multiple-Sequence Program Technique

Of the various organizational schemes which permit the simultaneous operation of many devices, we have chosen the "multiple-sequence program technique" for incorporation in TX-2. A multiple-sequence computer is one that has several program (instruction) counters. If the program sequences associated with these program counters are arranged to time-share the hardware of the central computer, a machine can be obtained which will behave as if it were a number of logically separate computers. We call these logical computers sequences and therefore refer to TX-2 as a multiple-sequence computer. By associating each input-output device with such a sequence, we effectively obtain an input-output computer for each device.

Since the one physical computer in which these sequences operate is capable of performing only one instruction at a time, it is necessary to interleave the sequences if they are to operate simultaneously. This interleaving process can take place aperiodically to suit the needs of and under the control of, whatever individual input-output devices are operating. The number of sequences which can operate simultaneously, and the complexity of the individual sequences, is limited by the peak and average data-handling rate of the central computer hardware.

In a multiple-sequence computer, the main body of the computation can be carried out in any sequence, but if maximum efficiency of input-output operation is to be achieved, the bulk of arithmetic operations must be confined to a few special sequences, called main sequences, which have no associated input-output devices. The input-output sequences may then be kept short, and a large number of them can be executed at once.

C. Multiple-Sequence Operation in TX-2

In TX-2, one-half of the index-register memory has been made available for storing program counters. Thus, a total of 32 sequences may be operated in the machine. (Actually an additional sequence of special characteristics is obtained by using index register number 0 as a program counter. This special sequence will be discussed later). Some of these sequences are associated with input-output devices. Others perform functions, such as interpreting arithmetic overflows that are called into action by conditions arising within the central computer. Finally, there are the main sequences which are intended to carry out the bulk of the arithmetic computations performed by the machine.

A priority scheme is used to determine which sequence will control the computer at a given time. If more than one sequence requires attention at the same time, control of the machine will go to the sequence having the highest priority and instructions addressed by its program counter will be executed.

Table I is a list of the sequences currently planned for inclusion in TX-2. They are listed in approximate order of priority with the highest at the top. Asterisks mark sequences which are not associated with any particular in-out device. A special sequence (number 0) has first priority and will be used to start any of the other sequences at arbitrary addresses. The next two sequences interpret alarms (under program control). These three sequences have the highest priorities, since they must be capable of interrupting the activities of other sequences. The input-output devices follow, with high-speed, free-running units carrying next highest priorities. The main sequences (we anticipate three) are at the bottom of the list. The priority of any sequence may be easily changed, but such changes are not under program control. Priorities are intended to remain fixed under normal operating conditions. The list totals about 25 sequences, leaving eight spaces for future expansion.

Switching between sequences is under the control of both the input-output devices (generalized to include alarms, etc.) and the programmed instructions within the sequence.

Once a sequence is selected and its instructions are controlling the computer, further switching is under control of the programmed instructions. Program control of sequence switching is maintained through two bits, called the break and dismiss bits, in each instruction. The break bit governs changes to higher-priority sequences. When the break bit permits a change, and some higher-priority sequence requests attention, a change will be made. The dismiss bit indicates that the sequence has completed its operation (for the moment, at least) and that lower-priority sequences may receive attention. The interpretation of the break and dismiss bits will be discussed in more detail.

D. The TX-2 Input-Output Element

The TX-2 input-output element is shown schematically in Fig. 13. It consists of a number of input-output devices, associated buffers, and a sequence selector. Each device has enough control circuitry to permit it to operate in some selected mode once it has been placed in that mode by signals from the central computer. Associated with each device is a buffer storage of appropriate size. This buffer may be large or small, to suit individual data-rate requirements, but the buffers used in TX-2 will generally be the smallest possible. For the most part, buffering for only one line of data from the device (e.g., 6 bits for a paper-tape reader) will be provided. Each input-output device is associated with one stage of the sequence selector. The sequence selector provides the control information necessary for proper interleaving of the program sequences. When it is desired to add a new input-output device to the computer, the three packages, in-out unit, buffer, and sequence-selector stage, must be provided.

As shown in Fig. 13, data is transferred between the input-output element and the central computer by way of the exchange element. Fig. 13 indicates two-way paths between the E-register and all in-out buffers. Actually, most devices are either readers or recorders, but not both, and therefore require one-way paths only. Only the necessary paths are provided; the drawing simply shows the most general case.

Signals from the sequence selector connect the appropriate buffer register to the E-register to transfer data. When a sequence is selected (i.e., its program counter is supplying instruction locations), the associated buffer is connected to the E-Register, and all other buffers are disconnected. A read instruction will effect a transfer of information between the buffer and the E-register. A particular buffer is thus accessible only to read instructions in the sequence associated with the buffer's in-out unit.

Fig. 13 shows paths from the Sequence Selector to a coder which proves an output called the program-counter number. These paths are used in the process of changing sequences to be described in a later section.

C-69383
C60-371
S-1716

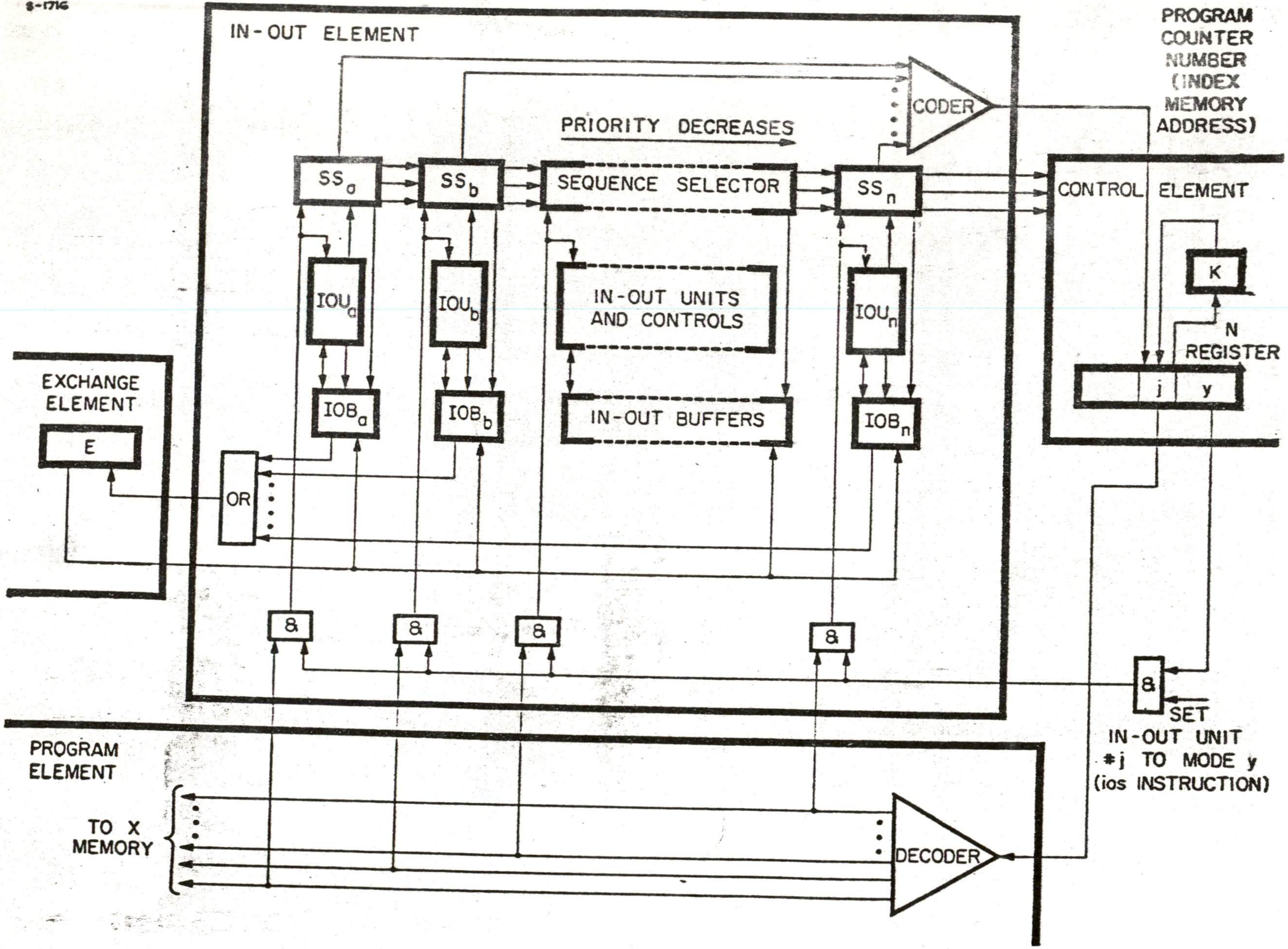


Fig. 14. TX-2 In-Out Element

6M-4968 S2 40

Fig. 13 also shows paths for mode selection in the in-out element. The use of these paths is described in the next section under 2., ios.

E. Input-Output Instructions

In addition to the break and dismiss bits on all instructions, the programmer has three computer instructions for operating the input-output system. There are two read instructions, rdn and rds, which transfer data between the in-out devices and the central computer memory. The third instruction, ios, selects the mode of operation of the in-out devices.

1. rdn and rds

Both of the read instructions obtain a word from memory. If the in-out device associated with the sequence in which the read instruction occurs is in a reading (input) mode, appropriate bits of the memory word are altered, and the modified word is replaced in memory. If the in-out device is in a recording (output) mode, appropriate bits of the memory word are fed to the selected in-out buffer, and the word is replaced in memory. Thus, the same read instruction suffices for both input and output operations. The distinction between rdn and rds lies in the assembling of full memory words from short buffer words. An rdn instruction will place the 6 bits from a tape reader in the right 6 bits of a 36-bit memory word. The remaining 30 bits will be left unchanged. An rds instruction for the same tape reader will place the 6 bits in a splayed pattern (every sixth bit across the memory word) and will shift the entire word one place to the left before replacing it in memory. Except for the shift, the other 30 bits remain unchanged. A sequence of 6 rds instructions, one for each of 6 tape lines and all referring to the same memory address, will suffice to assemble a full 36-bit word.

The distinction between rdn and rds could be obtained from mode information in the in-out device, but the inclusion of both instructions in the order code allows the programmer to interchange the two types freely to suit his needs. The rdn instruction makes use of the permutation aspect of the TX-2 configuration control and is, therefore, particularly convenient for dealing with alphanumeric Flexowriter characters. Configuration is not applicable to the rds instruction.

2. ios

The ios instruction serves to put a particular in-out device into a desired mode of operation. The j-bits of the instruction word, normally the index register number, in this case specify the unit number of the in-out device. This number is the same as the program

C-69382
C60-373
S-1717

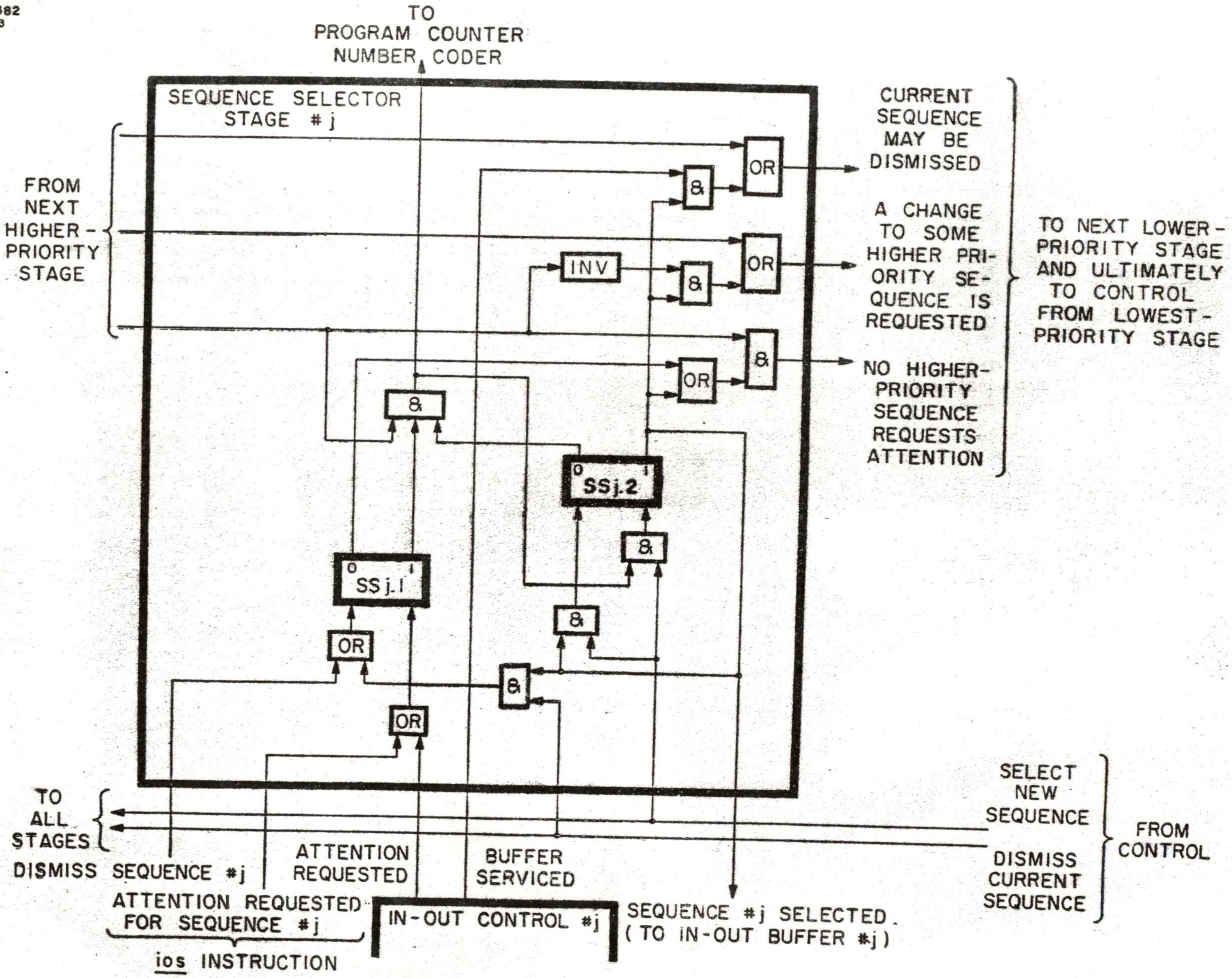

TO PROGRAM COUNTER NUMBER CODER
SEQUENCE SELECTOR STAGE #j
FROM NEXT HIGHER-PRIORITY STAGE
OR
&
INV
&
OR
&
OR
&
SSj.2
0
1
&
&
SSj.1
0
1
OR
&
OR
CURRENT SEQUENCE MAY BE DISMISSED
A CHANGE TO SOME HIGHER PRIORITY SEQUENCE IS REQUESTED
NO HIGHER-PRIORITY SEQUENCE REQUESTS ATTENTION
TO NEXT LOWER-PRIORITY STAGE AND ULTIMATELY TO CONTROL FROM LOWEST-PRIORITY STAGE
TO ALL STAGES
SELECT NEW SEQUENCE
DISMISS CURRENT SEQUENCE
FROM CONTROL
DISMISS SEQUENCE #j
ATTENTION REQUESTED
BUFFER SERVICED
ATTENTION REQUESTED FOR SEQUENCE #j
ios INSTRUCTION
IN-OUT CONTROL #j
SEQUENCE #j SELECTED (TO IN-OUT BUFFER #j)

Fig. 13. TX-2 Sequence Selector Stage

counter number for the associated sequence, although the correspondence is not necessary. The y-bits of the instruction word specify the mode of operation in which the unit is to be placed. Two of the y-bits are sent directly to the jth sequence selector stage and serve to control the sequence, regardless of the mode of its associated in-out device. These two bits allow ios instructions to arbitrarily dismiss or request attention for any sequence in the machine. By means of these instructions, one sequence can start or stop all others in the machine. A third y-bit determines whether the mode of the in-out device is to change as a result of the instruction. If it is to change, the remaining 15 bits specify the new mode. An ios instruction occurring in any sequence can thus start or stop any sequence and/or change the mode of its in-out device.

A further property of the ios instruction is that it leaves in the E-register a map of the state of the specified in-out control prior to any changes resulting from the instruction itself; ios instructions may, therefore, be used to sense the state of the in-out system without altering it in any way.

F. Sequence-Changing and Operation of the Sequence-Selector

At some point just before the completion of the instruction memory cycle in TX-2, the Control must decide whether the next instruction would be taken from the current sequence or from some new sequence. The information on which this decision must be based comes from the break and dismiss bits of the instruction word currently in use and from the sequence selector. Fig. 14 is a detailed drawing of one stage of the sequence selector. All stages, except that with the highest-priority are identical. The lowest-priority stage returns the final three control signals to the control element.

Each stage of the sequence selector retains two pieces of information concerning its associated sequence. One flip-flop (ss j.1) remembers whether or not the sequence is selected (i.e., whether or not it is receiving attention). The priority signal (labelled no higher priority sequence requests attention) passes from higher to lower priority stages until it encounters a stage which requests, but is not receiving, attention. Such a stage is said to have priority at the moment, and its output to the program-counter-number coder prepares the number of the new program counter in anticipation of a sequence change.

The process of changing sequences involves storing the program counter for the old sequence and obtaining the counter for the new. Actually, to speed up the over-all process, the new program counter is obtained first, so that it may be used while the old is being stored. Using the paths shown in Fig. 13, the new program counter number is placed in the j-bits of the N-register. The new program counter is then obtained from the X-memory and interchanged with the old program counter contents which have been in the P-register.* The K-register, which has been holding the old program counter number since the last sequence change, is now interchanged with the j-bits, and the old counter is stored at the proper location in the X-memory. The state of the sequence selector is changed, to conform to the change of sequence, by sending a select new sequence command from Control. This command clears the ss j.2 flip-flop in the old-sequence stage and sets the ss j.2 flip-flop to a ONE in the new-sequence stage.**

F. Interpretation of the Break Bit

The programmer uses the break bit of an instruction word to indicate whether or not change to a higher priority sequence may occur at the completion of the instruction. The fact that a programmer permits a break does not mean that the sequence has completed its current task, but merely that no harm will be done if a change to some higher-priority sequence is made. Breaks should be permitted at every opportunity if a number of in-out devices are operating. The sort of situation in which a break cannot be permitted occurs when the E-register is left containing information which the program requires at a later step. If a change occurred in this case, the contents of the E-register would be destroyed and lost to the program.

When a break is permitted by the current instruction, a sequence change will actually take place only if some higher-priority sequence requests attention. A signal from the sequence selector to the control element provides this information (Fig. 14). When a break type of sequence change is made, the ss j.1 flip-flop in the sequence selector remains unchanged, and the sequence which was abandoned in favor of one of a higher-priority continues to request attention.

H. Interpretation of the Dismiss Bit

The dismiss bit is used by the programmer to indicate that the sequence presently in use has completed its task. To provide synchronization in the in-out system, dismiss bits must be programmed between attention requests from the in-out devices. In this case, the dismiss in-operation guarantees that the computer will wait for the next signal from the in-out device before proceeding with the associated program sequence.

The dismiss bit is also used to accomplish the halt function in

* The P-register is shown in Fig. 6 of Section II., P. 10.

**The relative timing of the central computer actions during the change process is shown in Fig. 8d., p. 19.

TX-2. A multiple-sequence computer halts when all sequences have been dismissed and all in-out units turned off. The priority signal from the sequence selector to the control element provides the information as to whether or not any sequence in the machine requests attention. When none request attention, the control stops all activity in the machine as soon as a dismiss bit appears on an instruction in the sequence being used. Activity is resumed in the machine as soon as some in-out device or push button requests attention.

The sequence change which results from a dismiss bit is identical with that resulting from a break except that a dismiss current sequence command accompanies the select new sequence command from Control to the Sequence Selector (Fig. 14).

I. Starting a Multiple-Sequence Computer

In a single-sequence computer the starting process involves resetting the program counter to some arbitrary value and starting the control. In a multiple-sequence computer, the program counter for a particular sequence must be reset and the sequence started. In TX-2 a special sequence (number 0) has the highest priority and is used to facilitate starting. This sequence has the special feature that its program counter always starts at an initial memory location specified by a set of toggle switches. Attention for the sequence is requested by pushing a button on the console. By executing a short program stored in the toggle-switch registers of the V-memory, this sequence can start (or stop) any other sequence in the machine. The starting process for an arbitrary sequence involves resetting its program counter by means of an ldx (load index register) instruction, and starting its sequence with an ios instruction.

J. The Arithmetic Element in Multiple-Sequence Operation

While efficient operation requires that the bulk of arithmetic operations be carried out in a main sequence, the arithmetic element in TX-2 is available to all sequences. Since once a change has been made to a higher-priority sequence, control cannot return to a lower-priority sequence until the higher-priority one has been dismissed, a simple rule allows the arithmetic element to be used in any sequence without confusion. If, whenever a higher-priority sequence requires the arithmetic element, it stores the contents of any registers it will need (A,B,C,D, or F) and reloads them before dismissing, all lower-priority sequences will find the registers as they left them. This storing and loading operation requires time and, therefore, lowers the total handling capacity, but the flexibility obtained may well be worth the loss in capacity.

The step-counter class of arithmetic element instructions is a special problem. These instructions can require many microseconds to complete, and while TX-2 is designed to allow in-out and program element instructions to take place while the arithmetic element is busy, the case can arise in which an arithmetic element instruction (load, store, etc.) appears before the AE is finished with a step-counter class instruction. The machine would normally wait in an inactive stage until the operation is complete, but since there is a chance that some higher-priority sequence may request attention in the interim and have instructions which can be carried out, provision is made to keep trying changes to higher-priority sequences as they request attention. The machine thus waits in an inactive state only when no higher-priority sequences have instructions which can be performed. This provision allows the programmer to ignore the arithmetic element in considerations of peak- and average-peak rate calculations when he desires to operate a maximum number of in-out devices.

K. Conclusions

Multiple-sequence operation of input-output devices, as realized in TX-2, has a number of significant characteristics. Among them are:

1. A number of in-out devices may be operated concurrently with a minimum of buffering storage.

2. Machine time is used efficiently, since no time need be lost waiting for input-output devices to complete their operation. Other machine activity may proceed meanwhile.

3. Each input-output device may be treated separately for programming purposes. Efficiency of operation is obtained automatically when several separately programmed devices are operated simultaneously, although average- and peak-rate limitations must be considered.

4. Maximum flexibility in programming for input-output devices is obtained. The full power of the central machine may be used by each input-output sequence if desired. Routines for each device may be as long or as short as the particular situation requires.

5. The modular organization of the input-output equipment simplifies additions and modifications to the complement of in-out devices

6. The organization of buffering storage allows the amount and kind of such storage to be tailored to the needs of the individual devices and the data-handling requirements to be met by the system.

7. The multiple-sequence program technique appears to be particularly well suited to the operation of a large number of relatively slow input-output devices of varying characteristics, as opposed to a smaller number of high-speed devices.

TABLE I

TX-2 Sequence Assignments in the Order of Their Priority

* Start-Over (special index register number 0 sequence)
* In-out alarms
* Arithmetic alarms (overflows, etc.)
Magnetic Tape units (several sequences)
High-speed printer
Analog-to-digital converter
Photoelectric paper tape readers (several sequences)
Light Pen (photoelectric pick-up device)
Display (several sequences)
MTC (Memory Test Computer)
TX-0
Digital-to-analog converter
Paper tape punch
Flexowriters (several sequences)
* Main sequences (three)

* The sequences have no input-output device.

IV. MEMORY UNITS IN THE LINCOLN TX-2

A. Introduction

The three, high-speed memories in TX-2 are random access and use ferrite cores. The largest has 65,536 words, 37 digits long, and operates at a cycle time of 6.5 μsecs; it is referred to as the S-memory. The T-memory is driven entirely by transistors and has a capacity of 4,096 37-digit words with a 5.5-μsec cycle time. The smallest and fastest is the X-memory which has a capacity of 64 19-digit words; its external word-selection and two cores per bit make possible an access time of 0.8 μsec and a cycle time of 4.0 μsec.

