

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

1 July 1964

TO: TX-2 Users et al
FROM: Alex
SUBJ: TX-2 Schedule

1. Users List

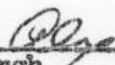
A list of users with sign-up initials and phone numbers has been posted in the computer room. Additions to the list must be approved by the Group 23 office. Please notify Alex of any changes in telephone numbers, room numbers, etc.


2. Demonstrations

Demonstration time (time labeled demonstration and extending the time normally available to a user) should be arranged by the Group Leader through the Group 23 office. If possible please give advanced notice of at least a week.

3. M.C. Time

M.C. stands for "Marginal Checking" and M.C. Time will normally be used for this purpose. It will also be used to compensate users for time lost during the day due to computer malfunction.


A. Vanderburgh


J. Mitchell

AV/jk

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

20 July 1964

TO: MK4 Users
FROM: L. G. Roberts
SUBJ: MK4 Notes and Problems

There are several mistakes or loopholes in MK4 which should be kept in mind by users since I do not intend to fix them in the near future. Some of them have been with us for years and may be well known to many. The intent of this list is to provide a checklist of things to check for when trouble occurs as well as to prevent trouble.

A. Problems which may cause alarms or loops on LW readin.

- 1.) Wrong number of brackets or parentheses on a line. (Don't match up)
- 2.) The use of the bar in subscript in a macro parameter. There is no cure for this except to revert to the old form for a double index.

Wrong: MAC | (MKZ_{2.9}|₂T)

Cure: MAC | MKZ_{2.9}* {T₂}

- 3.) Remember to use ~~END~~ EMD at the end of a macro definition or modification. Either ~~END~~ END or ~~EMD~~ EMD can be used to complete a normal insertion or replacement.

B. Problems which cause a wrong but undetected error during assembly.

- 1.) A tab, comment after a macro statement when a parameter is expected will look like a zero parameter. (parameter = tab)

Wrong: A₁ **COMMENT

Cure₁: A₁**DONT TAB IN THIS CASE

Cure₂: (A₁) **PARENTHE SIZE

O.K.: A₁B **COMMENT

- 2.) Don't ~~use~~ KEEP a statement with both a tag and a statement which depends on a possible missing parameter. The same holds for ~~use~~ IFN|P when the statement uses P. The result will be that the statement is kept on first pass but not on second pass so that some addresses are skipped in your binary program. The cure is to put the tag on the line above the ~~use~~ KEEP with no statement after it.

Wrong: ~~use~~KEEP TAG-> MAC|P,Q

Cure: TAG->
 ~~use~~KEEP MAC|P,Q

- 3.) Don't put both a normal statement and an equality on the same line in a macro. This is legal in your main program but causes the same first pass, second pass problem as above when it occurs in a macro.

Wrong: LDA T T=P+Q

Cure: LDA T
 T=P+Q

- 4.) When using the line extension feature, a minus sign before the carriage return, to write a long macro statement, it is not legal to split in the middle of a symex or after parameters and terminators which occur before the macro name. Thus, if A|B~~CR~~C,D is a macro form:

Wrong: (A|B-
 ~~CR~~C,D)

Cure: (A|B~~CR~~-
 C,D)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

30 September 1964

TO: TX-2 Users
FROM: L. Fleck and L. Roberts
SUBJ: Xerox Package

There is now a Xerox Package available which was taken from M4 and modified to use only two index registers and no buffer at the moment. It will be used in Coral and by any user who need a separate Xerox routine. It uses 665₈ locations and had no origin.

The user will make an initial hJPQ XEROX with the title in A and name in B, which turns on the Xerox and stores the name and title to be printed on each page. The title is set up with the appropriate LW codes, one in each quarter, beginning in quarter 1. This applies to the name also, except only 3 characters are used.

Thereafter, the desired control codes and LW codes are transmitted one at a time to the routine by storing them in quarter 1 of A and following with the macro SEND, or else, SEND|code will accomplish this.

If no control codes are given for paging or frame length, the normal paging mode of 8 1/2" and 46 lines including title line will be used with the 66 MS frame length.

Upper case LW characters are indicated by setting bit 1.7 in the LW code and color red by bit 1.8, resulting in ~ under the Xeroxed character.

Following are the special LW codes for format control:-----

- 14 Finish page and title next page
- 114 No paging mode. If already in this mode, it will cause immediate chop and title.

- 15 66 MS frame length, 46 lines/page.
- 115 40 MS frame length, 74 lines/page.
- 16 Small character separation (for numbers).
- 116 Large character separation (for text).
- 17 Large bar and return to large separation.
- 71 New horizontal position to be given as next character (0 → 777);
new line if new position is less than last horiz. position.
- 171 Same as 71, but no new line (for graphs).
- 76 Finish page and stop Xerox.

Subroutines in Xerox Package: (use index α and AE)

1. Decimal Fraction -

Given octal number in A, convert to decimal fraction and print.

JES _{γ} DFRAC

2. Decimal Integer -

Given octal number in A, convert to signed decimal integer and print.

- JES _{γ} DINT - left adjusted start point, no spaces, print n digits.
- JES _{γ} DINTF - right adjusted, leaves (11 - n) spaces, prints n digits.
- JES _{γ} DINT α - right adjusted for short words. Set α to number of digits expected, leaves (α - n) spaces for n digits.

3. Octal Number -

Given octal number in A, print as octal number.

- JES _{γ} OCTH - print left 6 digits.
- JES _{γ} OCTIF - print full word, spacing for preceding zeros.
- JES _{γ} OCTI - print full word, no spacing for preceding zeros.

4. Octal Address -

Print octal address in right half of A, suppressing first 3 digits as in MK4. Set bit 1.1 of BEXX on first entry for full address.