B. S-Memory (65,536 Words)

The S-Memory, (Fig. 15) is a coincident-current, magnetic-core unit with a storage capacity of 65,536 37-bit words. The bits in the word are read out in parallel, with a cycle time of 6.5 μsec and an access time of 2.8 μsec. (Cycle time is the time between successive strobe pulses and access time is the minimum delay between setting the address register and strobing). The block diagram (Fig. 16) shows that two, 256-position, magnetic-core switches are used to supply the READ and WRITE current pulses to the X and Y selection lines. The operating characteristics of these switches are such that the contents of the address register are no longer needed after the READ half of the cycle, and the in-

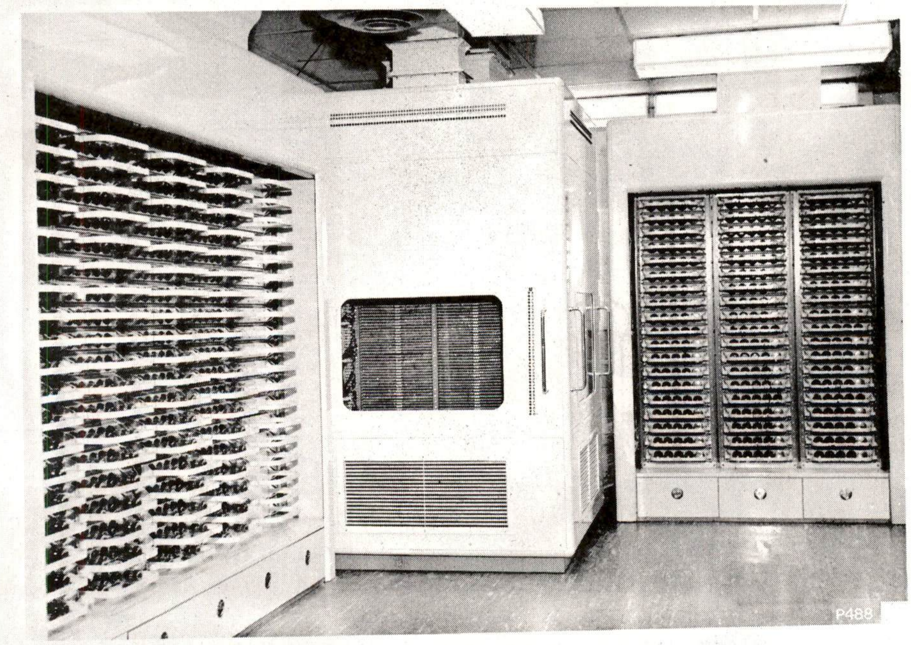


Fig. 15. S-Memory

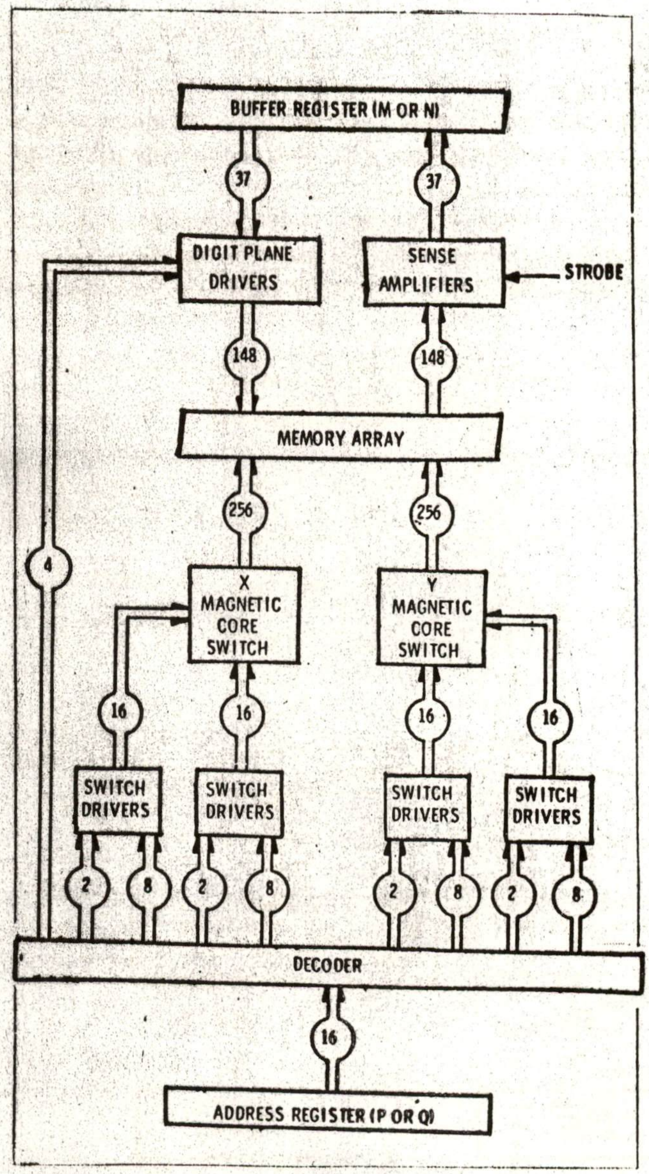


Fig. 16. Block Diagram, S-Memory

terval between READ and WRITE may be extended several μsecs under computer control to permit the other operations to occur. Two coordinates are used to select a register during READ and three coordinates are used for WRITE. In each case, a 2:1 current selection ratio is used. The S-memory with 604 tubes and 1406 transistors, is a 37-digit version of the 19-digit TX-0 memory that has been described in the literature.[1] The basic operation of this type memory has also been described and will not be repeated here.[2]

C. "T" Memory (4096 Words)

The coincident-current, magnetic core, T-memory has 4096 37-bit words (see Fig. 17). The bits of the word are read out in parallel with a cycle time of 5.5 μsec and an access time of 2.4 μsec. A timing diagram is shown in Fig. 18. The computer program can extend, by any amount, the interval between READ and WRITE. Again, A 2:1 current selection ratio is used with two coordinates used to select for READ and three coordinates for WRITE. A total of 1460 transistors and 64 diodes are used (not counting the address and buffer registers and control).

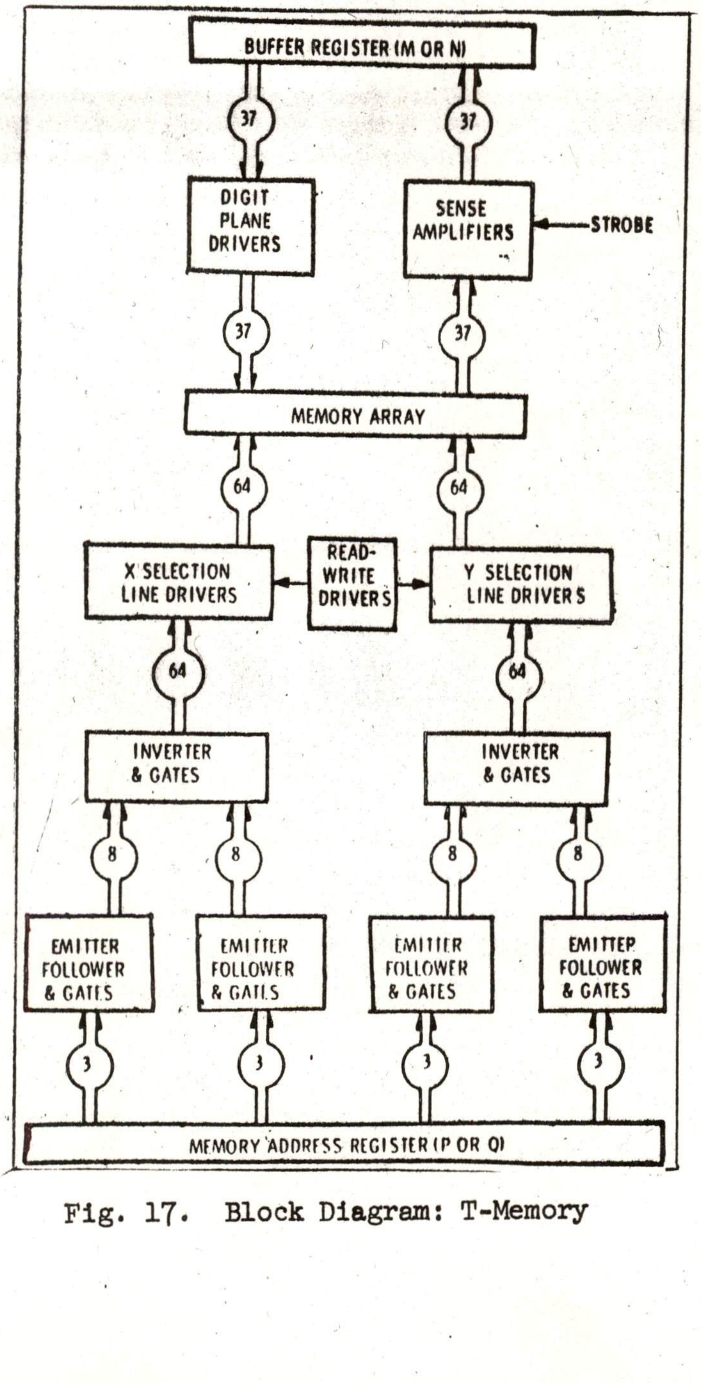


Fig. 17. Block Diagram: T-Memory

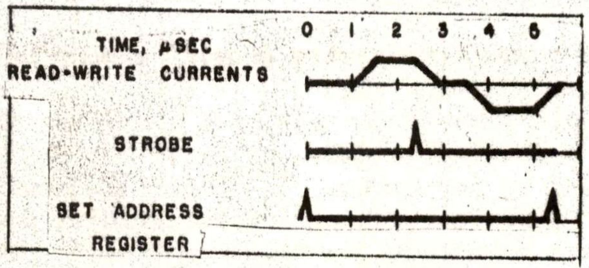


Fig. 18. Timing, T-Memory

1. Mechanical Features

The 64 x 64 x 37 pluggable array (Fig. 19) is contained within a 5-inch cube. (One plane is used for parity checking). The ferrite cores are 47 mils O.D., 27 mils I.D., and 12 mils thick. The material is similar to General Ceramics' S-1 and that is also used in the S-memory.

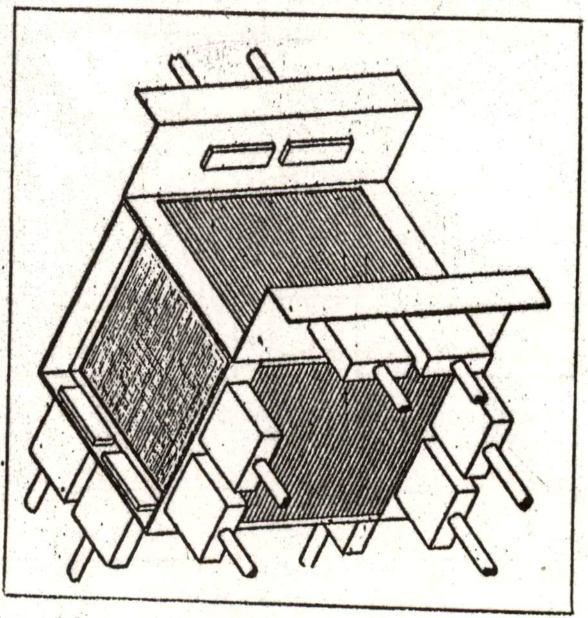


Fig. 19. Pluggable Array, T-Memory

2. Selection Circuits

The logic and circuitry of the memory-address register decoder is shown in Figs. 17 and 20. The input to the emitter follower AND gates is a d-c level of zero or -3 volts. Silicon diodes add a bias shift without the loss that would be associated with

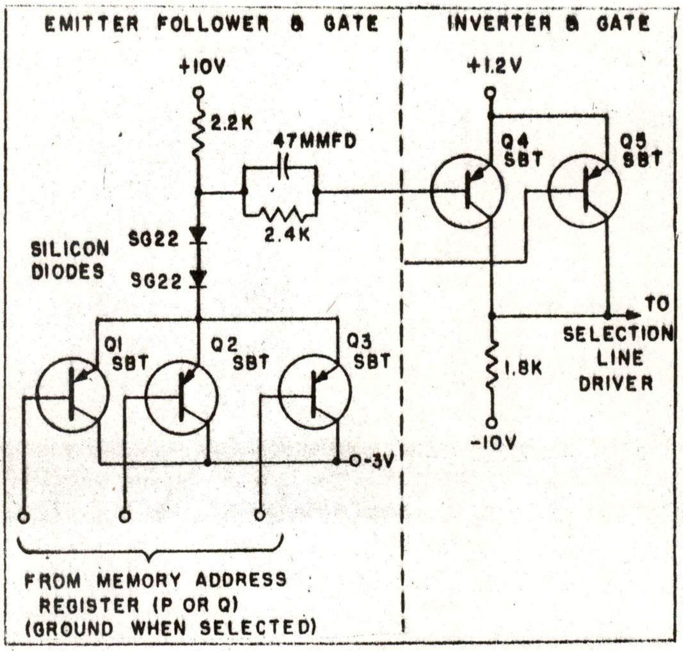


Fig. 20. One Channel, Emitter Follower and Inverter AND Gates, T-Memory

a simple voltage divider. The +1.2-volt supply for the inverter AND gates is a single divider for the whole memory - the load on the divider is constant.

Each inverter AND gate feeds a selection line driver (Fig. 21). Q4 passes the full selection line current (+250 and -250 ma.) and is selected to have a minimum β of 10 for current of either polarity. The transient back voltage of the selection line for this current is 12 volts. The series-connected emitter followers (Q1 and Q2) supply the large amount of current needed to cut off Q3 quickly during selection. The load for Q2 is such that a large surge of current is delivered to Q3 to turn it on quickly when this line is deselected.

Fig. 22 shows the circuit geometry planned for the read-write driver. Currents for READ and WRITE operations are supplied by the 150-volt supplies through R2 and R1 respectively. The currents are switched into the proper selection line drivers by cutting off Q1 or Q2.

3. Digit Circuits

The input of the digit-plane driver shown in Fig. 23 is a standard logic level of 0 or -3 volts. Q2 acts as a switch which connects a voltage source across the digit winding and the parallel R-C

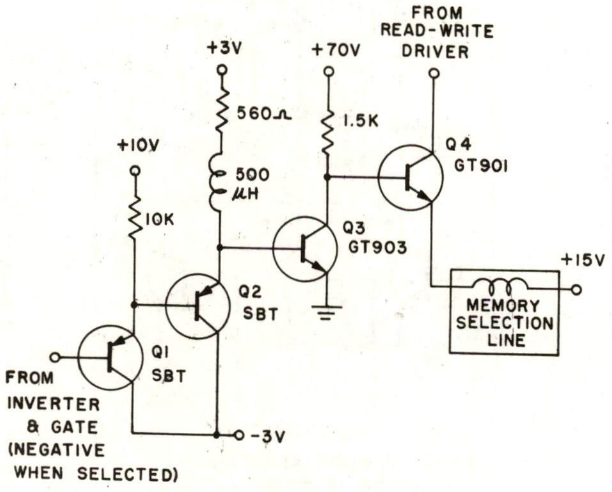


Fig. 21. One Channel, Selection Line Driver, T-Memory

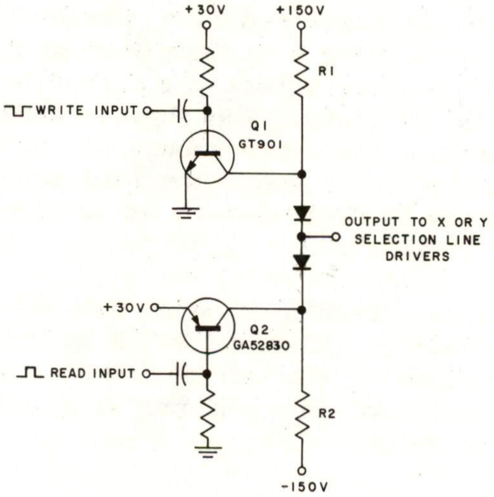


Fig. 22. Read-Write Driver (2 needed), T-Memory

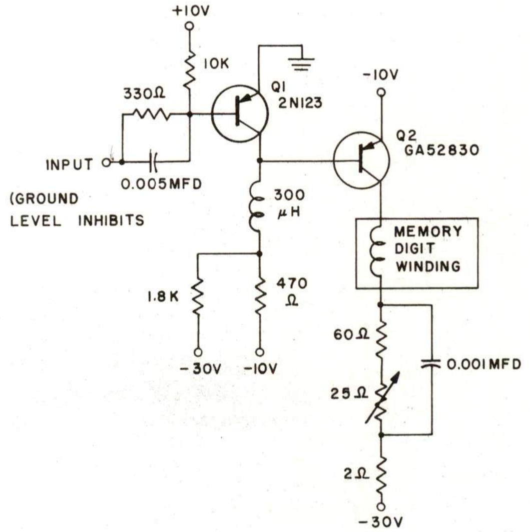


Fig. 23. Digit-Plane Driver, T-Memory

combination. It can be turned on and off very quickly by virtue of the large overdrive of current into its base which is supplied by the combination of Q1 and its collector load. The adjustable resistor is used to set the correct d-c inhibit current which is measured across the 2-ohm resistor; the 0.001-mfd capacitor speeds the current rise time in the digit winding.

In the sense amplifier, shown in Fig. 24, two 160-ohm resistors terminate the sense winding and tie it down to ground. Constant d-c emitter currents are supplied to Q1 and Q2 by the 13K resistors. The voltage divider forms a stable, d-c, collector-to-base voltage and is composed of 1.3K resistors, which form a virtual center tap on the sense-winding by operating in conjunction with the 3.3K resistor. Thus, with both the d-c emitter current and the base-to-collector voltage stabilized, the operating point of Q1 and Q2 is also stabilized. Two 60-μfd electrolytic capacitors in series tie the emitters of Q1 and Q2 together for signal gain. The 6.8K resistors damp the transformer windings. There is no gain for a common-mode input, and a gain of about 22 for a difference-signal input (output measured across half the transformer secondary). The current through the 16K resistor normally flows through Q5, but can be interrupted by an input signal that is large enough to overcome the bias on the 68-ohm resistor. The 5K variable resistance is adjusted so that a 50-mv input signal will be just enough.

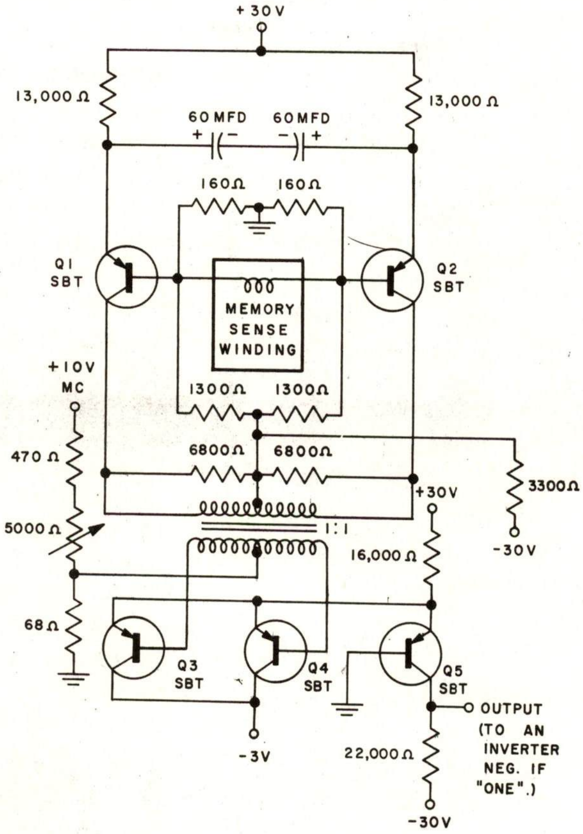


Fig. 24. Sense Amplifier, T̃-Memory

The normal ONE input signal is 100 mv. All of the +10-volt, marginal-check lines are tied together so that the sense-amplifier clip levels may be remotely checked to determine the memory margins.

D. X-Memory (64 Words)

There are three modes of operation of the X-memory: (1) READ-WRITE, (2) READ, and (3) CLEAR-WRITE. This magnetic-core memory requires external word-selection (two cores per bit) and has a storage capacity of 64 19-bit words. The bits of the word are read out in parallel with a cycle time of 4.0 μsec 3.8 and an access time of 0.8 .6 μsec. In this memory, cycle time is the time between successive strobe pulses with a repetitive READ-WRITE cycle; access time is, again, the minimum delay between setting the address register and strobing. A total of 434 transistors, 8 diodes, and 1 vacuum tube are used, excluding those for the address and buffer registers and control.

1. Operating Principle

The winding configuration of the single-plane unit is shown in Fig. 25. A word is selected externally by connecting the upper end of a word line (pt. Y, for instance) to a fixed point. The READ driver then puts out a current pulse 4-1/3 times that required to switch a core on a 2:1 basis (Fig. 26). Only one of the two cores (per bit) is switched to the cleared state by this pulse because any previous WRITE operation would have left one core set and one cleared. The switched core generates a pulse in its digit line. This line passes through one of the cores in the same direction as the word line and through the other core in a direction opposite to that of the word line. Thus, the polarity of the pulse on the digit line during READ, indicates whether a ONE or a ZERO is being read out.

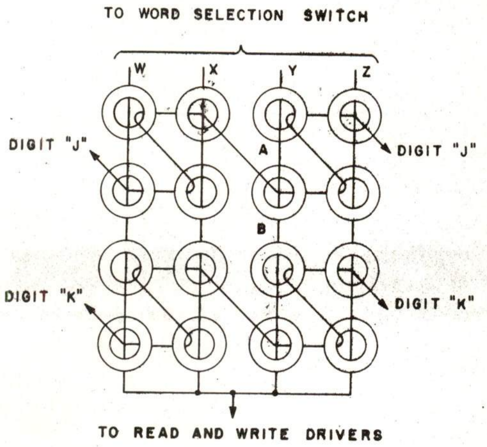


Fig. 25. Winding Configuration, X-Memory

Current always flows in the digit winding. The polarity is controlled by the flip-flop associated with that digit, and the amplitude is 1/3 of the required switch current in a 2:1 system. The digit current is swamped out by the large read current and, therefore, has no effect during READ. During WRITE, a current of 2/3 is sent down the selected word line. The digit current adds to the write current in one core and subtracts from it in the other, so that one core has a current of unity and the other a current of 1/3. Thus, the current ratio used during WRITE is 3:1 with a disturb current of no more than 1/3. Fig. 26 shows the timing and current relationships in cores A and B of Fig. 25.

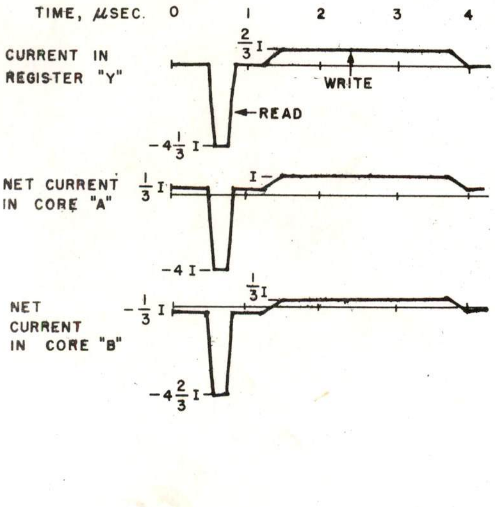


Fig. 26. Timing Diagram, X-Memory

The cores used for the X-memory are 47 mils OD, 27 mils ID, and 12 mils thick. The core material is similar to General Ceramics' type S-3 which differs from S-1 in that the coercive force required for switching is lower and the switching time is longer. We needed the low coercive force so that we could drive the cores with transistors. Access time remained short because, even with transistors, we could overdrive the cores during

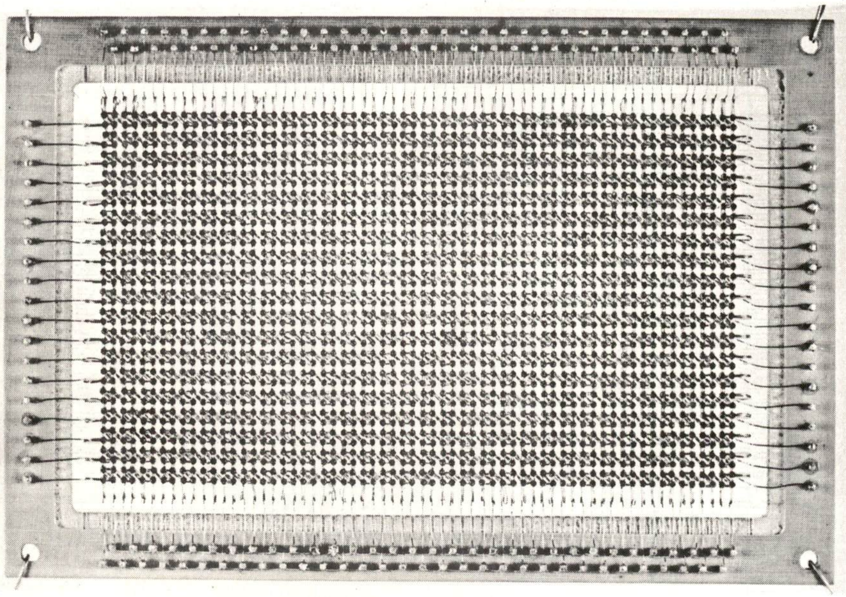


Fig. 27. 64 x 38 Memory Plane, X-Memory

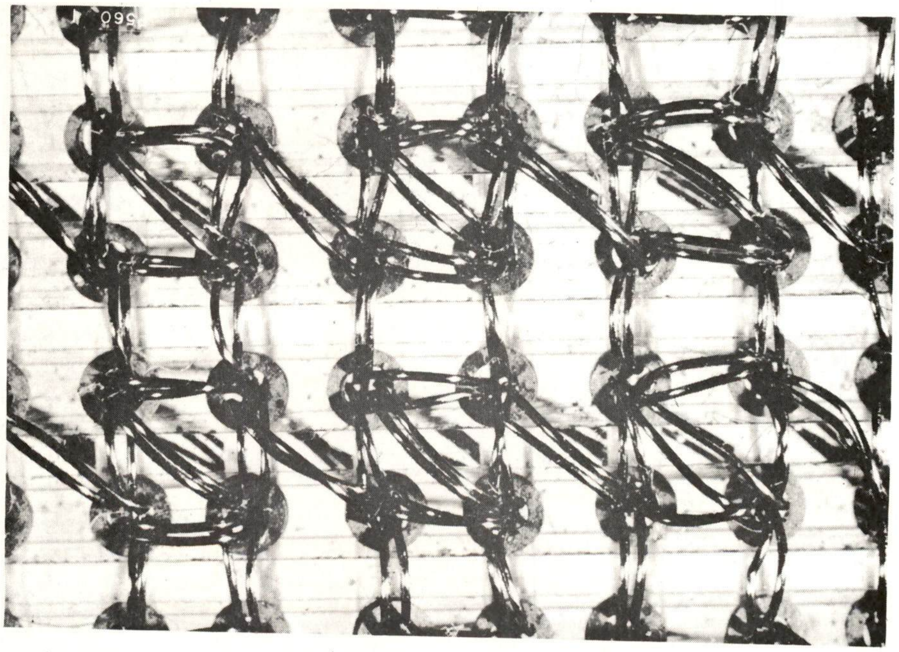


Fig. 28. Section of Memory Plane, Enlarged, X-Memory

READ. Each winding makes sour turns on each core through which it passes. Fig. 27 shows the complete memory plane (4-1/4 x 6-1/4 inches) and Fig. 28 shows part of it enlarged. The cores are mounted on a lucite plate; the wires pass through openings made by the intersection of milled slots on one side of the plate with similar slots on the other side that are milled at right angles to the first. With each winding making four turns per core, the digit current is 8 ma, the write-driver output current is 18 ma, and the read-driver current is 117 ma.

A block diagram of the X-memory is shown in Fig. 29. The method of word selection used is determined partially by the computer's use of the outputs of the j-bits decoder. Either of two write drivers are used and the output of the first level selection determines which one.

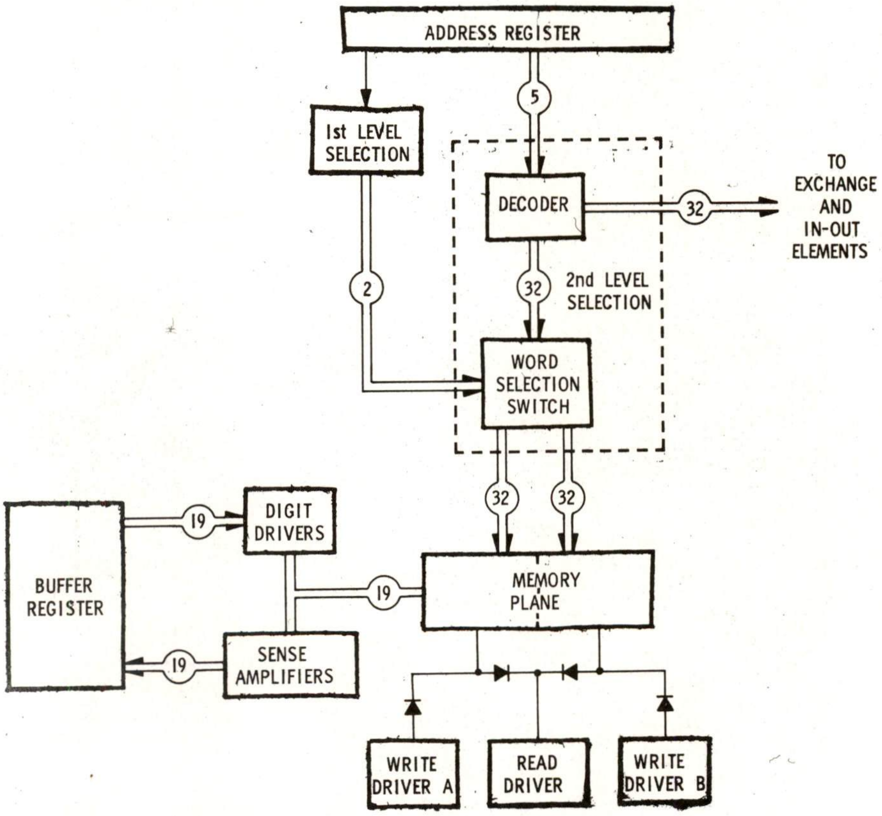


Fig. 29. Block Diagram, X-Memory

2. Selection Circuits

One channel of the selection circuit is shown in Fig. 30. The j-bits decoder uses 5-way emitter follower AND gates which drive parallel inverters (Q6 and Q7). The collector lead of these transistors provides an overdrive of base current into Q8 or Q9 during both selection and de-selection. When neither the read not the write driver is active, the word lines are free to float between 0 and -10 volts. Only one of the first level selection transistors (Q10 or Q11) will be saturated, so base current flows only into either Q8 or Q9. The read driver generates a negative pulse so

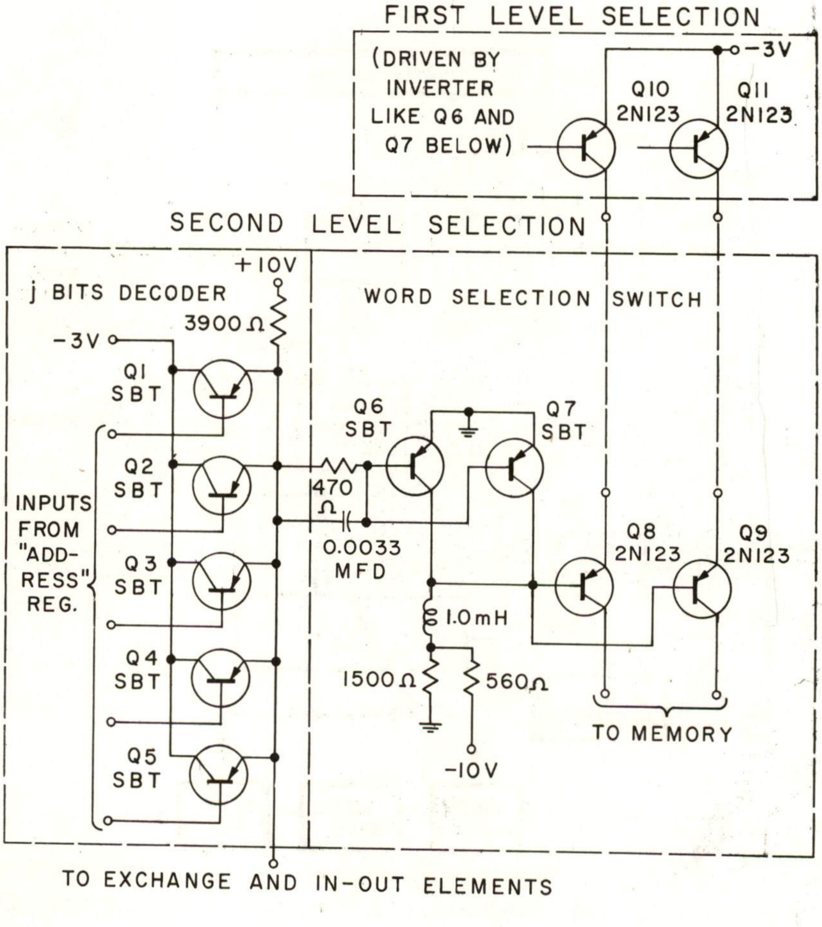


Fig. 30. Register Selection Circuit, X-Memory

that the large read current (117 ma) flows in the normal direction through the 2N123's. The write current flows in the reverse direction, but it is only 18 ma, and doesn't require a very high reverse β. A decoder such as that in the "T" memory would have used fewer transistors, but access time is at a premium here, so the faster circuit was used.

3. Read-Write Drivers

The read driver shown in Fig. 31 consists of three SBT transistors in series (because of the voltage needed) driving a 6197 pentode to saturation. The back voltage presented by the cores to ~~this driver is~~ constant because, as mentioned before, it always switches one of the two cores in each pair. The 5:1 transformer holds the tube load to a low value.

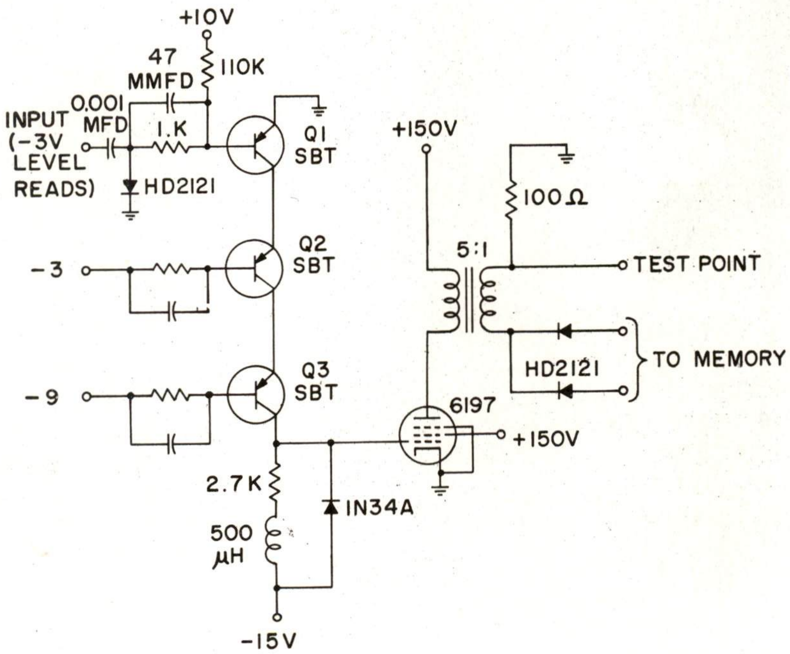


Fig. 31. Read Driver, X-Memory

The write driver (Fig. 32) is very simple - the current in the 1640-ohm resistor is switched into the memory load during WRITE by saturating Q2 which cuts off Q1. Since the selection circuits are returned to -3 volts, the output terminal of this circuit is always below ground.

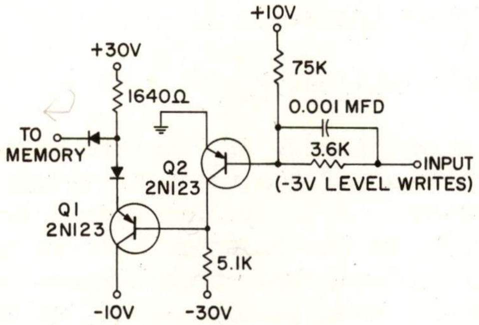


Fig. 32. Write Driver, X-Memory

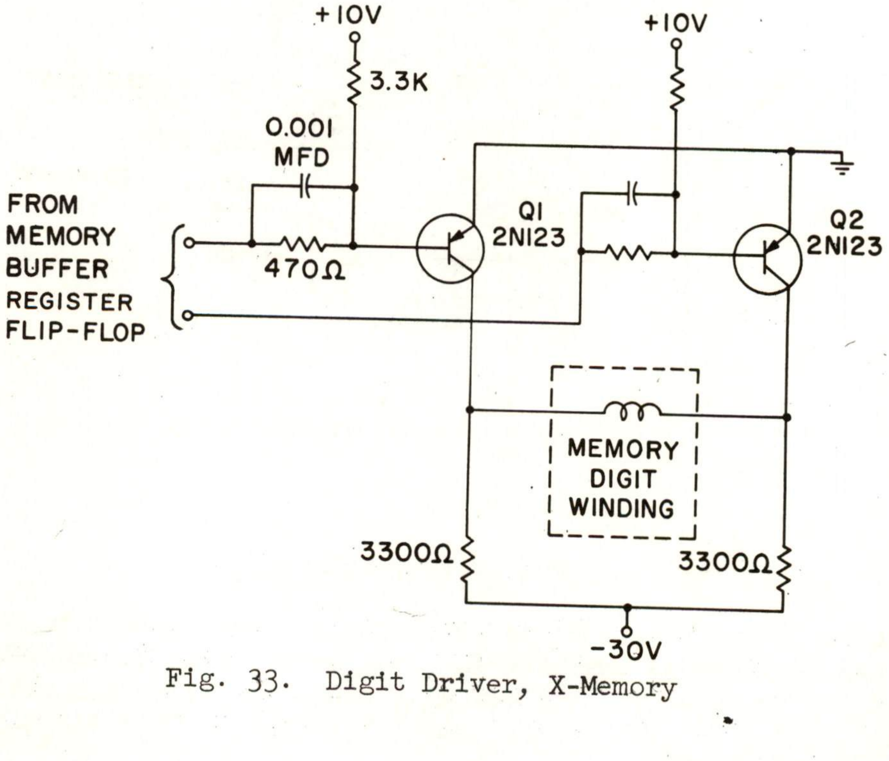


Fig. 33. Digit Driver, X-Memory

4. Digit Circuits

The digit driver (Fig. 33) is connected directly to the corresponding flip-flop in the buffer X-register. One of the two transistors is always saturated so that current always flows in the digit winding and in a direction determined by the flip-flop. The terminals of the digit winding are connected to the input stage (Q1 and Q2) of the sense amplifier shown in Fig. 34 which responds to the voltage difference between the inputs. The open-circuit READ signal on the digit winding is a 0.25-μsec pulse ± 0.5 volt amplitude. The sense amplifier loads the winding to reduce the pulse to about half of this amplitude. A saturation signal is fed to the gates Q3 and Q5 so that the strobe pulse forces the flip-flop to the correct position. If the signal on the free end of the digit winding is positive, the flip-flop is left in the same state; if negative, the flip-flop is complemented.

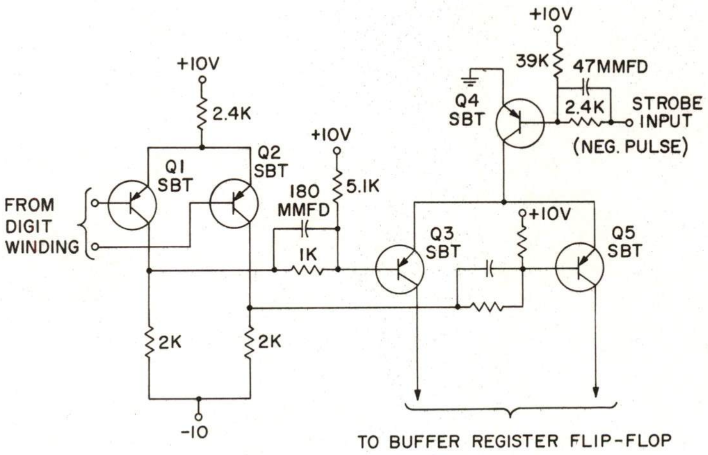


Fig. 34. Sense Amplifier, X-Memory

5. Modes of Operation

The three modes of operation of the X-memory are READ-WRITE, READ, and CLEAR-WRITE. READ-WRITE has been described above. The READ operation, used when the contents of two registers are needed quickly, performs the necessary function of clearing both cores in each bit before writing. When the computer returns to WRITE in registers that have had a READ cycle only, the CLEAR-WRITE cycle is used. CLEAR-WRITE is the same as READ-WRITE except that the strobe pulse is eliminated. Actually, a WRITE cycle alone would be sufficient but the CLEAR-WRITE cycle was added as an aid to program trouble-shooting, since if a WRITE operation should follow a previous WRITE operation on the same register, some bits would have both cores set. A subsequent READ would clear both cores, their outputs would subtract in the digit winding, and the response of the sense amplifier would be unpredictable.

E. Acknowledgment

The achievements reported above were the result of the imagineering of many people associated with core memory development at Lincoln Laboratory. Major contributions to the system and circuit design were made by S. Bradspies, G.A. Davidson, D.H. Ellis, and J.L. Mitchell. E.A. Guditz was responsible for most of the ideas incorporated in the mechanical design and packaging.

V. TX-2 CIRCUITRY

A. Circuit Configurations

Two basic circuits perform most of the logical operations in the TX-2 computer: a saturated transistor-inverter and a saturated emitter-follower. To the logic designer who works with them, these circuits can be considered as simple switches which are either open or closed.

The schematic diagram of an emitter follower and the symbol used by the logic designers is shown in Fig. 35. With a negative input, the output is "shorted" to the -3 volt supply, as through a switch. When several of these emitter followers are combined in parallel, as in Fig. 36, any one of them will clamp the output to -3V. We have then an OR circuit for negative signals and an

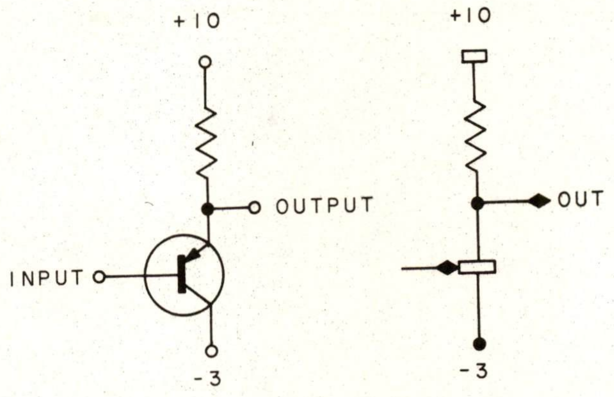


Fig. 35. Emitter Follower

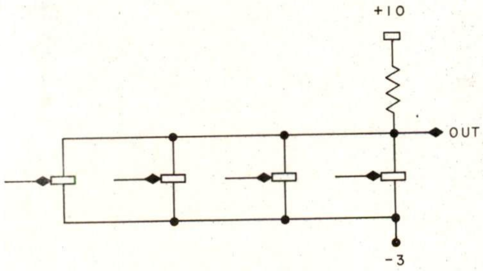


Fig. 36. Parallel Emitter Follower

AND circuit for positive signals. The transistor inverter is shown in Fig. 37 with its logic symbol. Basic AND and OR circuits result from the connection of these simple switches in series or parallel, as in Figs. 38 and 39. More complex networks like the TX-2 carry circuit use these elements arranged in series-parallel as shown in Fig. 40.

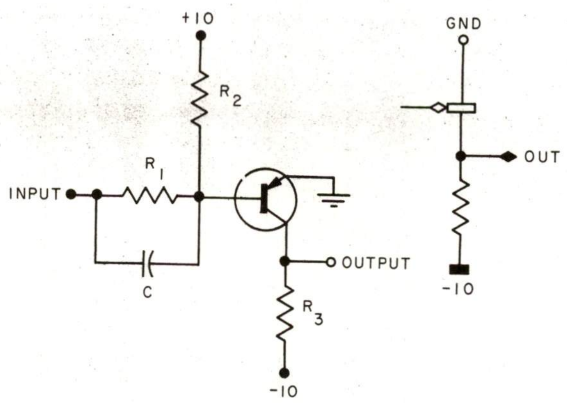


Fig. 37. Inverter

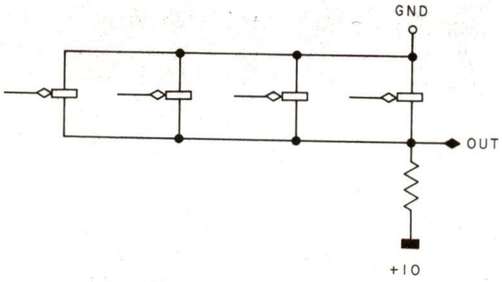


Fig. 38. Parallel Inverters

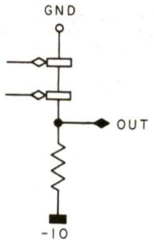


Fig. 39. Series Inverters

In Fig. 37 the resistor, R_1, is chosen so that under the worst combinations of stated component and power supply variations, the drop across the transistor will be less than 200 mv during the "on condition". R_2 biases the transistor base positive during the off condition to provide greater tolerance to noise, I_{co}, and signal variations. Capacitance, C, was selected to remove all of the minority carriers from the base when the transistor is being turned off. The effect of C on a test circuit driven by a fast step is shown in Fig. 41. Note that the delay due to hole storage is only a few millimicroseconds.

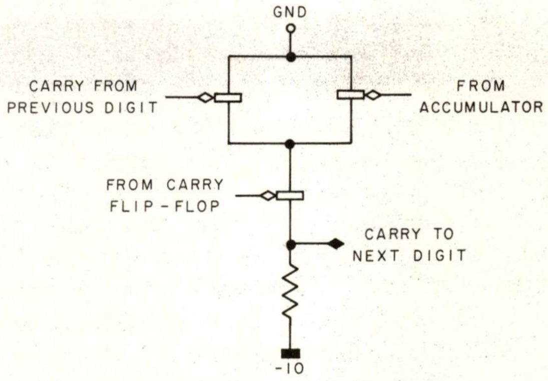


Fig. 40. TX-2 Carry Circuits

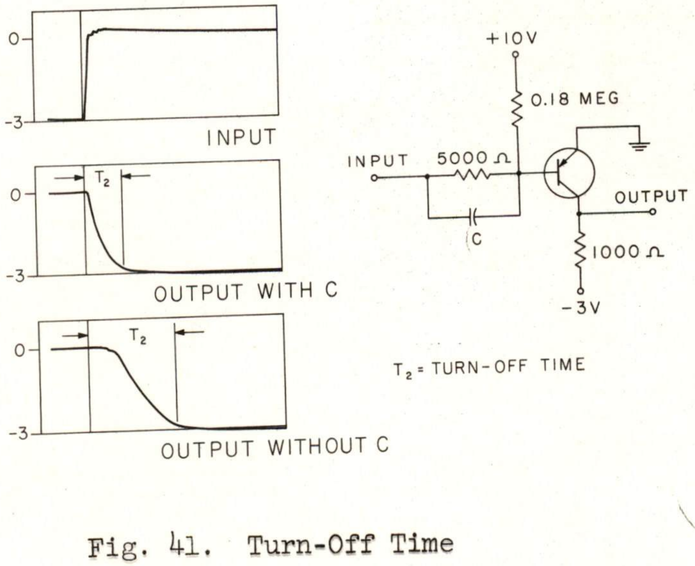


Fig. 41. Turn-Off Time

We run the circuits under saturated conditions to achieve stability and a wide tolerance to parameters without the need for clamp diodes. Unlike vacuum tubes, which always need an appreciable voltage across them for operation, a transistor requires practically no voltage across it. In spite of the delay in turning off saturated transistors, these circuits are faster than most vacuum-tube circuits. Faster circuit speed is not due to the fact that the transistors are faster than vacuum tubes, but because they operate at much lower voltage levels. A vacuum tube takes a signal of several volts to turn it from fully "on" to fully "off"; a transistor takes less than one volt.

B. Flip-Flop

On the basis of previous experience, we decided that the advantages of having one standard flip-flop were worth some complication in TX-2 circuitry. The circuit diagram of the flip-flop package in Fig. 42 is basically an Eccles-Jordan trigger circuit with a three-transistor amplifier on each output. The input amplifiers isolate the pulse input circuits and give high input impedance. The amplifiers give enough delay to allow the flip-flop to be set at the same time that it is being sensed. Fig. 43 shows the waveforms of this flip-flop package when complemented at a 10-megapulse rate. The rise and fall times, about 25 mμsec, are faster than one normally sees in a single inverter that pulls to ground and an emitter follower that pulls to -3 volts. Fig. 44 is a plot of the pulse amplitude necessary to complement the flip-flop at various frequencies. Note the independence of trigger sensitivity to pulse repetition rate. This circuit will operate at a 10-megapulse rate, twice the maximum rate at which it will be used in TX-2.

The TX-2 circuits reproduced most often were designed with a minimum number of components to achieve economies in manufacture

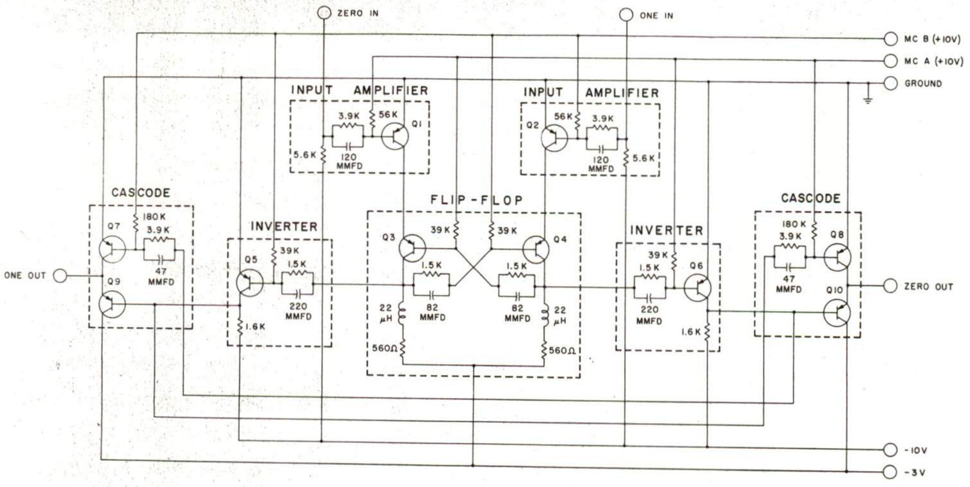


Fig. 42. TX-2 Flip-Flop

and maintenance. The design of less frequently reproduced cir-

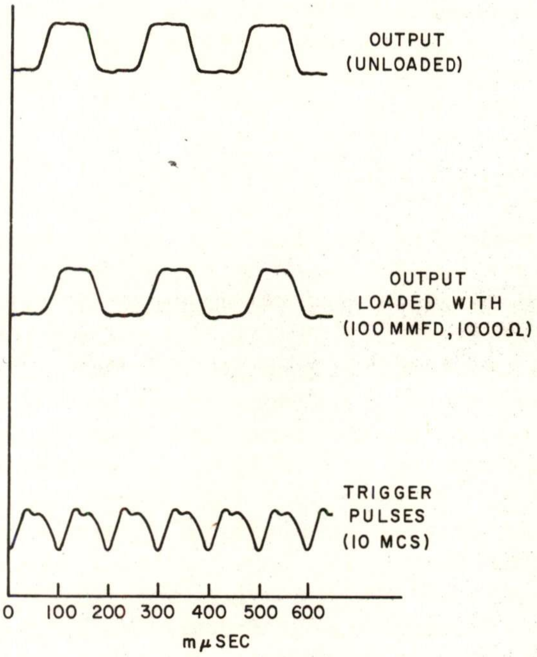


Fig. 43. Flip-Flop Waveforms

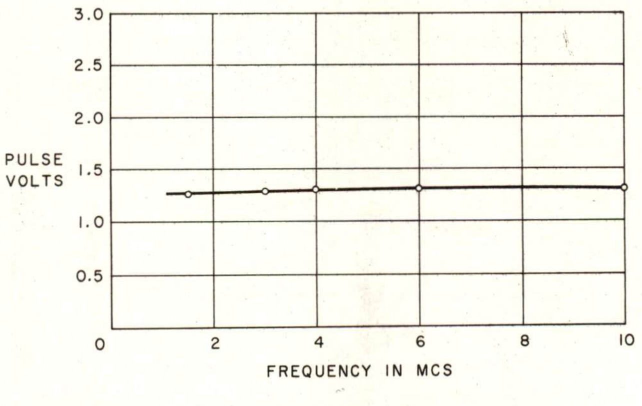


Fig. 44. Trigger Sensitivity

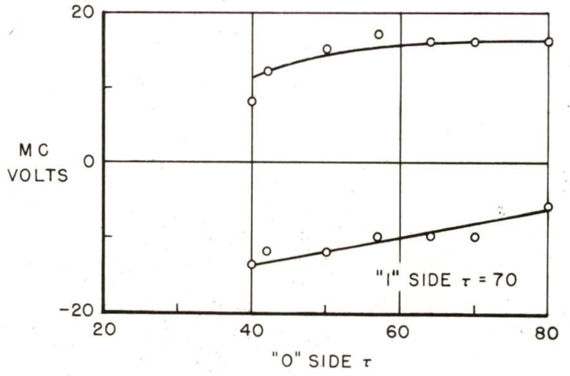


Fig. 45. τ Margins

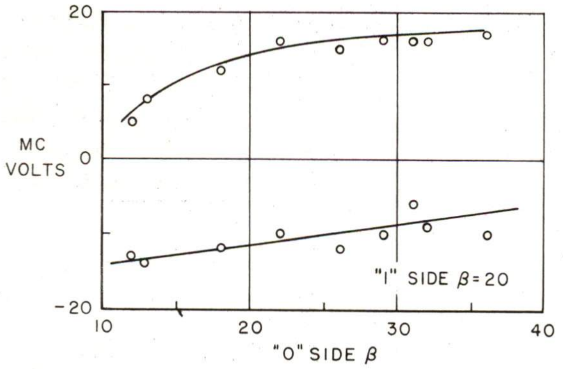


Fig. 46. β Margins

cuits made liberal use of components - even redundancy - to achieve long life and broad tolerance to component variations. The goal was system simplicity and high performance with a lower total number of components than might otherwise be possible. For example, the number of flip-flops in the TX-2 is small compared to the gates which transfer information from one group of flip-flops to another; so the flip-flops were allowed to be relatively complicated. But the TX-2 transfer gates were made very simple; a transfer gate is, in fact, only a single inverter. The emitter is connected to the output of the flip-flop being read and the collector is connected to the input of the flip-flop being set. The output impedance of the flip-flop is so low that, when the output is at the ground level, a pulse on the base of the transfer gate shorts the input of the other flip-flop to ground and sets its condition.

C. Marginal Checking

We planned to incorporate marginal checking in the design of these circuits so that, under a process of regularly scheduled maintenance, deteriorating components could be located before they caused failure in the system. We also found it practical to use this technique, during the design of the circuits, to locate the design center of the various parameters and to indicate the tolerance of circuit performance of these parameters. A further application of marginal checking has been found in other systems during shakedown and initial operation to pin-point noise and other system faults not serious enough to cause failure and, therefore, very difficult to isolate by other means.

The operating condition of the inverters is indicated by varying the +10-volt bias. In the flip-flop schematic in Fig. 42, the inverters were divided into two groups for marginal checking, and the two leads, labeled MCA and MCB, were varied one at a time for the most critical checking of the circuit. The curves in Figs. 43 and 44 show the locus of failure points for various parameters as a function of the marginal checking voltage. Fig. 45 shows the tolerance to tau, a measure of hole storage, and Fig. 46 shows the tolerance to β, the current gain. Operating margins for supply voltages, temperature, and pulse amplitude are shown in Figs. 47 through 50.

Packaging

The number of types of plug-in units was kept small for ease of production and to keep the number of spares to a minimum. The circuits are built on dip-soldered, etched boards and the components are hand-soldered to solid turret lugs. The boards are mounted in steel shells shown in Fig. 51 to keep the boards from flexing. The male and female contacts are machined and gold-plated. The sockets are hand-wired and soldered in panels as in Fig. 52.

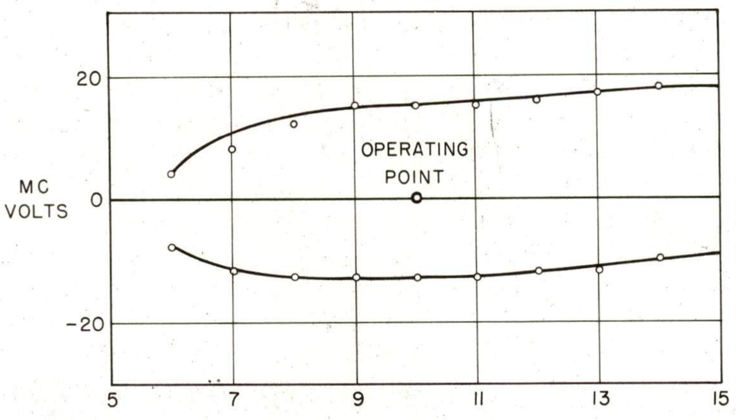


Fig. 47. -10 Volt Supply Margins

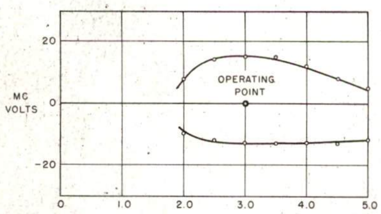


Fig. 48. -3 Volt Supply Margins

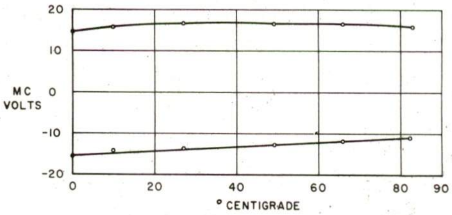


Fig. 49. Temperature Margins

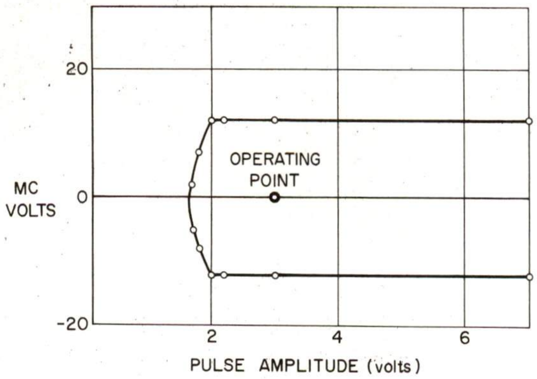


Fig. 50. Pulse Margins

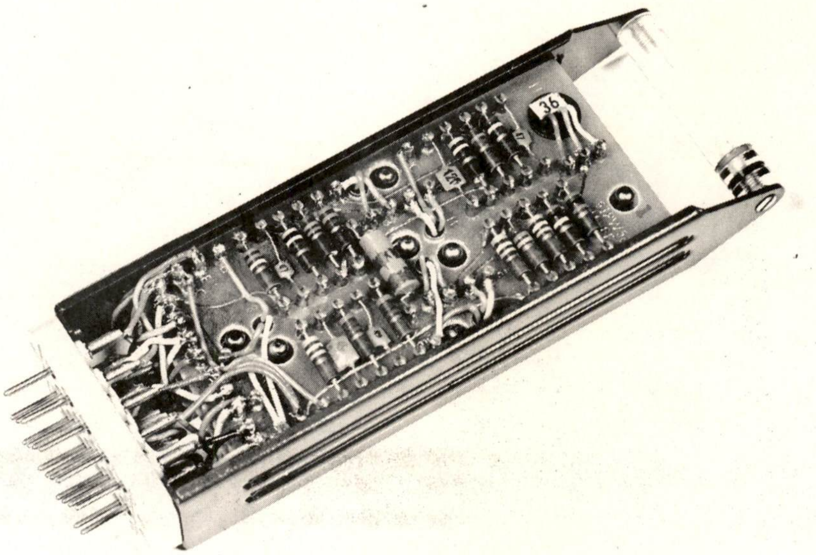


Fig. 51. TX-2 Plug-in Unit

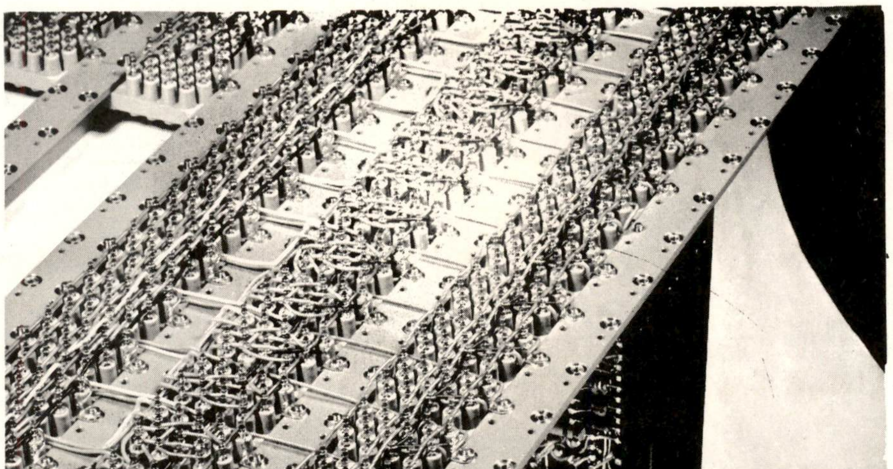


Fig. 52. TX-2 Back Panel

Conclusion

The result of these design considerations is a 5-megapulse control and arithmetic element which will take less than 40 square feet of space and dissipate less than 800 watts of power. The simplicity of the circuits has encouraged a degree of logic sophistication which would not have been chanced before.

Memorandum 6M-5097

Sheet 1 of 22 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: TX-0 Direct Input Utility System

To: Distribution List

From: John T. Gilmore, Jr.

Date: April 10, 1957

Approved: ____________________
Wesley A. Clark

Abstract: High speed memories have arrived. Many of our present procedures were born of necessity because of limited storage. It is time to review our present techniques and philosophies in planning and programming computer applications. At present we write programs which perform complicated and monotonous tasks efficiently and rapidly. But writing and debugging these programs are also complicated and monotonous tasks. Now that we have larger memories, why not try writing programs which will help us with our dirty work? This memo describes a utility system which is basic in form but unique in the sense that it assists the programmer in debugging and modifying his program at the console. This is accomplished by moving the utility system into memory alongside the program to be debugged and providing direct communication between the utility system and the programmer.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

TABLE OF CONTENTS

Distribution List:

Group 63 Staff
Bagley; P. R.
Heart, F.
Mayer, R.
Dinneen, G. P.
Ziegler, H.
Vanderburgh, A.
Arden, Dean (Barta)
Attridge, W.
Hazel, F. P.
Hosier, W.
Grandy, C.
Buzzard, R.
Holmes, L.
Bennington, H. (Rand)
Margaret Clark (Rand)
Burrows, J. H.
Collins, L. B.
Goldberg, S.
Jones, N.T.
Marnie, G. L.

Daggett, N.
Arnow, J.
Zraket, C. A.
Israel, D.
Rich, E.
Bailey, D.
Rising, H. K.
Frachtman, H. E.
Vance, R. R.
Neeb, Donna (Rand)
Thomas, L. M.
Dustin, D. E.
Tritter, A. L.
Briscoe, H.
Carma Darley
Joseph Thompson (Rand)
Paddock, R. B.
Pughe, E. W.
Rowe, A.
Zraket, C. A.

INTRODUCTION

TX-0 is an experimental computer which was built to test transistor circuitry and a 256x256 magnetic core memory. Its terminal equipment consists of a photo-electric paper tape reader, a paper tape punch, an oscilloscope, and an input-output Flexowriter typewriter. Additional equipment such as magnetic tape, high speed printers, etc., will be added at a later date. Since it is a test computer, most of the computer time is devoted to marginal checking and the operation of test programs. However, the fact that such a large internal memory of 65,536 registers now exists prompted us to try some new programming techniques in the time available between test operations.

One technique was to use the bulk of the memory as a secondary storage medium instead of magnetic drums or magnetic tapes. Wes Clark wrote a conversion program using this idea. The result was a fast conversion process and one which could translate a flexible symbolic language. For example, address tags, address sections of instructions, and constants can now be expressed as English words as well as code letters and numerals. The conversion program has no tape scanning problems and can look at the whole program to be converted as many times as necessary in a very short period of time. A memo describing the rules of the conversion program's vacabulary is being written by Wes and should be available soon.

Another valuable technique which will be described in this memo is that of moving a complete utility system into memory (see Figure 1) as well as the program (or programs) to be operated and debugged. Using the input-output Flexowriter typewriter as a means of communication with the utility system, debugging at the console should be reasonably simple and convenient.

Debugging at the console is not new. In fact, it is the oldest form of debugging. For the past few years it has been regarded as a great professional sin, but like most sins, it still exists, is rarely talked about, and hardly ever admitted. High costs and inefficiency are the main reasons why it is condemned. At the present time most computers rent for approximately three hundred dollars per hour. The time saved by debugging at the console of the computer is usually far less valuable than the cost of the additional computer time. The second reason is that at present, it is extremely difficult to get an objective view of what a program is doing or has done while at the console. An attempt to find out usually requires manual switch settings, limited examinations by push buttons, or trace program data which is not immediately available. A parallel argument is that thinking under pressure promotes poor reasoning and lots of bad guesses. The general result is wasted time----debugging time as well as precious computer time.

In the face of these arguments why try to encourage or develop techniques for debugging at the console? For the reason that there are certain situations and applications for which the length of time between the instant that the first instruction is written and the moment that the results are produced, is of the utmost importance. In such cases any time that can be saved by using more computer time to debug at the console is more than worth the additional rental cost. Data Processing, Operational Research, and Real Time problems are rapidly producing more and more situations like this. Although these critical situations will involve only a small percentage of the programs being operated on a given. computer, utility systems must be available to handle them.

There is still the problem of the programmer thinking on his feet, trying to find out what went wrong with his program and how to modify it. It is this phase in which we are extremely interested.

At the present time most utility programs do not provide enough immediate information to enable a programmer to find his error while he is still at the console. And if he can find it, few systems are capable of allowing him to make an immediate change without the preparation of a card, paper tape, or some other secondary input medium, especially if he wants to retain the symbolic language of the conversion program. Another handicap is that he cannot examine any given part of his program without having previously prepared a request card or tape. And finally, the use of trace programs is desirable only to the extent that they are simple enough to invite their use by the programmer. A tracing program that can give a visual picture of what a program is doing is of much more value than one which produces reams of printed information.

II DESCRIPTIONS OF INDIVIDUAL ROUTINES

Our first and most important reason for writing a direct utility system was to determine what routines would be most useful in assisting a programmer at the console. We therefore began with a minimum of specifications. The programming language was restricted to absolute addressed instructions and constants. This does not prevent a TX-0 programmer from writing his original program in relative form. The conversion program written by Wes Clark has a very flexible language and permits one to obtain an absolute addressed translation of a program which has been written in relative form. Later, when we have determined what the ideal direct utility system should contain we will include a more flexible programming language.

At the present time the system contains over a dozen routines. Some of them are very necessary and others provide services whose only imporance lies in the fact that they reduce impatience on the part of the programmer. Each routine obtains its required information by asking specific English questions on the console typewriter. The answers are typed on the same typewriter by the programmer and may be English words or numbers depending on the nature of the question asked.

Expedite

The system was designed around a master dispatch routine called EXPEDITE. Its job is to read the Flexowriter coded symbols coming from the typewriter and recognize English words. There is a small English vocabulary in the system and each stored word has an instruction associated with it. This instruction tells EXPEDITE what to do once an incoming word has been recognized. Most of the words are the names of the other routines in the system and the associated instructions inform EXPEDITE where the routines start. EXPEDITE not only acts as a dispatcher but is also used by the other routines as a subroutine when they require English answers from the console. Whenever a word is received which is not listed in the vocabulary, EXPEDITE informs the typist of the error. Also, should EXPEDITE receive its own name it will print out the words in its vocabulary.

Hark

The conversion program of the system is called HARK. Using HARK the programmer is able to call for specific registers for examination and, if necessary, modification. When an examination is requested, HARK translates the binary contents of the requested register into an octal addressed instruction. If the register contains the binary form of an operate-class command for which there is a three letter mnemonic code, HARK will type out the three letters which represent that operate-class command. When a modification is desired, the programmer may type the change as an octal instruction, a coded operate-class command, an octal number, or any combination of these three as long as they are separated by plus or minus signs. (Part IV provides explicit typing instructions.) Certain control keys of the typewriter allow a programmer to examine and modify a series of registers very rapidly. When it is desirable, HARK will provide a binary punched record of the modifications (in Read-In Mode format). HARK is also capable of recognizing a request to transfer control to any register in the program being debugged. When the programmer is through with HARK, the Stop Code button is pressed and EXPEDITE is ready to receive the name of another requested routine.

Prince

Every utility system must have a storage print-out routine. Ours is called PRINCE. It requires three pieces of information from the programmer. They are:

1. The kind of page layout.
2. How the binary contents of storage are to be translated.
3. What range of storage is to be translated and printed.

To determine the kind of page layout PRINCE types, DO YOU WANT VERTICAL COLUMN LAYOUT. The answer "YES" informs PRINCE that each printed page is to contain two columns of words. Each column will be 100 (octal) words long and the address sequence will run down the left column and then down the right. PRINCE will also lay out each page so that the address of the first word position in the left column will have a factor of 100 (octal). If the first word to be printed does not have an address with a factor of 100, PRINCE will print the first word in the left column relative to its position from the last address with a factor of 100. For example, if the address of the first word to be printed is 507, PRINCE will skip the first seven positions in the left column before printing the contents of register 507. In this way each page is laid out in a uniform manner. If "NO" is the answer to the first question, PRINCE will lay out 10 (octal) words to a line and 2^9 words per page. Although the vertical column layout printing time is longer than the horizontal, it is more popular because it has room for comments and is laid out in the way in which most programs are written. Since PRINCE has only two kinds of page layout the answers YES or NO are sufficient. If more layouts had been available the question would have been reworded and the answer would have required specific English word(s); e.g., DOUBLE VERTICAL, SINGLE VERTICAL, HORIZONTAL, etc.

DO YOU WANT OCTAL INSTRUCTIONS is the next question. "YES", is obvious. "NO", will inform PRINCE that each register in the range is to be translated as a constant.

The range is determined by the questions, WHAT IS THE FIRST ADDRESS TO BE PRINTED, and WHAT IS THE LAST ADDRESS. An octal address is required by the programmer after each question has been typed. It is interesting to note that PRINCE uses EXPEDITE as a subroutine when it requires an English answer and HARK when it requires a numerical answer.

Following the reception of the last address to be printed, the range is printed out in the desired form. When the range has been printed PRINCE asks, IS THERE MORE TO BE PRINTED. "YES", returns control to the first question again. "NO", returns control to EXPEDITE.

Punchy

In order to allow the programmer to retain a corrected binary record of his program, we have included a storage punch-out routine called PUNCHY. The first question asked by PUNCHY is, DO YOU WANT A TITLE. "YES", will cause the question, WHAT IS THE TITLE. The programmer then types what he wants labeled on the beginning of his tape. For example, SAMPLE TEST PROGRAM 25 FEB 57. "NO", will bypass the title question and lead to the question that follows the title punch-out, namely, DO YOU WANT THE NORMAL INPUT ROUTINE LAYOUT. "YES", will remind PUNCHY to punch out the Input Routine on tape in Read-In Mode format before punching the requested range of storage in the normal Input Routine format. "NO", will inform PUNCHY that the range is to be punched out in Read-In Mode format. A FUNCTIONAL DESCRIPTION OF THE TX-0 COMPUTER (6M-4789), explains the difference between the two kinds of binary layout. The Read-In Mode format is better for short ranges. The next two questions ask for the first and last address of the range to be punched out. The range is then punched out and then the question, ARE THERE MORE BLOCKS TO BE PUNCHED OUT, is asked. "YES", returns control to the questions asking for range addresses. "NO", causes the question, WHAT IS THE ADDRESS OF THE STARTING INSTRUCTION, to be asked. After an address has been typed by the programmer, the question, DO YOU WANT THE COMPUTER TO STOP AFTER READING IN THIS PUNCHED PROGRAM, is typed. "YES", informs PUNCHY to punch out the starting instruction in a form which will cause the Read-In Mode or the Input Routine to stop the computer before transferring control to the indicated address. "NO", will cause the opposite. Once the YES or NO has been received by PUNCHY, the starting instruction is punched out in the desired form and control is returned to EXPEDITE.

R And Me

The system contains a read-in routine similar to the Input Routine described in 6M-4789. When EXPEDITE receives an R followed by a carriage return, control is transferred to the input routine. Once R has read in a tape, the routine will print either GO TO or SUM ERROR, depending on whether or not the tape was read in correctly. In either case control will be transferred back to EXPEDITE so that a programmer can call for any one of the various routines in the system before operating his program. When he is ready to operate his program, the word ME will tell EXPEDITE to transfer control to the starting instruction which was on the end of the last tape read in by R.

Surprise

The SURPRISE routine is a useful one which reviews all the registers of a given program and types out the addresses of those registers whose contents have changed from their original form. It is called SURPRISE because the information it produces usually comes as a big surprise to the programmer.

After a program has operated and erred, the binary tape is again placed in the photoelectric reader. EXPEDITE is given control and the word SURPRISE is typed by the programmer. The tape is then read in by SURPRISE and the address of each register whose contents have changed is typed out along with the present contents and the original contents (translated as octal instructions or coded operate class commands). The original contents are restored in each case and when the tape review has been completed, control is transferred back to EXPEDITE.

Find

FIND is a routine which was born when we started to combine several programs in storage. We found that redundancy and confusion resulted when two or more programs were formed in memory. Areas which appeared to be empty were being used by other programs for temporary storage and programs were killing each other by store instructions or just strolling through certain crucial routines. We wrote FIND to help us find and avoid these difficulties. Its usefulness to a programmer probably depends on the length of his program and the number of small routines used to build it. FIND begins by typing the question, DO YOU WANT TO FIND A WORD, REFERENCES TO AN ADDRESS, OR SEARCH FOR AN OUTLAW TRN INSTRUCTION. The three possible answers are WORD, ADDRESS, and OUTLAW.

Word - When WORD is typed, FIND returns with, WHAT IS THE WORD. The answer may be an octal instruction, a coded operate-class command, an octal number, or any combination of the three as long as they are separated by plus or minus signs. FIND then searches the whole of memory (20 seconds) and types out the addresses of those registers which contain that word. When all of memory has been examined, FIND returns control to EXPEDITE. If nothing is found, FIND will print NO ADDRESS before returning control to EXPEDITE.

Address - When ADDRESS is typed by the programmer, FIND returns with, WHAT IS THE ADDRESS. The answer must be an octal number. FIND searches the whole of memory and types out the addresses and contents of those registers whose address sections agree with the address specified by the programmer. When all of memory has been examined, FIND will return control to EXPEDITE.

Outlaw- Have you ever had a program stop in some section of memory that was not even part of your program---or worse still, found it sitting still in a block of registers which contained constants instead of instructions? How it got there is a good question, and usually a hard one to answer, since it probably did not come to a complete stop as soon as it left your program. When OUTLAW is typed by the programmer, FIND returns with, WHERE DID YOUR PROGRAM STOP. An octal number is required. As soon as FIND receives the address it examines that register and those that precede it. It doubles back through storage until it finds an instruction or set of instructions which would cause an unconditional control transfer. The address given it by the programmer and the address following the unconditional transfer constitute an area which cannot be penetrated by control except by the use of a transfer control instruction. FIND then searches the whole of memory and examines each transfer instruction's address section. If the address section's value falls within the area in question, the address of the register and the transfer instruction will be printed out. When all of memory has been examined, FIND returns control to EXPEDITE. If nothing is found, NO ADDRESS will be printed before returning control to EXPEDITE.

Flow Chart Display

We mentioned earlier that we felt it was desirable to have a trace program which was able to give a programmer a visual picture of what his program was doing while he was at the console. The system has a routine which does this in a limited way. It is rather primitive but it may give an idea on how to accomplish the desired result. We call it The Flow Chart Display Subroutine. It performs two functions, the first of which is to display the four sides of a given box in a given area of the display scope. The second function is to keep a record of the sequence in which requests were made for all boxes. When the subroutine is requested the main program must supply three pieces of information, namely:

1. The x,y coordinates of the lower left corner of the box to be displayed.
2. The width and height of the box.
3. The Flexowriter code for some letter or numeral which the subroutine can use to identify this specific box.

The purpose of the subroutine is to indicate the control path in a program to be debugged. This is accomplished by first dividing the program into sections and drawing a flow diagram in which each section is represented as a box. The diagram must be drawn on transparent paper superimposed on a TX-0 Octal Graph Chart (see figures 3 and 4). When the diagram is completed the programmer notes the coordinates and dimensions of each section's box by referring to the Octal Graph Chart. In writing his program, he will insert a transfer control instruction to the Flow Chart Display Subroutine followed by the necessary information before each section's list of instructions. When the program is ready to be operated, the transparent copy of the flow diagram is placed on the face of the oscilloscope. During the operation of the program, as each section is performed, its corresponding box on the flow diagram will be illuminated. In this way the programmer will be able to follow the control path through his program. The number of times a box is illuminated for one operation of the section it represents, can be controlled externally (by means of the toggle switch accumulator). This allows the programmer to control the speed of his picture or to shut it off completely.

Path

When the program has stopped, the programmer may obtain a printed record of the sequence in which the sections were performed. This is done by transferring control to EXPEDITE and typing the word PATH. The result is a printed sequence of alpha numeric symbols (81 per line). When the print-out is completed, FINISHED is typed by PATH, and control is returned to EXPEDITE.

As an example, the boxes on the flow chart of figure 4 were lettered a,b,c,d,e,f,g,h,l,r,m, and s. A printed sequence from PATH might appear as:

```
abcdrmgcdrmgcdrmgcldrmgcldrmgcdefgcdefs
finished
```

Be Brief

Once a programmer has become familiar with the questions of each routine there is no need to continue the lengthy wording of each question. When BE BRIEF is typed by the programmer EXPEDITE will reduce each question in every routine to one or two words. When it has completed this task and is ready to receive title requests again, it informs the programmer by typing YES SIR.

III SUBROUTINES

In writing the Direct Input Utility System we made an effort to incorporate as many subroutines as possible. This was done for two reasons. First, it makes the job of modifying the system much easier, and second, it privides a few subroutines that a programmer can use in his own program.

Each subroutine has been written so that it must be entered with a transfer control instruction in the accumulator. The address section of this instruction should contain the address of the register where control is to be returned when the subroutine has completed its task. Unfortunately, TX-0 is such a simple machine that it does not have the equivalent of a Whirlwind A Register. Therefore, it is necessary to program the return address when using subroutines. But since the computer and this system only exist to discover new potentials from large memories, we have no reason to complain. TX-2, the eventual home of our large memory, is a very efficient computer in every respect. The ideas developed on TX-0 will be that much more powerful on TX-2.

The following is a list of subroutines in the system which may be useful in other programs written for TX-0:

I. Storage Print-out

A. Vertical column layout

1. 2^6 words/column, 2 columns/page

2. initial conditions

a. 173560 = +0 = instructions
173560 = -0 = octal numbers

b. 173561 = first address of range to be printed

c. 173562 = last address of range

3. Starting address of subroutine = 173323

B. Horizontal layout

1. 2^3 words/line, 2^6 lines/page

2. initial conditions

a. same as for Vertical Column Layout

3. Starting address of subroutine = 172767

II. Computer Word Print-out

A. Instructions

1. initial conditions

a. 177341 = word to be translated and printed

2. Starting address of subroutine = 173140

B. Octal number (address)

1. initial zero suppression

2. Initial conditions

   a. 177337 = word to be translated and printed

3. Starting address of subroutine = 177370

C. Octal Number

1. All six digits are printed

2. Initial conditions

   a. 177337 = word to be translated and printed out

3. Starting address of subroutine = 177710

III. English Word Printer

A. See appendix for list of English words available and their corresponding addresses.

B. The addresses of the words to be printed must be stored in the registers immediately following the transfer control instruction to this routine. The return address must be the address of the register following the register containing the address of the last English word to be printed.

C. Starting address of the subroutine = 173700

D. Example: print WHAT DO YOU WANT and a carriage return

```
       WHAT = 174311
         DO = 174234
        YOU = 174237
       WANT = 174243
 CAR RETURN = 174231
```

```
Absolute Address Programming      Symbolic Programming

100| cla                          question,   cla
     add 200                                  add| tn+answer
     trn 173700                               trn English
     174311                                   w h a t
     174234                                   d o
     174237                                   y o u
     174243                                   w a n t
     174231                                   c a r   r e t u r n
110| cla                          answer,     cla
     etc                                      etc
     etc                                      etc

200| trn 110                      English = 173700
```

Note: The TX-0 Conversion Program written by Wes Clark knows the addresses of all the words in the vocabulary of the system. This is achieved by a Flexowriter dictionary tape. Each word on the tape has a space separating each letter so that should there be a redundancy between an English word to be processed by this subroutine, and one which the programmer is using as an address tag, the conversion program will not be confused; i.e., t e s t vs. test.

IV Computer Word Reader (HARK)

A. This subroutine will read one word terminated by a carriage return or tab.

B. The binary translation of the word will be in register 176737 when control is transferred back to the main program.

C. Starting address of the subroutine = 176333

V. Word Answer Reader (EXPEDITE)

A. This subroutine will read a YES or NO terminated by a carriage return or tab.

B. Register 173322 will contain a +0 for YES and a -0 for NO when control is returned to the main program.

C. Starting address of the subroutine = 173307

D. Other English answers must be programmed and their words added to the vocabulary of EXPEDITE.

VI. Storage Punch-out

A. Read-In Mode Format

1. Initial conditions

a. 172506 = first address of range to be punched out
b. 172510 = last address of range

2. Starting address of subroutine = 172041

B. Normal Input Routine Format

1. Initial conditions

a. 172504 = +0 = Input routine will be punched out first in Read-In Mode Format before range is punched out.
-0 = no Input Routine leader.

b. 172505 = first address of range to be punched out

c. 172507 = last address of range

2. Starting address of subroutine = 172100

C. Starting Instruction Punch-out

1. Initial conditions

a. 172511 = add (starting instruction address) if stop after read-in is desired.
= trn (starting instruction address) if no stop after read-in is desired.

2. Starting address of subroutine = 172203

D. Blank tape feed-out

1. This routine feeds out six inches of blank tape.

2. Starting address of subroutine = 172213

Word - When WORD is typed, FIND returns with, WHAT IS THE WORD. The answer may be an octal instruction, a coded operate-class command, an octal number, or any combination of the three as long as they are separated by plus or minus signs. FIND then searches the whole of memory (20 seconds) and types out the addresses of those registers which contain that word. When all of memory has been examined, FIND returns control to EXPEDITE. If nothing is found, FIND will print NO ADDRESS before returning control to EXPEDITE.

Address - When ADDRESS is typed by the programmer, FIND returns with, WHAT IS THE ADDRESS. The answer must be an octal number. FIND searches the whole of memory and types out the addresses and contents of those registers whose address sections agree with the address specified by the programmer. When all of memory has been examined, FIND will return control to EXPEDITE.

Outlaw- Have you ever had a program stop in some section of memory that was not even part of your program---or worse still, found it sitting still in a block of registers which contained constants instead of instructions? How it got there is a good question, and usually a hard one to answer, since it probably did not come to a complete stop as soon as it left your program. When OUTLAW is typed by the programmer, FIND returns with, WHERE DID YOUR PROGRAM STOP. An octal number is required. As soon as FIND receives the address it examines that register and those that precede it. It doubles back through storage until it finds an instruction or set of instructions which would cause an unconditional control transfer. The address given it by the programmer and the address following the unconditional transfer constitute an area which cannot be penetrated by control except by the use of a transfer control instruction. FIND then searches the whole of memory and examines each transfer instruction's address section. If the address section's value falls within the area in question, the address of the register and the transfer instruction will be printed out. When all of memory has been examined, FIND returns control to EXPEDITE. If nothing is found, NO ADDRESS will be printed before returning control to EXPEDITE.

Flow Chart Display

We mentioned earlier that we felt it was desirable to have a trace program which was able to give a programmer a visual picture of what his program was doing while he was at the console. The system has a routine which does this in a limited way. It is rather primitive but it may give an idea on how to accomplish the desired result. We call it The Flow Chart Display Subroutine. It performs two functions, the first of which is to display the four sides of a given box in a given area of the display scope. The second function is to keep a record of the sequence in which requests were made for all boxes. When the subroutine is requested the main program must supply three pieces of information, namely:

1. The x,y coordinates of the lower left corner of the box to be displayed.
2. The width and height of the box.
3. The Flexowriter code for some letter or numeral which the subroutine can use to identify this specific box.

The purpose of the subroutine is to indicate the control path in a program to be debugged. This is accomplished by first dividing the program into sections and drawing a flow diagram in which each section is represented as a box. The diagram must be drawn on transparent paper superimposed on a TX-0 Octal Graph Chart (see figures 3 and 4). When the diagram is completed the programmer notes the coordinates and dimensions of each section's box by referring to the Octal Graph Chart. In writing his program, he will insert a transfer control instruction to the Flow Chart Display Subroutine followed by the necessary information before each section's list of instructions. When the program is ready to be operated, the transparent copy of the flow diagram is placed on the face of the oscilloscope. During the operation of the program, as each section is performed, its corresponding box on the flow diagram will be illuminated. In this way the programmer will be able to follow the control path through his program. The number of times a box is illuminated for one operation of the section it represents, can be controlled externally (by means of the toggle switch accumulator). This allows the programmer to control the speed of his picture or to shut it off completely.

Path

When the program has stopped, the programmer may obtain a printed record of the sequence in which the sections were performed. This is done by transferring control to EXPEDITE and typing the word PATH. The result is a printed sequence of alpha numeric symbols (81 per line). When the print-out is completed, FINISHED is typed by PATH, and control is returned to EXPEDITE.

As an example, the boxes on the flow chart of figure 4 were lettered a,b,c,d,e,f,g,h,l,r,m, and s. A printed sequence from PATH might appear as:

```
abcdrmgcdrmgcdrmgcldrmgcldrmgcdefgcdefs
finished
```

Be Brief

Once a programmer has become familiar with the questions of each routine there is no need to continue the lengthy wording of each question. When BE BRIEF is typed by the programmer EXPEDITE will reduce each question in every routine to one or two words. When it has completed this task and is ready to receive title requests again, it informs the programmer by typing YES SIR.

III SUBROUTINES

In writing the Direct Input Utility System we made an effort to incorporate as many subroutines as possible. This was done for two reasons. First, it makes the job of modifying the system much easier, and second, it privides a few subroutines that a programmer can use in his own program.

Each subroutine has been written so that it must be entered with a transfer control instruction in the accumulator. The address section of this instruction should contain the address of the register where control is to be returned when the subroutine has completed its task. Unfortunately, TX-0 is such a simple machine that it does not have the equivalent of a Whirlwind A Register. Therefore, it is necessary to program the return address when using subroutines. But since the computer and this system only exist to discover new potentials from large memories, we have no reason to complain. TX-2, the eventual home of our large memory, is a very efficient computer in every respect. The ideas developed on TX-0 will be that much more powerful on TX-2.

The following is a list of subroutines in the system which may be useful in other programs written for TX-0:

I. Storage Print-out
    A. Vertical column layout
        1. 2^6 words/column, 2 columns/page
        2. initial conditions
            a. 173560 = +0 = instructions
               173560 = -0 = octal numbers
            b. 173561 = first address of range to be printed
            c. 173562 = last address of range
        3. Starting address of subroutine = 173323
    B. Horizontal layout
        1. 2^3 words/line, 2^6 lines/page
        2. initial conditions
            a. same as for Vertical Column Layout
        3. Starting address of subroutine = 172767

II. Computer Word Print-out
    A. Instructions
        1. initial conditions
            a. 177341 = word to be translated and printed
        2. Starting address of subroutine = 173140
    B. Octal number (address)
        1. initial zero suppression

2. Initial conditions

   a. 177337 = word to be translated and printed

3. Starting address of subroutine = 177370

C. Octal Number

1. All six digits are printed

2. Initial conditions

   a. 177337 = word to be translated and printed out

3. Starting address of subroutine = 177710

III. English Word Printer

A. See appendix for list of English words available and their corresponding addresses.

B. The addresses of the words to be printed must be stored in the registers immediately following the transfer control instruction to this routine. The return address must be the address of the register following the register containing the address of the last English word to be printed.

C. Starting address of the subroutine = 173700

D. Example: print WHAT DO YOU WANT and a carriage return

```
      WHAT = 174311
        DO = 174234
       YOU = 174237
      WANT = 174243
CAR RETURN = 174231
```

```
Absolute Address Programming     Symbolic Programming

100| cla                         question,   cla
     add 200                                 add| tn+answer
     trn 173700                              trn English
     174311                                  w h a t
     174234                                  d o
     174237                                  y o u
     174243                                  w a n t
     174231                                  c a r   r e t u r n
110| cla                         answer,     cla
     etc                                     etc
     etc                                     etc

200| trn 110                     English = 173700
```

Note: The TX-0 Conversion Program written by Wes Clark knows the addresses of all the words in the vocabulary of the system. This is achieved by a Flexowriter dictionary tape. Each word on the tape has a space separating each letter so that should there be a redundancy between an English word to be processed by this subroutine, and one which the programmer is using as an address tag, the conversion program will not be confused; i.e., t e s t vs. test.

IV Computer Word Reader (HARK)

A. This subroutine will read one word terminated by a carriage return or tab.

B. The binary translation of the word will be in register 176737 when control is transferred back to the main program.

C. Starting address of the subroutine = 176333

V. Word Answer Reader (EXPEDITE)

A. This subroutine will read a YES or NO terminated by a carriage return or tab.

B. Register 173322 will contain a +0 for YES and a -0 for NO when control is returned to the main program.

C. Starting address of the subroutine = 173307

D. Other English answers must be programmed and their words added to the vocabulary of EXPEDITE.

VI. Storage Punch-out

A. Read-In Mode Format

1. Initial conditions

a. 172506 = first address of range to be punched out
b. 172510 = last address of range

2. Starting address of subroutine = 172041

B. Normal Input Routine Format

1. Initial conditions

a. 172504 = +0 = Input routine will be punched out first in Read-In Mode Format before range is punched out.
-0 = no Input Routine leader.

b. 172505 = first address of range to be punched out

c. 172507 = last address of range

2. Starting address of subroutine = 172100

C. Starting Instruction Punch-out

1. Initial conditions

a. 172511 = add (starting instruction address) if stop after read-in is desired.
= trn (starting instruction address) if no stop after read-in is desired.

2. Starting address of subroutine = 172203

D. Blank tape feed-out

1. This routine feeds out six inches of blank tape.

2. Starting address of subroutine = 172213

VII. Flow Chart Display and Sequence Record

A. Sequence reset subroutine

1. This routine is used when a new sequence is to be remembered. It is usually requested once at the beginning of a program.

2. Starting address of the subroutine = 171246

B. Flow Chart Display

1. This subroutine illuminates the boxes of a transparent copy of a flow diagram which is mounted on the face of the display scope. It also remembers the sequence in which the boxes were illuminated.

2. The three registers following each transfer control instruction to this subroutine must contain the following:

   a. First reg = octal values of the x,y coordinates of the lower left corner of the box to be displayed. e.g., if x = 10, and y = 300, then 1st reg. = 010300.

   b. Second reg = octal values of the width and height of the box to be displayed. e.g., if width = 70, and height = 50, then 2nd reg. = 070050.

   c. Third reg = flexowriter code for letter or numeral used to identify the box being displayed. The code is written as a six digit octal number. e.g., if letter = r, 3rd reg.= 010100.

3. The return address must be the address of the fourth register following the transfer control instruction to this subroutine.

4. Starting address of the subroutine = 171000

5. Example: Consider the program flow chart of figure 4. Let us assume that we are going to write the request which will display the box that represents that section which does the indexing and compares XL and XK. The request is written in the program just before the instructions which actually do the indexing and comparing. The coordinates of the lower left corner of the box are -40, -310. The width of the box is 100 and the height is 70. The Flexowriter code for g is 110100. The request in instruction form would be:

```
100    cla
       add 200
       trn 171000
       737467
       100070
       110100
       cla
       etc.        These instructions would
       etc.        contain the instructions
       etc.        for indexing the values
       etc.        and comparing XL and XK.
       etc.
       etc.
       etc.

200    trn 106
```

C. Sequence Print-out

1. This subroutine will print out the letters and numerals which represent the various sections of the program in the sequence that they were performed.

2. Using this routine, the sequence may be printed out at any time during the operation of the program. If it is not desirable to continue remembering the whole sequence then this routine should be followed by the Sequence Reset routine.

3. Starting address of the subroutine = 171257

PART IV Typing Directions

To use the Direct Input Utility System of TX-0, read in tape no. 18(latest mod). Be sure the TYPE IN switch is in the on position and start the program at register 175100. This will transfer control to EXPEDITE which is the title-read routine for the System. If you are not acquainted with the names of the other routines, type

expedite and a carriage return.

This will cause the computer to type out the list of the names.e.g.

```
expedite ↵

bebrief
word
address
find
hark
surprise
outlaw
punchy
prince
yes
no
expedite
me
r
path
```

If a name is typed incorrectly or is not in the above list, EXPEDITE will inform you of the fact. For example let us assume that you typed

be breif and a carriage return.

Spaces and upper and lower case are ignored but misspelling is not anticipated. Therefore EXPEDITE would have typed out the following:

```
            error bebreif
```

HARK

To transfer control to the system's conversion program, type

hark and a carriage return.

Case 1. TO EXAMINE A REGISTER:

Type the address of the register, a vertical bar and an equal sign. HARK will type out the contents of the register as an octal instruction, a tab, and then wait for a modification. Since no modification is to be made, the typist will type a carriage return and procede to examine some other register. e.g. examine registers 174 and 300.

$$\underbrace{174| =}_{\text{typist}}\ \underbrace{\text{add } 177 \overset{\text{TAB}}{\curvearrowright}}_{\text{HARK}}\qquad \underbrace{\overset{\text{C.R.}}{\hookleftarrow}}_{\text{typist}}$$

$$\underbrace{300| =}_{\text{typist}}\ \underbrace{\text{shr} \overset{\text{TAB}}{\curvearrowright}}_{\text{HARK}}\qquad \underbrace{\overset{\text{C.R}}{\hookleftarrow}}_{\text{typist}}$$

Case 2. EXAMINE AND MODIFY A REGISTER

Octal instructions, coded operate class commands, octal numbers, and any combination of these three separated by plus or minus signs may be used to express a modification. e.g. change register 174 to contain add 157.

$$\underbrace{174| =}_{\text{typist}}\ \underbrace{\text{add } 177 \overset{\text{TAB}}{\curvearrowright}}_{\text{HARK}}\qquad \underbrace{\text{add } 157 \overset{\text{C.R.}}{\hookleftarrow}}_{\text{typist}}$$

Case 3. MODIFY A REGISTER WITHOUT EXAMINING IT

$$\underbrace{174| \text{ add } 157, \hookleftarrow}_{\text{typist}}$$

Case 4. MODIFICATIONS OF A SERIES OF REGISTERS

174| add 157
sto 300
+234

In this case registers 174,175,and 176 would have been changed to add 157, sto 300, and +234. Each time a modification is typed and terminated the current address in HARK is indexed by one. Extra carriage returns and tabs are ignored.

Case 5. EXAMINATION AND MODIFICATION OF A SERIES OF REGISTERS

$$\underbrace{\text{174}| =}_{\text{typist}}\ \underbrace{\text{add 177} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{add 157} \hookleftarrow}_{\text{typist}}$$

$$\underbrace{\text{175}| =}_{\text{typist}}\ \underbrace{\text{sto 301} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{sto 300} \hookleftarrow}_{\text{typist}}$$

$$\underbrace{\text{176}| =}_{\text{typist}}\ \underbrace{\text{sto 230} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{+234} \hookleftarrow}_{\text{typist}}$$

Using the carriage return to terminate a modification you can see that it is necessary for the typist to type the next address and an equal sign in order to obtain the examination. This is not necessary if the modification is terminated by a period. e.g.

$$\underbrace{\text{174}| =}_{\text{typist}}\ \underbrace{\text{add 177} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{add 157.}}_{\text{typist}}\qquad \underbrace{\hookleftarrow}_{\text{HARK}}$$

$$\underbrace{\text{175}| = \text{sto 301} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{sto 300.}}_{\text{typist}}\qquad \underbrace{\hookleftarrow}_{\text{HARK}}$$

$$\underbrace{\text{176}| = \text{sto 230} \rightarrow}_{\text{HARK}}\qquad \underbrace{\text{+234.}}_{\text{typist}}\qquad \underbrace{\hookleftarrow}_{\text{HARK}}$$

$$\underbrace{\text{177}| = \text{sto 400} \rightarrow}_{\text{HARK}}\qquad \underbrace{\hookleftarrow}_{\text{typist}}$$

The period will cause HARK to store the modification, produce the carriage return, type the next address, the vertical bar, the contents of the register, a tab, and then wait for another modification from the typist.

Case 6. EXAMINATION OF A SERIES OF REGISTERS

The period is also helpful when a series of registers is to be examined without modifications. When HARK receives a period from the typewriter it checks to see whether a modification has been typed. If a modification has not been typed HARK will merely produce a carriage return, type the next address, a vertical bar, the contents of the register, and a tab. e.g.

```
      TYPIST
174| = add 157 →      .        ↵
175| = sto 300 →      .        ↵
176| = sto 234 →      ↵
     HARK          typist     HARK
```

Case 7. A SEARCHING EXAMPLE

Using the period it is quite simple to search a given set of registers and then modify the desired one. For example if an entry in a table was to be changed it would not be necessary to know its exact location. One would merely type the first address of the table and search for the entry by typing periods. Of course if the table was extremely long the PRINCE would be used instead. Let us assume that we want to search the table beginning at register 2000 and that we want to change the first register containing a minus zero to a one.

```
       TYPIST
2000| = sto 125       →      .       ↵
2001| = sto 100       →      .       ↵
2002| = sto 5         →      .       ↵
2003| = opr 177776    →      .       ↵
2004| = opr 177777    →     +1 ↵
HARK                      typist   HARK
```

Case 8. EXAMINATION, MODIFICATION, AND RECORDING

The comma performs the same function as the period but also enables the typist to obtain a punched binary tape record of his modifications. When a modification is terminated by a comma HARK stores the mod in the indicated register, punches the binary form of the current address on tape, punches the binary form of the modification on tape, produces a carriage return, types the next address, a vertical bar, an equal sign, the contents of the register, and a tab.

Case 8. continued.

Let us assume that we had to make modifications to registers 174,175,and 176 and that we also wanted a binary paper tape record as well.

```
   Typist
174| = add 177       add 157,
175| = sto 301       sto 300,
176| = sto 230       +234,
177| = sto 400

      HARK            typist        HARK
```

The paper tape would contain the following:

First three lines=	sto 174
next three lines=	add 157
next three lines=	sto 175
next three lines=	sto 300
next three lines=	sto 176
next three lines=	+234

This kind of layout is called the Read-In format and is discribed in 6m-4789, A FUNCTIONAL DISCRIPTION OF THE TX-0 COMPUTER .

Case 9. EXAMINATION AND RECORDING OF A SERIES OF REGISTERS

This would be obtained by typing a comma in place of a period.

Case 10. RECORDING THE ADDRESS OF THE STARTING INSTRUCTION ON PAPER TAPE

After a series of modifications are made it is desirable to record the address of the starting instruction. This enables the Read-In mode of TX-0 to transfer control to the program after the modifications have been read into storage from tape. To do this the typist merely types the address of the starting instruction,a vertical bar, and a comma. e.g.

174| ,

This will cause the next three lines on paper tape to contain add 174(the add signals the Read-In mode to stop the computer before transfering control to register 174) and then HARK will feed out some blank tape.

Case 11. TRANSFERING CONTROL FROM HARK TO SOME OTHER REGISTER IN MEMORY

To transfer control to some other register in memory the typist has to type the address followed by two vertical bars. e.g.

174||

Case 12. CONCEALING INFORMATION FROM HARK

We have found that the typewritten copies produced while using Hark have been very useful records. There is a feature in HARK which allows the typist to make comments alongside any of the examinations or modifications without HARK acting on the information. Whenever the vertical bar is the first piece of information that HARK receives after a terminating character, then all the information that is typed will be ignored until a carriage return is typed. e.g.

$$\underbrace{174| =}_{\text{TYPIST}}\ \underbrace{\text{add 177} \hookleftarrow}_{\text{HARK}}\quad \underbrace{\text{add 157}}_{\text{TYPIST}}\underbrace{\hookleftarrow |\ \text{add x is now add y}}_{\text{TYPIST}} \hookleftarrow$$

CASE 13. ERROR RECOGNITION BY HARK

Whenever an invalid word is typed Hark will inform the typist by typing the word "error" and the invalid word. Some examples of invalid words are:

1. Three letter combinations for which there are no operate class commands. i.e. clu,cpc,etc.
2. Any alphabetical letter used as a suffix to an octal number. i.e. 125j
3. The use of o and l for 0 and 1. i.e. 2oo and 175l0.
4. Two letter designations for the four operation symbols. i.e. ad, st, tn, op, etc.

When HARK does find an invalid word it will not disturb the current address when it prints out the error.

Case 14. NULLIFY CONTROL

There are many times when a typist will make a mistake and discover it before a terminating character has been typed. To nullify the whole word all that is required is the typing of a double x. This will erase the whole word or address being typed but will not disturb the current address. This feature is also provided by EXPEDITE.

Case 15. HOW TO TRANSFER CONTROL BACK TO EXPEDITE FROM HARK

To transfer control back to EXPEDITE the typist merely has to hit the STOP CODE button on the typewriter. This will cause EXPEDITE to guard the typewriter for incoming requests for other routines.

Case 16. SOME EXAMPLES OF LEGAL WORDS

We mentioned that a word may be an octal number, an operate class command in coded form, an octal instruction, or any combination of these three so long as they were separated by plus or minus signs. Here are some examples:

1. add 100 +5 -add -110 will be converted to -5.
2. add+add will become trn 0
3. +2-1+100 will become +101
4. -add-100 will become(the complement of add 100 or) trn 177677.
5. cla +200 will become opr 140200

Note. The addition of two coded operate class commands will not become a new combination including the functions of the two commands that were added together...ie. cll+clr ≠ cla.
cll+clr = trn 140001.

6. 125+40| will cause the current address to be 165.
7. 125+40|| will cause HARK to transfer control to register 165.

SUMMARY OF TYPING DIRECTIONS FOR HARK

```
Current Address--------------- address|

Examine Current Address------- =

Octal Instructions------------ sto   ( alone
                               add   ( or followed
                               opr   ) by an
                               trn   ( address

Octal Number------------------ six or less octal digits
                               alone, or preceded by a
                               plus or minus sign.

Coded Operate Class
Commands---------------------- cla  cll  clr  clc  cal
                               cyr  cyl  com  lac  alr
                               lpd  lro  lad  tac  tbr
                               shr  hlt  dis  ios  p7h
                               p6h  p6s  p7a  p6a  pnt
                               pna  pnc  rfa  rfl  rfr
                               r3c  r1c  r1l  r1r

Terminating Characters:
    1. Carriage return-------}will store modification in
    2. Tab-------------------)the register whose address
                               is the current address and
                               then index the current
                               address by one. If there is
                               no modification to be stored
                               HARK will ignore the tab or
                               carriage return and the
                               current address will remain
                               the same. Hark also ignores
                               extra carriage returns or
                               tabs.

    3. Period----------------- will store a modification
                               in the same way as the tab
                               and the carriage return do
                               but in addition it will
                               produce a carriage return,
                               print the next current
                               address, a vertical bar, an
                               equal sign, the contents of
                               the register whose address
                               is now the current address,
                               and a tab. If there is no
                               modification then none will
                               be stored. The rest of the
                               sequence will remain the same.
```

SUMMARY OF TYPING DIRECTIONS FOR HARK continued.

Terminating Characters cont.

4. Comma------------------ performs the same function as the period but in addition it will record the modification on paper tape in the format of the Read-In Mode. If no modification is made then it will record the original contents of the register whose address is the current address and then procede in same way as the period.

Additional Features

1. Recording the address of a starting instruction on paper tape---- address| ,

2. Transfering control to another section in memory-------------- address||

3. Concealing information from HARK------------↝| information ↩

4. Nullify control------- xx will nullify a whole word or address.

5. Transfering control back to EXPEDITE------ press the STOP CODE button.

Note: HARK will always type in the opposite color code of the typist.

IV IDEAS FOR NEW ROUTINES

As you can see, this utility system is very basic.

We have merely suggested a technique which will enable a programmer to see what his program is doing and examine and modify it rapidly while he is still at the console.

There are some new routines being written now which will be added to the system at a later date. When we have them written and operating we will write an addendum to this memo. The following is a list of some of the new routines and changes:

1. Several display subroutines
2. The flexo paper tape conversion program written by Wes Clark will be added to the system and will include the feature of informing the programmer of any illegal words on the flexo tape and asking him whether he wishes the conversion process to be continued with the illegal word (s) ignored or corrected. The correction would be added via the direct input flexo typewriter.
3. The direct input conversion routine, HARK, will have its vocabulary increased to understand relative and symbolic address tags.
4. In order to assure the programmer at the console that the utility system has not been damaged by his program, we plan on installing a check routine which will perform a sum check on all permanent instructions and constants in the system.
5. Any suggestions for routines or techniques which would increase the efficiency of debugging at the console will be appreciated.

John T. Gilmore, Jr.

JTG:bac

Attachments:

Appendix A
Appendix B
B-80354
D-80351

DL-1583
A-80287
A-68405

APPENDIX A

CODED OPERATE CLASS COMMANDS

CLA = opr 140000	Clear the accumulator
CLL = opr 100000	Clear the left nine bits of the accumulator
CLR = opr 40000	Clear the right nine bits of the accumulator
CLC = opr 140040	Clear the accumulator and complement it
CAL = opr 140200	Clear the accumulator and live register
CYR = opr 600	Cycle the accumulator one position to the right
CYL = opr 31	Cycle the accumulator one position to the left
COM = opr 40	Complement the accumulator
LAC = opr 140022	Transfer the contents of the live register to the accumulator
ALR = opr 201	Transfer the contents of the accumulator to the live register
LPD = opr 22	Partial add the contents of the accumulator and the live register and leave result in AC
LRO = opr 200	Clear the live register
LAD = opr 32	Add the contents of the live register and the accumulator and leave result in AC
TAC = opr 140004	Transfer the contents of the toggle switch accumlator to the accumulator
TBR = opr 140023	Transfer the contents of the toggle switch buffer register to the accumulator
SHR = opr 400	Shift the accumulator to the right one position (multiply by 2^{-1})
HLT = opr 30000	Stop the computer
DIS = opr 22000	Display a point on the face of the oscilloscope according to the value in the accumulator
IOS = opr 160000	In out stop and accumulator cleared
P7H = opr 27600	Punch seven holes on paper tape and cycle AC right one position (AC 2, 5, 8, 11, 14, 17 Tape 1 2 3 4 5 6)
P6H = opr 26600	Punch six holes on paper tape and cycle AC right
P6S = opr 166000	Clear AC and punch one line of blank tape
P7A = opr 27012	Punch seven holes and clear AC

CODED OPERATE CLASS COMMANDS (CONTINUED)

P6A = opr 26021	Punch six holes and clear AC
PNT = opr 24600	Print and cycle right (AC 2, 5, 8, 11, 14, 17 - Flexo 1, 2, 3, 4, 5, 6)
PNA = opr 24021	Print and leave AC cleared
PNC = opr 24061	Print and clear and complement AC
RFA = opr 141000	Clear AC and start petr running
RFL = opr 1031	Cycle AC left and start petr running
RFR = opr 1600	Cycle AC right and start petr running
R3C = opr 163000	Clear AC and read three lines from paper tape (Tape 1, 2, 3, 4, 5, 6 AC 0, 3, 6, 9, 12, 15)
R1C = opr 161000	Clear AC and read one line from paper tape
R1L = opr 161031	Clear AC, read one line and cycle left
R1R = opr 161600	Clear AC, read one line and cycle right

APPENDIX B

6M-5097

ENGLISH VOCABULARY
of the
TX-0 ENGLISH PRINTER SUBROUTINE

a	174170	outlaw	174444
address	174163	print	174212
after	174361	printed	174337
an	174413	program	174120
are	174100	punched	174113
area	174427	question	174455
be	174334	reading	174365
block	174104	reference	174416
car return (↲)	174231	routine	174271
column	174205	s	174160
comma	174410	search	174433
computer	174350	sir	174474
did	174466	starting	174142
do	174234	stop	174355
double	174173	tab (↷)	174110
find	174400	test	174131
finished	174135	the	174315
first	174321	there	174301
for	174440	this	174125
in	174372	title	174344
input	174265	to	174331
instruction	174147	trn	174451
is	174276	TX-0	174375
last	174325	vertical	174200
layout	174247	want	174243
more	174305	what	174311
no	174226	where	174462
normal	174260	word	174404
octal	174254	yes	174222
of	174155	you	174237
or	174424	your	174074
out	174216		

B-80354

	(4096)	(8192)	(12,288)	(16,384)	(20,480)	(24,576)	(28,672)	(32,768)	(36,864)	(40,960)	(45,056)	(49,152)	(53,248)	(57,344)	(61,440)
000	10,000	20,000	30,000	40,000	50,000	60,000	70,000	100,000	110,000	120,000	130,000	140,000	150,000	160,000	170,000
1,000	11,000	21,000	31,000	41,000	51,000	61,000	71,000	101,000	111,000	121,000	131,000	141,000	151,000	161,000	171,000
2,000	12,000	22,000	32,000	42,000	52,000	62,000	72,000	102,000	112,000	122,000	132,000	142,000	152,000	162,000	172,000
3,000	13,000	23,000	33,000	43,000	53,000	63,000	73,000	103,000	113,000	123,000	133,000	143,000	153,000	163,000	173,000
4,000	14,000	24,000	34,000	44,000	54,000	64,000	74,000	104,000	114,000	124,000	134,000	144,000	154,000	164,000	174,000
5,000	15,000	25,000	35,000	45,000	55,000	65,000	75,000	105,000	115,000	125,000	135,000	145,000	155,000	165,000	175,000
6,000	16,000	26,000	36,000	46,000	56,000	65,000	76,000	106,000	116,000	126,000	136,000	146,000	156,000	166,000	176,000
7,000	17,000	27,000	37,000	47,000	57,000	67,000	77,000	107,000	117,000	127,000	137,000	147,000	157,000	167,000	177,000 177,740 177,777
(4,095)	(8,191)	(12,287)	(16,383)	(20,479)	(24,575)	(28,671)	(32,767)	(38,863)	(40,959)	(45,055)	(49,151)	(53,247)	(57,343)	(61,437)	(65,535)

TX-0
DIRECT INPUT
UTILITY SYSTEM

TX-0
INPUT ROUTINE

TX-0 MEMORY REFERENCE CHART

FIG. 1

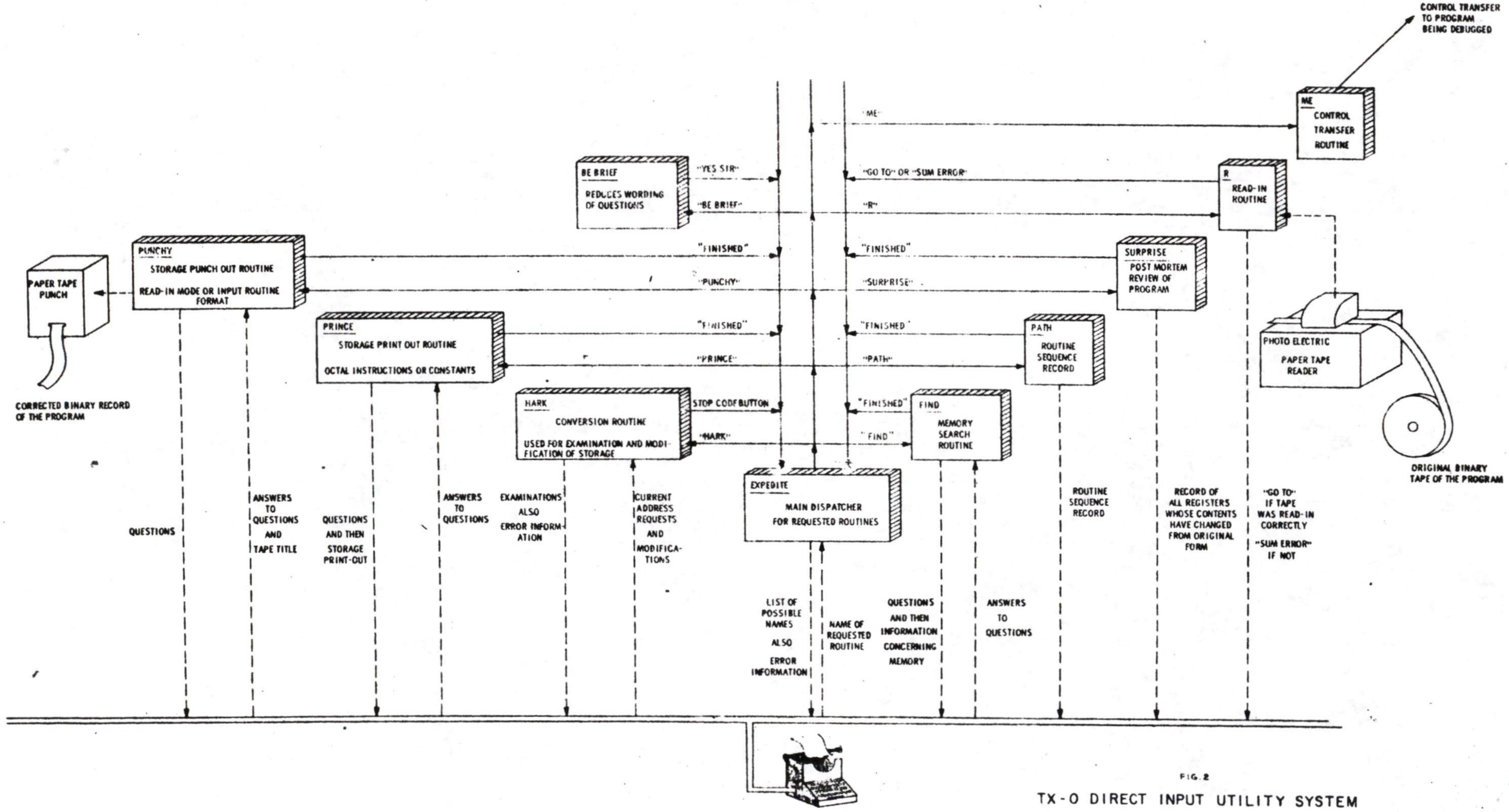
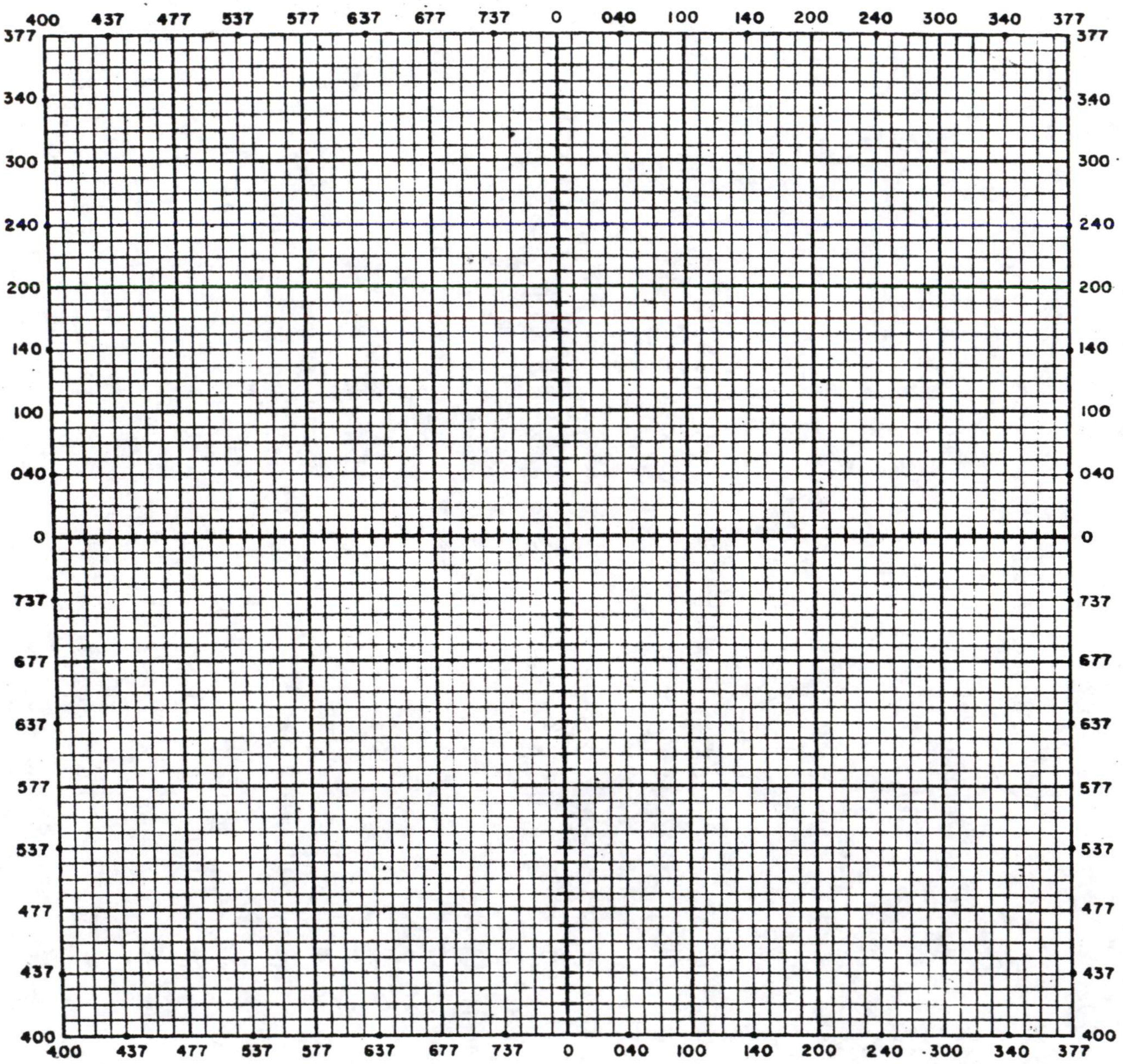

CONTROL TRANSFER TO PROGRAM BEING DEBUGGED
ME
CONTROL TRANSFER ROUTINE
"ME"
BE BRIEF
REDUCES WORDING OF QUESTIONS
"YES SIR"
"BE BRIEF"
"GO TO" OR "SUM ERROR"
"R"
R
READ-IN ROUTINE
PUNCHY
STORAGE PUNCH OUT ROUTINE
READ-IN MODE OR INPUT ROUTINE FORMAT
PAPER TAPE PUNCH
CORRECTED BINARY RECORD OF THE PROGRAM
"FINISHED"
"PUNCHY"
"FINISHED"
"SURPRISE"
SURPRISE
POST MORTEM REVIEW OF PROGRAM
PRINCE
STORAGE PRINT OUT ROUTINE
OCTAL INSTRUCTIONS OR CONSTANTS
"FINISHED"
"PRINCE"
"FINISHED"
"PATH"
PATH
ROUTINE SEQUENCE RECORD
PHOTO ELECTRIC
PAPER TAPE READER
ORIGINAL BINARY TAPE OF THE PROGRAM
HARK
CONVERSION ROUTINE
USED FOR EXAMINATION AND MODIFICATION OF STORAGE
STOP CODE BUTTON
"HARK"
"FINISHED"
"FIND"
FIND
MEMORY SEARCH ROUTINE
EXPEDITE
MAIN DISPATCHER FOR REQUESTED ROUTINES
QUESTIONS
ANSWERS TO QUESTIONS AND TAPE TITLE
QUESTIONS AND THEN STORAGE PRINT-OUT
ANSWERS TO QUESTIONS
EXAMINATIONS ALSO ERROR INFORMATION
CURRENT ADDRESS REQUESTS AND MODIFICATIONS
LIST OF POSSIBLE NAMES ALSO ERROR INFORMATION
NAME OF REQUESTED ROUTINE
QUESTIONS AND THEN INFORMATION CONCERNING MEMORY
ANSWERS TO QUESTIONS
ROUTINE SEQUENCE RECORD
RECORD OF ALL REGISTERS WHOSE CONTENTS HAVE CHANGED FROM ORIGINAL FORM
"GO TO" IF TAPE WAS READ-IN CORRECTLY
"SUM ERROR" IF NOT

FIG. 2

TX-0 DIRECT INPUT UTILITY SYSTEM



TX-O OCTAL GRAPH CHART

DL-1585

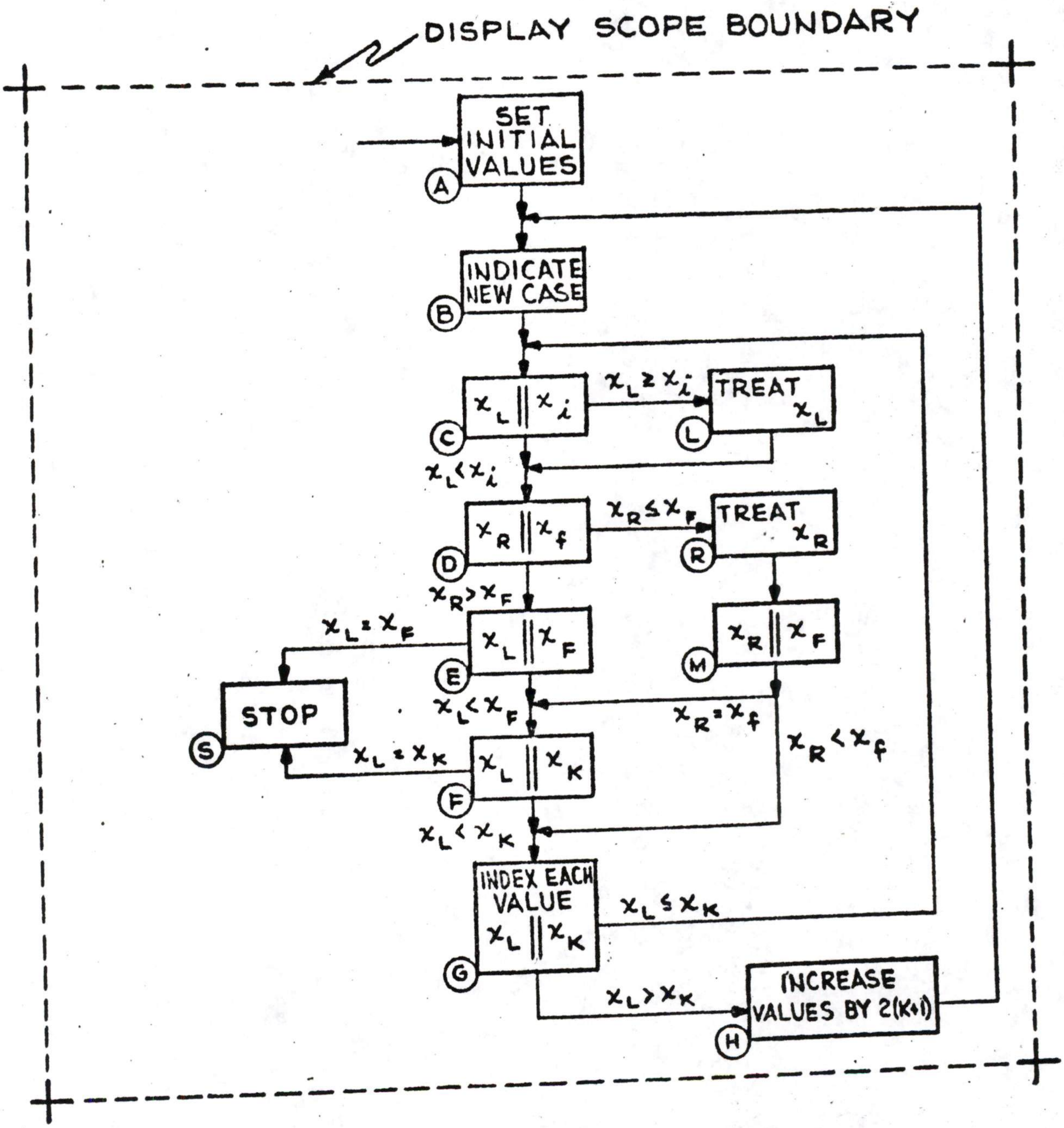
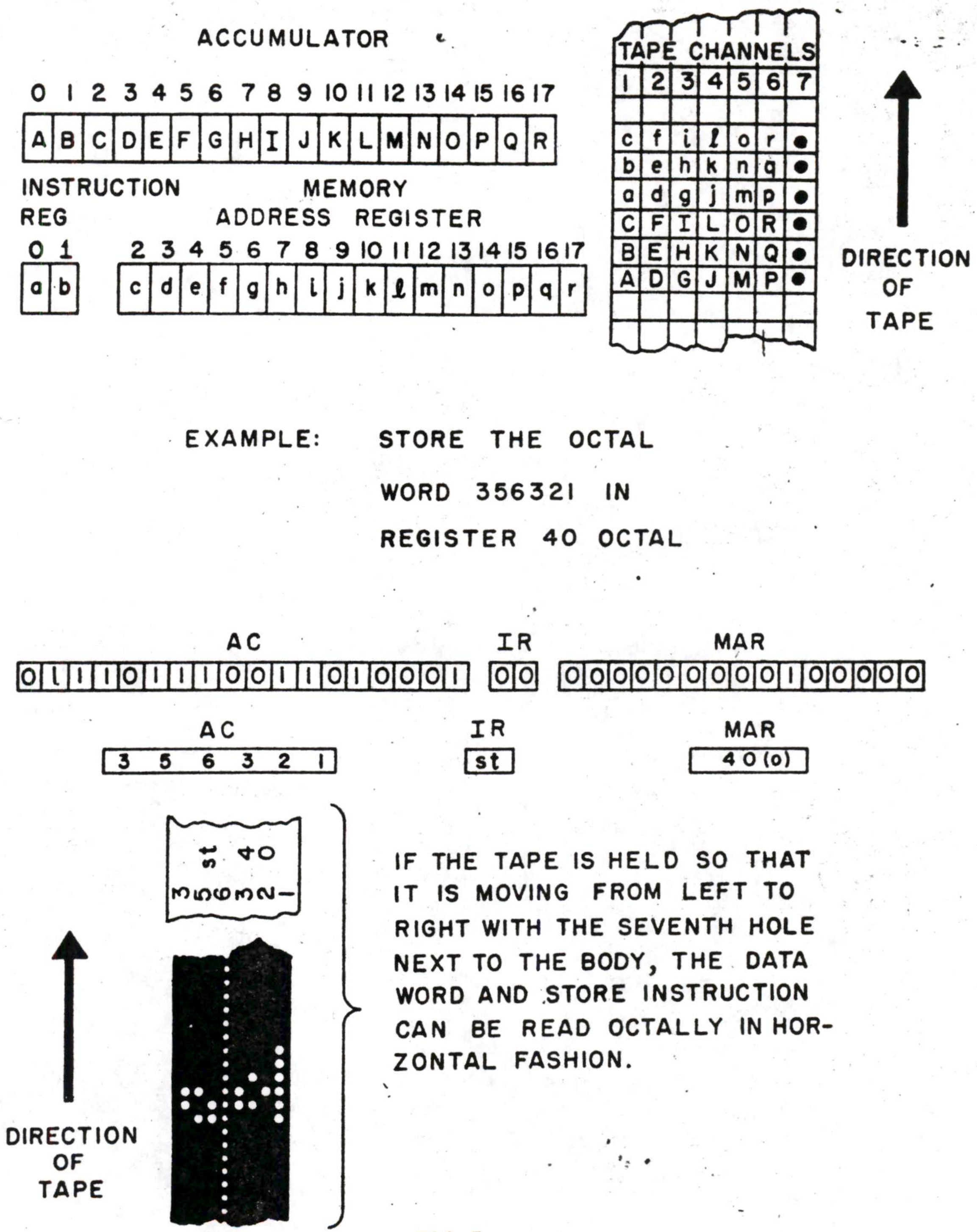


FIG. 4

FLOW CHART EXAMPLE

-80287



ACCUMULATOR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

INSTRUCTION REG

0	1
a	b

MEMORY ADDRESS REGISTER

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r

TAPE CHANNELS

1	2	3	4	5	6	7
c	f	i	ℓ	o	r	●
b	e	h	k	n	q	●
a	d	g	j	m	p	●
C	F	I	L	O	R	●
B	E	H	K	N	Q	●
A	D	G	J	M	P	●

EXAMPLE: STORE THE OCTAL WORD 356321 IN REGISTER 40 OCTAL

AC	IR	MAR
011101110011010001	00	000000000100000
3 5 6 3 2 1	st	40(o)

IF THE TAPE IS HELD SO THAT IT IS MOVING FROM LEFT TO RIGHT WITH THE SEVENTH HOLE NEXT TO THE BODY, THE DATA WORD AND STORE INSTRUCTION CAN BE READ OCTALLY IN HORZONTAL FASHION.

FIG. 5

TAPE LAYOUT FOR READ-IN MODE OF TX-0

A-68405

J. Fadiman
B063

Memorandum 6M-5780

Sheet 1 of 15 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: SOME EXAMPLES OF TX-2 PROGRAMMING

To: Distribution List

From: H. Philip Peterson H. Philip Peterson

Date: July 23, 1958

Approved: WAC

Abstract: Six short programs are presented here to illustrate many of the somewhat inscrutable features of TX-2 programming. These programs are called:

I: A Checkerboard Pattern Generator

II: The Inchworm

III: The Memory Mirror

IV: An Autocorrelation Program

V: The Flexo-Octal Converter

VI: A Binary Read-in Routine

Distribution List:

Group 63 Staff
Arden, Dean (Barta)
Arnow, J.
Attridge, W.
Bagley, P. R.
Bailey, D.
Briscoe, H.
Buzzard, R.
Daggett, N.
Dinneen, G. P.
Dustin, D. E.
Forgie, Carma
Vance, R. R.

Frachtman, H. E.
Frick, F.
Grandy, C.
Hazel, F. P.
Heart, F.
Holmes, L.
Israel, D.
Mason, Wm.
Pughe, E. (Servo Lab.)
Rising, H. K.
Thomas, L. M.
Tritter, A. L.
Zraket, C. A.

1. This document has been prepared for internal use only. It has not been reviewed by Office of Security Review, Department of Defense, and therefore, is not intended for public release. Further dissemination or reproduction in whole or in part of the material within this document shall not be made without the express written approval of Lincoln Laboratory (Publications Office).

2. The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No AF 19 (122)-458.

INTRODUCTORY REMARKS

The six example programs presented in this paper illustrate many of the somewhat inscrutable features of TX-2 programming. A few assumptions, however have been made by the author about the reader. These assumptions are:

1) that the reader knows how to program;
2) that the reader is familiar with TX-2 nomenclature: (this familiarity may be attained by studying "The Lincoln TX-2 Computer," (6M-4968); and
3) that the reader has a copy of "The TX-2 Programmer's Guide" for reference (6M-5807).

NOTATION

The code part of each instruction is written as a group of 3 capital letters (ADD, JMP, etc.). Any superscript numbers preceding the code part refer to a configuration memory location, except for JPX, JNX, JMP and SKM instructions. Superscript numbers following a code refer to an index memory location except for SKM type instructions where this is the number of the bit in the addressed word. Lower case numbers following a code are main memory addresses.

A colon means "hold control until the next instruction." Brackets mean "defer the address" and imply that bit 2.9 of the address is a ONE. Lower case letters are hopefully self-explanatory.

To the left of many instructions will be an explanatory notation using four little lines which show the permutation involved by how they cross; the active quarters of central machine registers by arrowheads, and, when necessary, the fracture (or coupling or subwords) by little cups. This configuration will be specified by the contents of the indicated configuration memory word.

A number or word followed by "slash equals" defines the address of the instruction or constant to the right of it. A word followed by "equals slash" is the name of the register following.

An address section with a large L prefixing it, as in 763 of Program I, means the address of "a register containing what is indicated."

All numbers in programs are octal unless otherwise indicated. Numbers are punctuated with commas separating the meaningful portions of the whole 36 bit word. A single comma separates 9 bit (3 octal digit) quarters when the word is dealt with in quarters. Two consecutive commas will separate the word into 18 bit (6 octal digit) pieces.

I. A Checkerboard Pattern Generator

The Problem

When a core memory is being checked for operating margins, a "bad" pattern of ONES and ZEROS is desired. (see Engineering Note E-488). One of the worst conditions starts with a checkerboard pattern which looks like this in each memory plane:

```
0 1 1 0 0 1 1 . . .
1 0 0 1 1 0 0 . . .
1 0 0 1 1 0 0 . . .
0 1 1 0 0 1 1 . . .
0 1 1 0 0 1 1 . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

The complement of this pattern is also a checkerboard. The addresses increase from left to right and top to bottom beginning with address 000 at the upper left. In the case of a 256^2 memory plane, it takes 8 bits to address a row or a column (16 address bits in all).

If one computes the parity of the two least significant bits of the row address and the two least significant bits of the column address, one will find that if the parity of these four bits is odd, a ONE will be at that address; if even, a ZERO will be there.

The problem is to construct a program which generates this pattern in all 65,536 bits of each memory plane. The program must fit into the 16 toggle switch registers.

The Solution

Program I generates the checkerboard pattern by using four SKZ instructions to look at the two sets of least-significant address bits. These bits are 1.1, 1.2, 1.9 and 2.1. When any one of them is a ONE, the SKZ doesn't skip and an MKC is executed which complements bit 3.1 of the E register. After examining the four address bits, E register bit 3.1 will be ZERO for an even parity or ONE for an odd parity. The whole address is kept in index register 1 and the DPX at 751* puts the address in the right half of the E register, leaving the left half all ZEROS since configuration 0 is used. After computing the parity, the left half of E is put in index register 2 and the LDE at 763* puts the word at 766 in E if the parity is even, or the word at 767 if it is odd. The word in E is stored away at the address and the address is counted down. The address was reset by the RSX in 750 to 177,777. This number is kept in the A register (377,740) which is being simulated by a toggle switch register as of this writing.

*The three most significant octal digits of addresses (377 in toggle switch addresses) will be omitted for brevity's sake.

OCTAL EQUIVALENT	ADDRESS		SYMBOLIC
02 11 01 377,740	377 750		^{2}RSX^{1} ⌊177,777
00 16 01 377,744	751		^{0}DPX^{1} e reg
10 17 41 377,744	752		SKZ$^{2.1}$ e reg
03 17 61 377,744	753		MKC$^{3.1}$ e reg
10 17 31 377,744	754		SKZ$^{1.9}$ e reg
03 17 61 377,744	755		MKC$^{3.1}$ e reg
10 17 22 377,744	756		SKZ$^{1.2}$ e reg
03 17 61 377,744	757		MKC$^{3.1}$ e reg
10 17 21 377,744	377 760		SKZ$^{1.1}$ e reg
03 17 61 377,744	761		MKC$^{3.1}$ e reg
02 11 02 377,744	762		^{2}RSX^{2} e reg
00 20 02 377,766	763		^{0}LDE^{2} ⌊word
00 30 01 000,000	764		^{0}STE^{1} memory
36 06 01 377,751	765		$^{-1}$JPX^{1} next
00 05 00 377,750	766	(word)	JMP restart
77 72 77 400 027	767		(- word)

Program I A Checkerboard Pattern Generator

OCTAL EQUIVALENT	ADDRESS	SYMBOLIC
02 11 01 377,740	377 750	^{2}RSX^{1} └177,777
00 16 01 377,744	751	↓ ↓ ↓ ↓ ^{0}DPX^{1} e reg
10 17 41 377,744	752	SKZ$^{2\cdot 1}$ e reg
03 17 61 377,744	753	MKC$^{3\cdot 1}$ e reg
10 17 31 377,744	754	SKZ$^{1\cdot 9}$ e reg
03 17 61 377,744	755	MKC$^{3\cdot 1}$ e reg
10 17 22 377,744	756	SKZ$^{1\cdot 2}$ e reg
03 17 61 377,744	757	MKC$^{3\cdot 1}$ e reg
10 17 21 377,744	377 760	SKZ$^{1\cdot 1}$ e reg
03 17 61 377,744	761	MKC$^{3\cdot 1}$ e reg
02 11 02 377,744	762	^{2}RSX^{2} e reg
00 20 02 377,766	763	↓ ↓ ↓ ↓ ^{0}LDE^{2} └word
00 30 01 000,000	764	↓ ↓ ↓ ↓ ^{0}STE^{1} memory
36 06 01 377,751	765	$^{-1}$JPX^{1} next
00 05 00 377,750	766	(word) JMP restart
77 72 77 400 027	767	(- word)

Program I A Checkerboard Pattern Generator

Note that the word stored away for even parity is the JMP instruction at 766 which restarts the process after the address has been counted down through 000. If one wishes to write the pattern just once, put ALL ZEROS in 766 and ALL ONES in 767. TX-2 will halt with an illegal instruction alarm (ICSAL) if it tries to execute an instruction with 00 as the operation code. Putting +0 in 766 and -0 in 767 has the advantage that the checkerboard patterns in each digit plane will be identical.

Exercises To Prove To Yourself That You Really Understand

By changing a single toggle switch, the checkerboard pattern will be complemented. Which switch? (Hint: any one or three of four will do.)

What would you do to put the pattern just in the lower addressed half of memory? Upper half? Middle quarter?

Two memory planes of the 38 will not have a checkerboard pattern in them. They are the parity bit plane and the meta bit (4.10) plane. What program changes will put the pattern in either plane? (Note: only an SKM can modify a meta bit.)

II. The Inchworm

The Problem

A classical programming exercise is to design a routine which will move itself along through memory, carrying with it as it goes all necessary constants for repeating this "inchworm" process. The program for starting the inchworm on its way must fit into TX-2's 16 toggle switch storage registers, naturally.

The Solution

Program II solves the problem by storing in registers 001 - 007 the program shown. This program in 001 - 007 then forms the one in 010 - 016 which duplicates itself in 017 - 025 and so on. The program in togs works like the ones in main memory except for a few special setting up instructions.

The SPF instruction in 752 specifies that configuration 34 will permute quarter 3 into quarter 1 and extend its sign into quarter 2. The RSX in 755, 1, 10, etc. uses this oddball configuration to reset index register 71 to a -6 from quarter 3 of registers 762, 6, 15, etc. This trick allows the inchworm program to avoid carrying constants per se along with it. Each "old" inchworm setment can simply "fall" into the newly formed one without jumping around some constants.

Index register 2 contains the constant necessary for the program to move itself into the next location. When moving from togs to core memory, this constant is 400 023 and the RSX in 753 fixes it up. When moving on in core memory, this constant is 000,007 and the RSX in 764 sets it up.

The RSX in 754 resets index register 3 to snap back to togs after the last address desired is reached. For illustrative purposes the address constant 577,760 was chosen. Since the JPX in 763, 7, 16 etc. jumps when the index register is positive, for our purposes it must be negative until the end is reached. Consequently 400,000 is added to 177,760 and that number (577,760) is set up in left half of 750. Each time an inchworm "segment" is executed, the corresponding JPX will subtract 7 from the contents of index register 3. When control gets to the segment in 177,757 - 766, index register 3 will have become positive and control will be transferred to togs (after a segment is written into 177,767 to 75) starting the process over again.

The routine in 755 to 763 maps itself into 000 to 007, preserving the address parts of the instructions in 757, 760 and 763 as they go to 003, 004, and 007 by the action of the SKN in 757 which skips over the ADX when bit 3.2 of a word is a ONE. These three invariant instructions refer to fixed locations so they must not be changed by the ADX as the other four are. Bit 3.2 was arranged to be ONE in the invariant instructions and ZERO in the variable instructions.

OCTAL EQUIVALENT	ADDRESS	SYMBOLIC
577 760,,400 023	377 750	last address,,1st const
362	751	config 34 set up
34 21 00 377,751	752	| | | ↓ ^{34}SPF 377,751
01 11 02 377,750	753	| | ↓ ↓ ^{1}RSX^{2} 377,750
02 11 03 377,750	754	^{2}RSX^{3} 377,750
74 11 71 377,762	755	: ^{34}RSX^{71} 377,762
40 20 71 377,763	756	↓ ↓ ↓ ↓ : ^{0}LDE^{71} 377,763 ←
54 17 62 377,744	757	: SKN$^{3\cdot2}$e reg
41 15 02 377,744	377 760	| | ↓ ↓ : ^{1}ADX^{2} e reg
40 30 71 000,007	761	↓ ↓ ↓ ↓ : STE^{71} 000,007
41 0 771 377,756	762	: $^{+1}$JNX^{71} 377,756
70 06 03 377,752	763	: $^{-7}$JPX^{3} restart
41 11 02 377,761	764	| | ↓ ↓ : ^{1}RSX^{2} 377,761
01 15 02 377,744	765	| | ↓ ↓ ^{1}ADX^{2} e reg
01 30 00 000,005	766	| | ↓ ↓ ^{1}STE 5
00 05 00 000,001	377 767	JMP 1

This program		then forms	this one
001	: ^{34}RSX^{71} 6	010	: ^{34}RSX^{71} 15
2	: ^{0}LDE^{71} 7 ←	11	: ^{0}LDE^{71} 16 ←
3	: SKN$^{3\cdot2}$377,744	12	: SKN$^{3\cdot2}$377,744
4	: ^{1}ADX^{2} 377,744	13	: ^{1}ADX^{2} 377,744
5	: ^{0}STE^{71} 16	14	: STE^{71} 25
6	: $^{+1}$JNX^{71} 2	15	: $^{+1}$JNX^{71} 11
7	: $^{-7}$JPX^{3} 377,752	16	: $^{-7}$JPX^{3} 377,752

and so on...

<u>Program II</u> <u>The Inchworm</u>

OCTAL EQUIVALENT	ADDRESS	SYMBOLIC
577 760,,400 023	377 750	last address,,1st const
362	751	config 34 set up
34 21 00 377,751	752	| | | ↓ ^{34}SPF 377,751
01 11 02 377,750	753	| | ↓ ↓ ^{1}RSX^{2} 377,750
02 11 03 377,750	754	^{2}RSX^{3} 377,750
74 11 71 377,762	755	:^{34}RSX^{71}377,762
40 20 71 377,763	756	↓ ↓ ↓ ↓ : ^{0}LDE^{71}377,763
54 17 62 377,744	757	: SKN$^{3\cdot2}$e reg
41 15 02 377,744	377 760	| | ↓ ↓ : ^{1}ADX^{2} e reg
40 30 71 000,007	761	↓ ↓ ↓ ↓ : STE^{71}000,007
41 0 771 377,756	762	:$^{+1}$JNX^{71}377,756
70 06 03 377,752	763	:$^{-7}$JPX^{3} restart
41 11 02 377,761	764	| | ↓ ↓ : ^{1}RSX^{2} 377,761
01 15 02 377,744	765	| | ↓ ↓ ^{1}ADX^{2} e reg
01 30 00 000,005	766	| | ↓ ↓ ^{1}STE 5
00 05 00 000,001	377 767	JMP 1

This program then forms this one

001	:^{34}RSX^{71} 6
2	: ^{0}LDE^{71} 7
3	: SKN$^{3\cdot2}$377,744
4	: ^{1}ADX^{2} 377,744
5	: ^{0}STE^{71} 16
6	:$^{+1}$JNX^{71} 2
7	:$^{-7}$JPX^{3} 377,752

010	:^{34}RSX^{71} 15
11	: ^{0}LDE^{71} 16
12	: SKN$^{3\cdot2}$377,744
13	: ^{1}ADX^{2} 377,744
14	: STE^{71} 25
15	:$^{+1}$JNX^{71} 11
16	:$^{-7}$JPX^{3} 377,752

and so on...

Program II The Inchworm

The "flaw in the ointment" is that register 005 will contain 400,023 + 000,007 = 400,032 after the first mapping. The STE instruction in 005 would have a deferred (indirect) reference to 32 and this is clearly bad. It must be changed to a direct reference to 016. This is accomplished by the ADX in 765 which adds the 000,007, which by then is in index register 2, to the 000,007 which remains in the right half of the E register (after the RSX in 764) resulting in an 000,016 in the E register. The STE in 766 puts it away into 005 and the JMP transfers control to 001 continuing the process in core memory.

Exercises To Prove To Yourself That You Really Understand

Write a program which uses another approach to the problem of what to put in the 16 toggle switch registers to make core memory look as it does above.

Is it possible to use a JPX in 762 and, if so, what would the program look like then?

III. Through the Looking Glass

The Problem

If all the registers in any block of memory registers were laid end to end, what program would put the mirror image of this mess back into the memory block? For example, if the block consisted of three 4-bit words, the transformation would look like this:

$$\begin{array}{cccc} F_1 & F_2 & F_3 & F_4 \\ S_1 & S_2 & S_3 & S_4 \\ T_1 & T_2 & T_3 & T_4 \end{array} \quad \Rightarrow \quad \begin{array}{cccc} T_4 & T_3 & T_2 & T_1 \\ S_4 & S_3 & S_2 & S_1 \\ F_4 & F_3 & F_2 & F_1 \end{array}$$

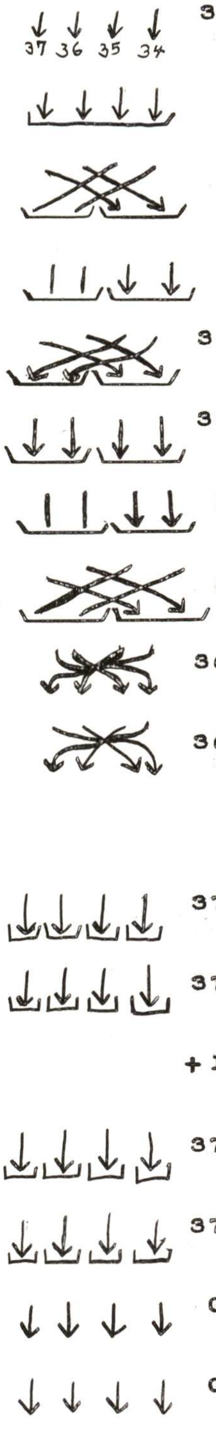
The Solution

Program III, which is written with floating addresses, performs this mirroring by the use of configurations and simultaneous cycling with only 20 instructions.

Four unusual configurations are needed and these are set up in configuration memory locations 37, 36, 35 and 34 by the SPG instruction.

From some register containing the first and last addresses of the memory block, the A register is set up and the "first" is put into the address section of the LDA instruction called "top," and the "last" is put into the LDB called "bot." The general idea is to index through the block, taking a pair of words at a time and exchanging and reversing them. One word comes from the top half of the block and the other comes from the bottom half. If the block has an odd number of words in it, the first pair will be the middle word used twice. If the block has an even number of words, the first pair will be the middle two words. The last pair dealt with is always the first and last words of the block.

Index register 8 contains a positive number which counts back from the middle of the block to the first. Index register 9 contains a negative number which counts up from the middle to the last. If there are 2n+2 or 2n+1 words in the block, index 8 starts out with +n and index 9 starts out with -n. These numbers are obtained from the first and last addresses after only two instructions. The first instruction is a SUB which subtracts, simultaneously, the last from the first and the first from the last! The left half of the A register then contains -(2n+1) for even blocks and -(2n) for odd blocks. The right half of A contains the complement of the left half.



start=| ^{34}SPG ⌊600,605,200,202

^{0}LDA ⌊first,,last

^{2}STA top

^{1}STA bot

^{34}SUB a reg

^{35}SCA ⌊-1,--,-1,--

^{1}RSX^{8} a reg

^{2}RSX^{9} a reg

top=| ^{36}LDA^{8} first

bot=| ^{36}LDB^{9} last

RSX^{1} ⌊-10

again=| ^{37}CAB ⌊-1,-1,-1,-1

^{37}CYB ⌊2,2,2,2

$^{+1}$JNX^{1} again

^{37}CYA ⌊-1,-1,-1,-1

^{37}CYB ⌊-1,-1,-1,-1

^{0}STA (top)

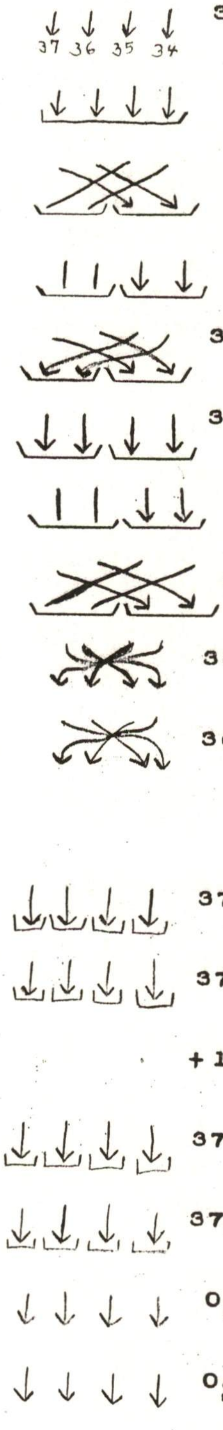
^{0}STB (bot)

$^{+1}$JNX^{9} d

d=| $^{-1}$JPX^{8} top

Done, halt or something...

Program III Memory Mirror



start=| ^{34}SPG ⌊600,605,200,202

^{0}LDA ⌊first,,last

^{2}STA top

^{1}STA bot

^{34}SUB a reg

^{35}SCA ⌊-1,--,-1,--

^{1}RSX^{8} a reg

^{2}RSX^{9} a reg

top=| ^{36}LDA^{8} first

bot=| ^{36}LDB^{9} last

RSX^{1} ⌊-10

again=| ^{37}CAB ⌊-1,-1,-1,-1

^{37}CYB ⌊2,2,2,2

$+^{1}$JNX^{1} again

^{37}CYA ⌊-1,-1,-1,-1

^{37}CYB ⌊-1,-1,-1,-1

^{0}STA (top)

^{0}STB (bot)

$+^{1}$JNX^{9} d

d=| $-^{1}$JPX^{8} top

Done, halt or something...

Program III Memory Mirror

The next instruction, SCA, shifts each half one place to the right, leaving -n in the left half and +n in the right half of A. Index registers 8 and 9 are then set up from the appropriate half of A.

The basic iterative loop starts now and is executed n times. The inner loop is executed 9 times for each of the n times through the outer loop. This number 9 is the number of bits in a quarter of a TX-2 word. If the reader wishes to work through an example with, let's say, 4 bit quarters, then he should go through the inner loop four times. The index register (1) is preset to -8 however, since the JNX jumps on zero.

The STA and STB instructions (at d-3) have deferred addresses which they get from "top" and "bot" respectively. This is actually inefficient timewise if n is greater than 2. Two more instructions when setting up could have put direct references to "first" and "last" in these STA and STB instructions. This would have cost 4 memory time cycles. However, each deferred address costs one memory cycle and so 2n-4 extra memory cycles are being executed in the basic loop. This illustrates how one can trade space for time or vice versa.

The two decimal numbers 8 and 9 were used to indicate general index registers. Of course, 1 is general too.

Exercises To Prove To Yourself That You Really Understand

One need execute the inner loop only 8 times if a slightly different correction is made afterwards. What are the new correcting cycle instructions?

Configuration 35 is not really needed. What other one used by Program III would serve just as well?

IV. 50 Million Multiplications Can't Be Wrong

The Problem

In the analysis of electroencephalographic data, the autocorrelation function of the data is often desired (see B. G. Farley). A specific useful example is the following: about 50 thousand samples are stored away in memory. Each sample is a sign and 8 bits (9 bits in all).

We wish to find

$$\sum_{j=1}^{j=50,000} Sj \cdot Sj+1, \qquad \text{for } i=0,1,\cdots,1000$$

where Sj is the j^{th} sample. These 1000 numbers are proportional to the autocorrelation function.

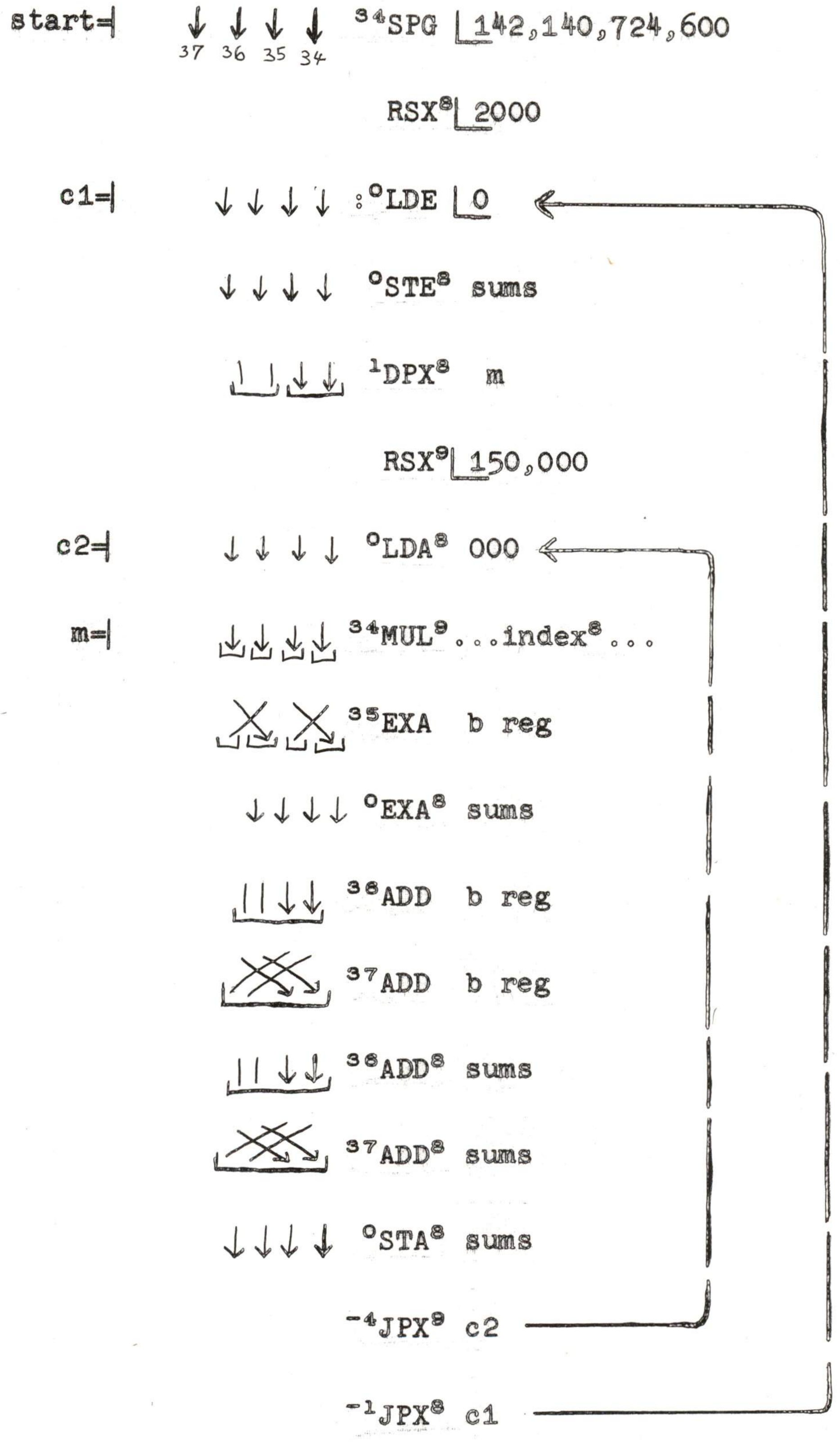
The Solution

Program IV computes this function in a most efficient way timewise. The key to the speed is to do four multiplications simultaneously. The data, however, must be in memory in a particular format, namely

0 =	S_1,	S_2,	S_3,	S_4
1	S_2,	S_3,	S_4,	S_5
2	S_3,	S_4,	S_5,	S_6
3	S_4,	S_5,	S_6,	S_7
4	S_5,	S_6,	S_7,	S_8
5	S_6,	S_7,	S_8,	S_9

Note that there are four of the 9 bit samples (Sj) in each TX-2 word and that registers 0, 4, etc. and 1, 5, etc. will contain eight different successive samples.

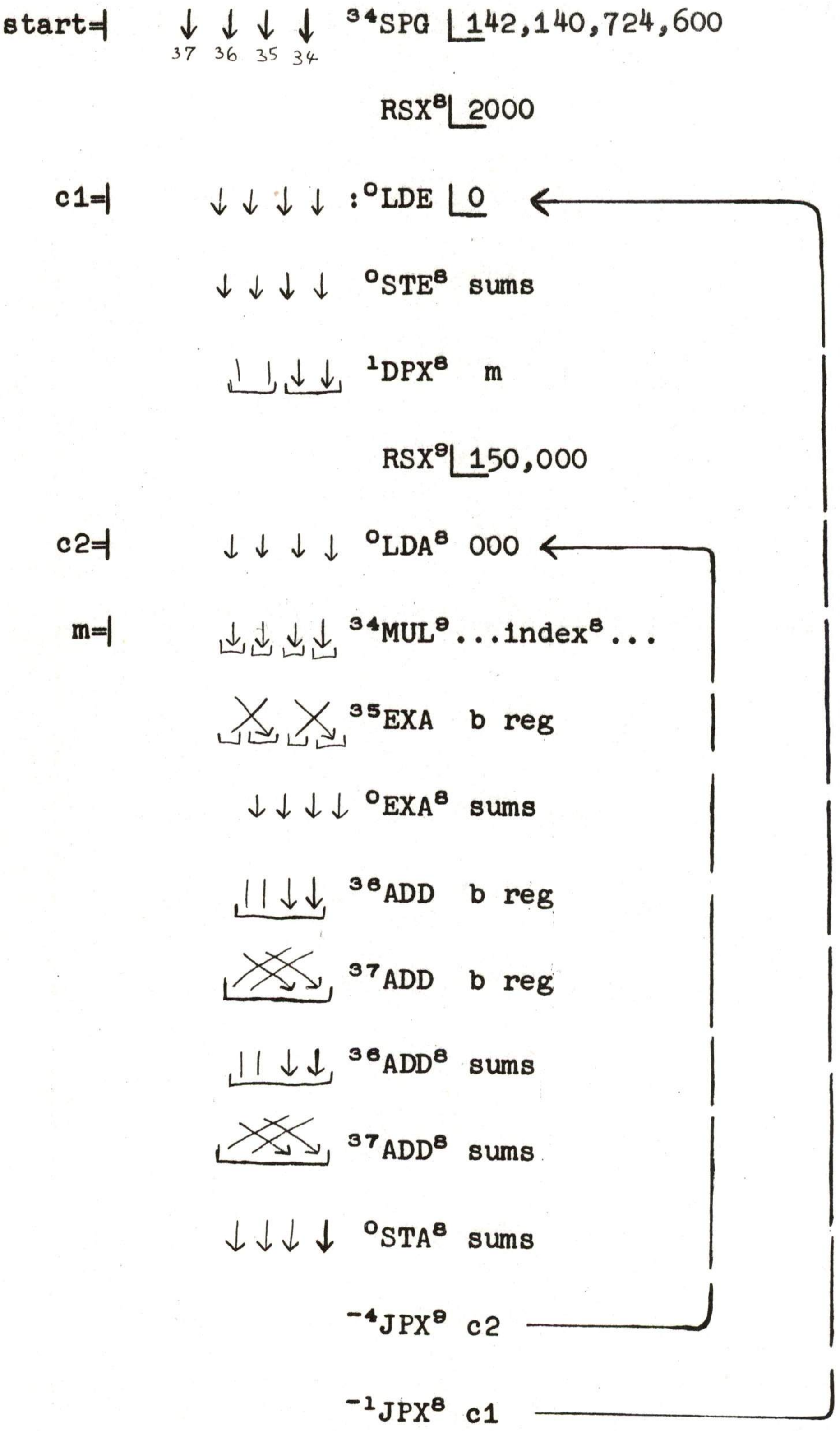
The program starts out by setting up the four special configurations needed and reseting index register 8 to 2000 octal (about 1000 decimal). Index register 8 corresponds to the subscript i in the summation above.



```
start=|   ↓ ↓ ↓ ↓   ^34SPG ⌊142,140,724,600
          37 36 35 34
                    RSX^8⌊2000
c1=|      ↓ ↓ ↓ ↓   :^0LDE ⌊0
          ↓ ↓ ↓ ↓   ^0STE^8 sums
                    ^1DPX^8  m
                    RSX^9⌊150,000
c2=|      ↓ ↓ ↓ ↓   ^0LDA^8 000
m=|                 ^34MUL^9...index^8...
                    ^35EXA  b reg
          ↓ ↓ ↓ ↓   ^0EXA^8 sums
                    ^38ADD  b reg
                    ^37ADD  b reg
                    ^38ADD^8 sums
                    ^37ADD^8 sums
          ↓ ↓ ↓ ↓   ^0STA^8 sums
                    ^-4JPX^9 c2
                    ^-1JPX^8 c1
```

Done, display results (the 2000 sums)...

Program IV An Autocorrelation Program



start=| ^{34}SPG ⌊142,140,724,600

RSX^{8}⌊2000

c1=| :^{0}LDE ⌊0

^{0}STE^{8} sums

^{1}DPX^{8} m

RSX^{9}⌊150,000

c2=| ^{0}LDA^{8} 000

m=| ^{34}MUL^{9}...index^{8}...

^{35}EXA b reg

^{0}EXA^{8} sums

^{36}ADD b reg

^{37}ADD b reg

^{36}ADD^{8} sums

^{37}ADD^{8} sums

^{0}STA^{8} sums

$^{-4}$JPX^{9} c2

$^{-1}$JPX^{8} c1

Done, display results (the 2000 sums)...

Program IV An Autocorrelation Program

The outer iterative loop then clears the i th <u>current sum register</u> and puts i into the address section of the MUL instruction at m. Index register 9 is set to 150,000 octal (about 50,000 decimal). Index 9 corresponds to the subscript j in the summation. This outer loop is executed about a thousand times.

The inner loop computes one complete summation (fixed i) taking four samples at a time. After the multiplication, the A and B registers look like this:

$$A = P_1, \quad P_2, \quad P_3, \quad P_4$$

$$B = L_1, \quad L_2, \quad L_3, \quad L_4$$

where P is the most significant 9 bits of the product and L is the least significant.

To eliminate round-off errors, the sums of each whole 18 bit product are accumulated. To put the 9 bit pieces of the product together, the A register is exchanged with the B register in such a manner that the result looks like this:

$$A = P_1 \quad L_1 \;,, \; P_3 \quad L_3$$

$$B = P_2 \quad L_2 \;,, \; P_4 \quad L_4$$

These four 18 bit numbers are then added to the current sum which is a 36 bit number. Notice how the sign extension feature allows a signed 18 bit number to be added to a signed 36 bit number.

Index register 9 is counted down by 4 (!!) since only every fourth register of four samples need be multiplied. This means the inner loop is executed only about 13,000 times instead of 50,000 times.

The whole program with its 50,000,000 multiplications will take 8 minutes if the overlapped memory feature is used (i.e. if instructions and data are in different memories).

<u>Exercises To Prove To Yourself That You Really Understand</u>

The data should extend to register 152,000. Why?

Write a program, using appropriate configurations (no shifting) and the TSD instruction, which will read the samples into memory in the desired format. This program would operate in the Epsco Datrac (an analog-to-digital converter) sequence. Each TSD will put a signed 9 bit number into quarter 1 of the E register. Ignore In-Out Select instructions. Nine instructions will do nicely.

Write a new inner loop to Program IV which handles data with only one sample per word. Five instructions including the JPX will do it. This inner loop will have to be executed the full 50 million times. How long will it take?

V. The Flexo-Octal Converter

The Problem

In the beginning of a binary computer's programming life, it is difficult to communicate with the machine. A series of programs must be written to "bootstrap" one's way into easy communication. This bootstrap series might go like this:

First) A three (or so) word program in toggle switch storage which would allow words to be written into memory one at a time. Call this P1.

Second) A short routine to convert programs to binary which have been typed on a flexo in a rigid, simple, fixed-address format. Call this P2. Associated with P2 is a program to punch out storage as a binary tape and a program to read in this binary tape. P1 loads P2 into memory. P2 converts the punch-out and read-in programs. The punch-out program punches out P2, the read-in routine and itself. From now on, the read-in routine can read in P2 and the punch-out routine, eliminating the need for P1.

Third) A longer routine which converts programs typed in a symbolic code, relative-address format. Call this P3. P2 converts P3 and punches it out, eliminating the need for P2.

Fourth) A routine to convert programs typed in a symbolic code, floating address format (P4). P4 is written in P3 format and converted by P3. At this point P1, P2 or P3 aren't needed any more and communication is fairly easy. In TX-0, P4 was called TODAL. A fifth stage might be an algebraic format converter like FORTRAN.

Programs V, VI and VII are proposed examples of the second stage. The octal converter recognizes the eight flexo symbols 0,1,2,3,4,5,6 and 7 takes their order into account. Some control characters are needed, such as carriage return to signify the end of a word and slash to allow address specifications. The space, tab, and comma are used to give some format control. The nullify is recognized so that tape "goofs" can be fixed up. The last four are ignored by the converter. A stop code signifies the end-of-tape condition.

The Solution

The program to do the octal conversion is Program V.

To decide what action to take on each character as it is read in, an Action Table is set up as is shown beside the program. An entry is made at the address, starting at 100, whose last 2 digits correspond to the flexo code of the appropriate character. The right 18 bits of each entry tell where to transfer control when that character is read in, and the left nine bits tell what the binary equivalent is

ACTION TABLE	Description	ROUTINE			
105| = 0,0,000 201	start at →	200| =		:IOS^{52}	read unsplayed
107| = 3,0,000 210	if slash →	201	||↓↓	^{1}STA	213
110| = 0,0,000 204		202	↘↘↘	^{2}RSX^{2}	105
113| = 4,0,000 210		203	↓↓↓↓	^{0}DPX^{0}	a reg
117| = 2,0,000 210	if ignored →	204		IOS^{52}	dismiss
123| = 5,0,000 210		205		:TSD	e reg
125| = 1,0,000 210		206	|||↓	^{3}RSX^{1}	e reg
127| = 7,0,000 210		207		^{0}JMP^{1}	(100)
133| = 6,0,000 210	if number →	210	↓↓↓↓	^{0}CYA	107
145| = 0,0,000 204		211	↘↘↘↘	^{6}ADD^{1}	100
151| = 0,0,000 213		212		JMP	204
161| = 0,0,000 216	if car/ret →	213	↓↓↓↓	^{0}STA^{2}	memory
176| = 0,0,000 210		214	↘↘↘↘	^{6}AUX^{2}	125
177| = 0,0,000 204		215		JMP	203
	if stop code →	216		IOS^{52}	shut off

address| = value,0,where to go

Program V The Flexo-Octal Converter

ACTION TABLE	Description	ROUTINE		
105| = 0,0,000 201	start at →	200| =	:IOS^{52}read unsplayed	
107| = 3,0,000 210	if slash →	201	^{1}STA	213
110| = 0,0,000 204		202	^{2}RSX^{2}	105
113| = 4,0,000 210		203	^{0}DPX^{0}	a reg
117| = 2,0,000 210	if ignored →	204	IOS^{52}	dismiss
123| = 5,0,000 210		205	:TSD	e reg
125| = 1,0,000 210		206	^{3}RSX^{1}	e reg
127| = 7,0,000 210		207	^{0}JMP^{1}	(100)
133| = 6,0,000 210	if number →	210	^{0}CYA	107
145| = 0,0,000 204		211	^{6}ADD^{1}	100
151| = 0,0,000 213		212	JMP	204
161| = 0,0,000 216	if car/ret →	213	^{0}STA^{2}	memory
176| = 0,0,000 210		214	^{6}AUX^{2}	125
177| = 0,0,000 204		215	JMP	203
	if stop code →	216	IOS^{52}shut off	

address| = value,0,where to go

Program V The Flexo-Octal Converter

when the character is a number. Quarter 3 of each entry is not used.

The IOS in 200 sets the mode of the PETR to read one contiguous 6-bit flexo code (unsplayed) into the right 6 bits of the E register, clearing the other 3 bits in that quarter.

Starting at 203 with a DPX which clears the A register, the character is read in and placed in index register 1. The JMP then defers control to a location specified by the appropriate Action Table entry. Note that all instructions with deferred addresses are indexable.

If the character is a number, then control goes to 210 where the A register is cycled left 3 places and the binary equivalent of the number is added into A, returning control to 204.

If the character is a slash, control "bounces off" register 105 to register 201 where the number in A is stored in 213 and index register 2 reset to a zero. The slash then causes the number that has been built up in A to be the new address of the word which follows.

If the character is a carriage return, 213 has control and stores the word in A away in the proper memory location. The AUX in 214 adds a 1 to index register 2 so that the next time a carriage return appears, the word in A will be stored in the memory register following the last one.

The nullify, space, and tab simply return control to 204 to read-in the next character. When a stop code comes along, the IOS in 216 shuts off the photo reader and dismisses the sequence.

The sequence must be dismissed after each character is read and the IOS in 204 does this. The TSD in 205 empties a buffer that has been filled by the PETR. When the buffer is filled, the sequence is activated and the character read-in is dealt with.

Exercises To Prove To Yourself That You Really Understand

What are the implications of throwing out the IOS in 204 and not holding on the TSD which follows? In other words, let the TSD dismiss the sequence after transferring the data. Work out the new program and format rule(s).

The instructions in 210-11 are on rather shaky ground because TX-2 is an allegedly multi-sequence machine. Some lower priority sequence may have been using the A register and will be very upset at finding it disturbed. What changes will fix this up? Don't forget 203!!

Is there anything fishy about the RSX in 206?

VI. A Binary Read-In Routine

The Problem

In one of its modes, the photoreader reads the six bits of a line of tape into every sixth bit of some specified word and cycles the word left one place. This is the "splayed" mode of the photoreader sequence. After reading in six lines, a full 36 bit word is assembled. This mode would usually be used to read in binary tapes.

The main problem associated with a binary read-in routine is what format to use. In general, data words are read into blocks of consecutive memory registers and three provisions are made; (1) to read in more than one block, (2) to check the sum of each block thereby detecting almost any error, (3) to specify what should happen to control after all blocks are read in.

The Solution

Program VI uses the following format for each block of binary words:

-n ,, last address

Word_0

Word_1

.

.

.

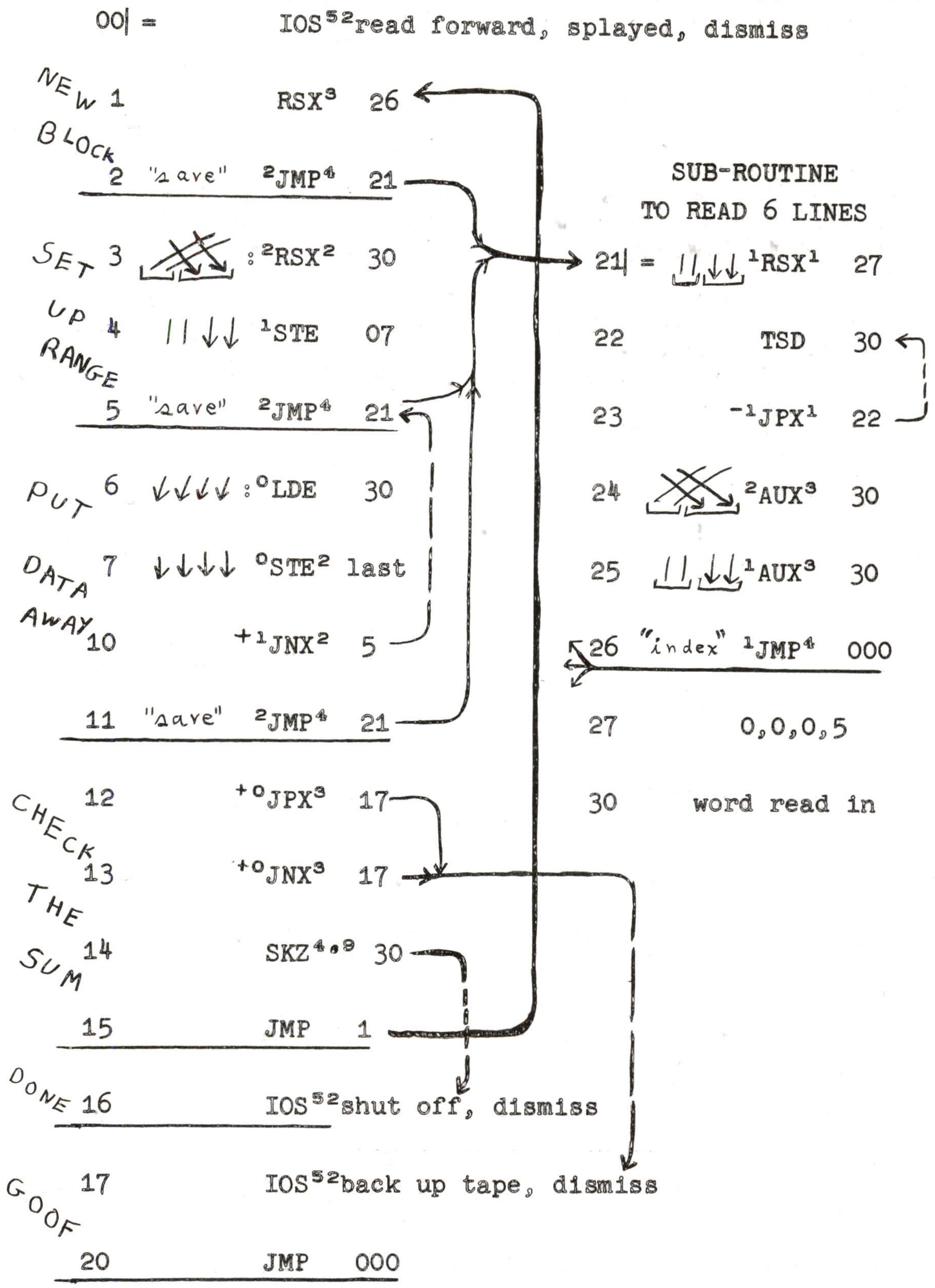
Word_n

more? ,, - sum

The first word in each block consists of two 18 bit numbers (see instructions at 3 and 4) which designate the addresses of the actual data words which follow.

The right half of the last word is the complement of the sum of all the other half words in the block. In other words, if all the words in a block are added up in 18 bit pieces (instructions at 24 and 25) the sum must be zero (instructions at 12 and 13) or there has been an error. If there is an error, the tape is backed up (instruction 17) and read in again. (TX-2, as you may have guessed by now, can read paper tape in either direction and can identify the front of the tape.)

The sign bit (4.9) of the left half of the last word in a block tells whether there are more (if 4.9 is a ONE) blocks to be read in or not (if 4.9 is a ZERO, see instruction 14). If there are more



00| = IOS^{52}read forward, splayed, dismiss

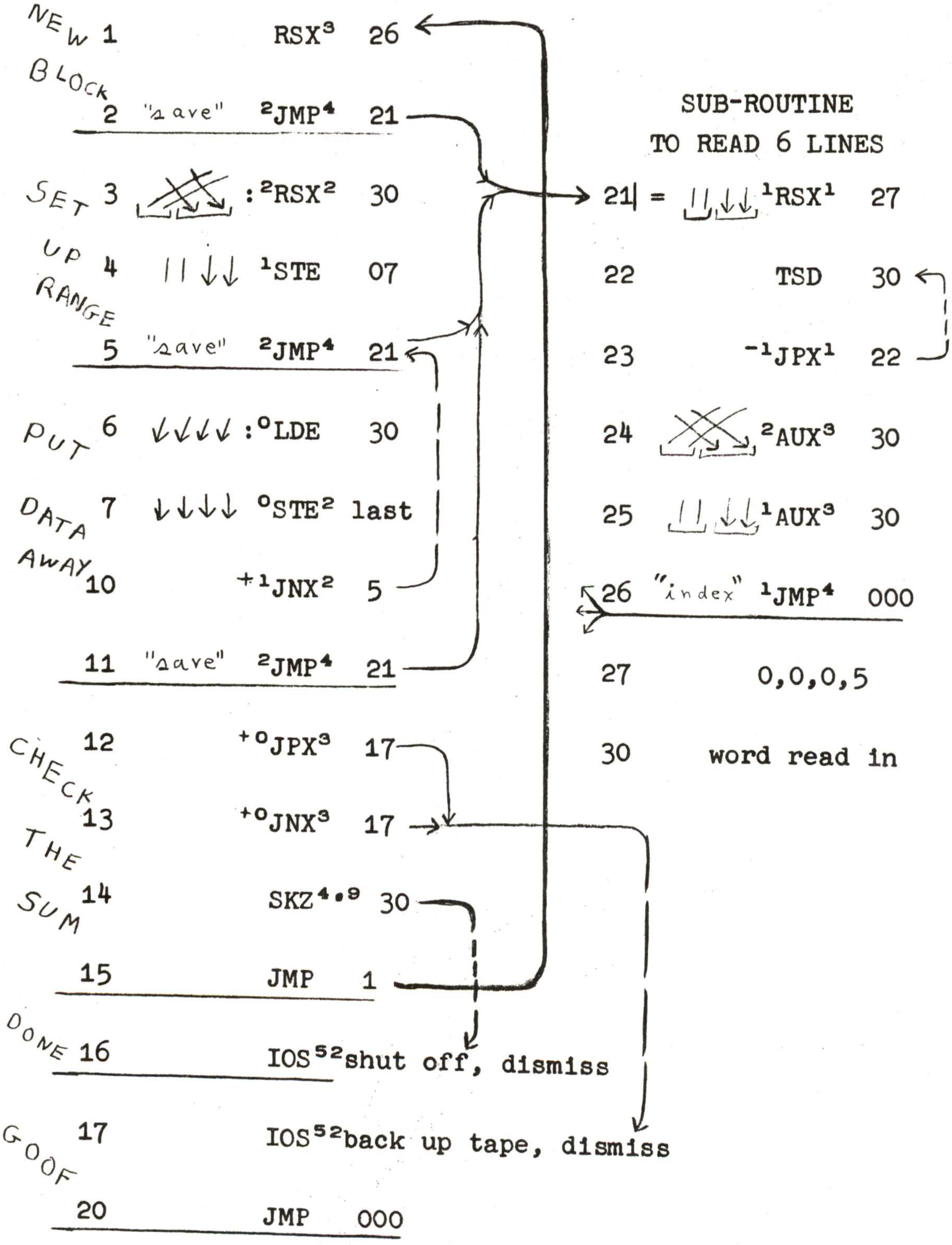
NEW BLOCK	1		RSX^{3}	26
	2	"save"	^{2}JMP^{4}	21
SET UP RANGE	3	:	^{2}RSX^{2}	30
	4	||↓↓	^{1}STE	07
	5	"save"	^{2}JMP^{4}	21
PUT DATA AWAY	6	↓↓↓↓ :	^{0}LDE	30
	7	↓↓↓↓	^{0}STE^{2}	last
	10		$^{+1}$JNX^{2}	5
	11	"save"	^{2}JMP^{4}	21
CHECK THE SUM	12		$^{+0}$JPX^{3}	17
	13		$^{+0}$JNX^{3}	17
	14		SKZ^{4,9}	30
	15		JMP	1
DONE	16		IOS^{52}shut off, dismiss	
GOOF	17		IOS^{52}back up tape, dismiss	
	20		JMP	000

SUB-ROUTINE TO READ 6 LINES

21| =	||↓↓	^{1}RSX^{1}	27
22		TSD	30
23		$^{-1}$JPX^{1}	22
24		^{2}AUX^{3}	30
25	||↓↓	^{1}AUX^{3}	30
26	"index"	^{1}JMP^{4}	000
27		0,0,0,5	
30		word read in	

Program VI A Binary Read-in Routine

00| = IOS^{52}read forward, splayed, dismiss



NEW BLOCK

1 RSX^{3} 26

2 "save" ^{2}JMP^{4} 21

SET UP RANGE

3 :^{2}RSX^{2} 30

4 ||↓↓ ^{1}STE 07

5 "save" ^{2}JMP^{4} 21

PUT DATA AWAY

6 ↓↓↓↓ :^{0}LDE 30

7 ↓↓↓↓ ^{0}STE^{2} last

10 $^{+1}$JNX^{2} 5

11 "save" ^{2}JMP^{4} 21

CHECK THE SUM

12 $^{+0}$JPX^{3} 17

13 $^{+0}$JNX^{3} 17

14 SKZ$^{4.9}$ 30

15 JMP 1

DONE 16 IOS^{52}shut off, dismiss

GOOF 17 IOS^{52}back up tape, dismiss

20 JMP 000

SUB-ROUTINE
TO READ 6 LINES

21| = ||↓↓ ^{1}RSX^{1} 27

22 TSD 30

23 $^{-1}$JPX^{1} 22

24 ^{2}AUX^{3} 30

25 ||↓↓ ^{1}AUX^{3} 30

26 "index" ^{1}JMP^{4} 000

27 0,0,0,5

30 word read in

Program VI A Binary Read-in Routine

blocks, control goes to register 1 and reads in the next block, providing of course that there were no check sum errors. If there are no more blocks, instruction 16 shuts off the PETR and dismisses the sequence.

Exercises To Prove To Yourself That You Really Understand

Note that there is no provision made in the tape format of Program VI for turning on any other sequence after the last block has been read in. There is really no necessity for a control change since the Start-Over sequence can start up the program just read in at the poke of a button.

However, pay homage to the (W. A.) Clarkian philosophy of minimal button poking and make the necessary additions of Program VI and its format which will start the program in sequence #S at a register called START if bit 4.8 of the last word in the last block is a ONE. If 4.8 is a ZERO, make Program VI do what it does now. This addition can be accomplished with eleven more words (maybe fewer).

Why are the CF bits of instruction 12 all ZEROS?

Do they need to be ZEROS in instruction 13? Why?

VII. A Punch Out Routine

To prove to yourself that you really, really understand, write a program to punch out storage in the block format required by the read in routine (VI). Control it from a toggle switch register in the following manner:

> Let the left half of the toggle switch register be the first address, and the right half, the last address of the block to be punched out.
>
> Let the meta bit (4.10) designate whether this is the last block or not.
>
> Let bit 4.9 be a ONE when the toggles are being changed, and a ZERO when the program can look at the register.

The author has written this program with 33 instructions. The best solution submitted by a reader, will be published in a supplement to this memo.

CONCLUDING REMARKS

The six programs in this memo illustrate many of the characteristics of TX-2. There are other features which haven't been illustrated. For example, conditionally saving the P and/or Q register in E after a JMP; using multiple step deferred (indirect) addresses; using the Boolean instructions or the skip if E is different from word instruction; using the operate class commands and many sequences operating simultaneously.

There will be supplements to this memo from time to time which illustrate features such as those mentioned in the preceding paragraph. Any suggestions, improvements, discoveries, or remarks in general will be appreciated by the author and probably also by his associates.

HPP/mk

Insertions:

Pages 3a
6a
9a
12a
15a
18a

NOTES