JES_γ LADR

5. Address and Contents-

Print octal address as above, followed by large bar and address contents printed as two 6-digit octal numbers, and large bar, as in MK4 listing.

JES_γ OLSI

6. Text

Print information represented by LW codes following the call instruction.

JES_γ LWTAB

4, 3, 2, 1

-4 LW codes/word

8, 7, 6, 5

0, 0, 400, 9

-Setting bit 1.9 following the last code terminates the test. Return point is to the next address.

NOTE: See Laurice Fleck for directive and questions.

M. Morissey

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Lincoln Laboratory

19 October 1964

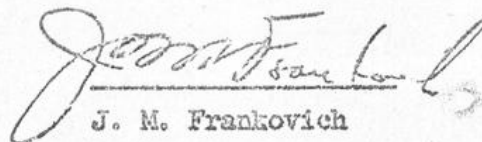
TO: TX-2 Users
FROM: J. M. Frankovich
SUBJECT: Availability of Fastrand Drum on TX-2

A Univac Fastrand drum finally passed acceptance tests on 16 October 1964. This unit will now be available for TX-2 use. A memo describing specifications for the unit and details for programming for it will be available soon from John Laynor. General use, however, of the drum will be obtained via either the Mark IV or the new APEX operating systems. Direct use of the drum by non-system programs will not be permitted except by arrangement with users who are prepared to take full responsibility for preserving other system and user information stored on the drum.

Details of the appropriate Mark IV drum commands can be obtained from Larry Roberts and Alex Vanderburgh as soon as they are stabilized. To the system user the drum will be made to look like TX-2 magnetic tape but with approximately four times the capacity and a comparatively insignificant search and transfer time.

The new APEX operating system with improved file maintenance procedures and more efficient use of the drum will be available some time after the first of the year. Details on this are available from Jim Forgie.

Users of the drum should, at least initially, indicate in the TX-2 log their use of the drum. Drum errors will be indicated on the Lincolnwriter and, hopefully, will usually fall in the category of parity read errors. Attempts to read again can sometimes overcome this type of error. Other errors should be adequately documented so that both Univac and TX-2 maintenance people can analyze the problem.


J. M. Frankovich

JMF/smc

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

22 October 1964

To: TX-2 Users
 From: Larry & Alx
 Subject: Fastrand Drum Version of M4 (Brown Reel)

1. There is now a BROWN REEL. It puts a drum read program in registers 100-500 which reads in a new version of M4 from the drum. The inside program on the brown reel reads the old m4 from the drum.
2. The Drum format is "permanent" - That is, you can expect your data to remain there until the advent of APEX. (And, perhaps for a while thereafter.)
3. The drum version has NO ACCESS TO TX-2 TAPE. (Save, Read, Core, Tape -- are ignored)
4. The Drum version has several new meta-commands:

FFAREAD	Similar to	FFREAD
FFAWRITE	Similar to	FFSAVE
FFAREAD	Similar to	FFCORE
FFAWRITE	Similar to	FFTAPE

But note that the core area need not be given if you wish it to go back where it came from.

FFBLIST	Bins the listed part. Same format as	FFLIST	
FFEDIT	Like	FFDIR	Memo will follow. Edit and Look use the Line Drawing Scope.
FFLOOK	Like	FFTYPE	Use "begin" key to return to M4. Look lets you see part of a directive. Dont ask for more than 20 lines.
FFCORAL	Puts CORAL subroutines where they belong. (206000-215500)		

5. The TX-2 Tape subroutines (JPQ 160005,6) DO NOT WORK, and there are no similar Drum routines available as yet. (They are still OK on the std. M4.) The other subroutines are OK.
6. The reply from the Drum meta-commands may give you two check sum words. This means trouble of some sort. You may want to report it. Possible causes:
 - a. Reading first of overlapped pair.
 - b. Drum error on either read or write.

The drum is parity checked, and the check sum word is saved with the data. The first check sum word reported is the one computed in core. When it is not the same as the drum word, you get another.

Alx

Fastrand Allocation

The drum is addressed with 19 bits, the high order 7 of which are the "position" number and the low order 12 are the "sector" number. Each sector is 28 words long, therefore MK4 uses 5 sector blocks, wasting a few words, to provide 200₈ word blocks like TX-2 tape. A "position" contains 1400₈ blocks so each user has been given a complete position. Position 0 contains the system programs in binary and is locked against writing by hardware. Position 137, the top position, is used for saving memory during swaps. Positions 1-42 are allocated for users as provided on the following chart.

<u>Position #</u>	<u>Initials</u>	<u>User</u>
1	JSH	Jane Heart
2	PLF	Phil & Laurie Fleck
3	ALX	Alex Vanderburgh
4	PAT	Pat Fergus
5	RAY	Ray Weisen
6	LGR	Larry Roberts
7	MAT	Maintenance
10	MK4	Mark 4
11	CDF	Carma Forgie
12	RAD	Charley Rader
13	DON	Don Malpass
14	JAF	Jerry Feldman
15	WIL	Willy Kantrowitz
16	LR2	Larry Roberts II
17	WRS	Bert Sutherland
20	HEN	Ben Gold
21	BOG	Ben (other) Gold
22	TOM	Tom Stockham
23	CLO	Cynthia Lo
24	GUP	General Utility Program
25	APE	Apex
26	YNT	Douwe Yntema
27	LES	Lester Ernest
30	TM2	Tim Johnson II
31	TIM	Tim Johnson
32	STO	Arthur Stowe
33	MLW	Marilyn Wendell
34	CKM	Connie McElwain
35	FRE	Free
36	JMF	John Frankovich
37	KRG	Kathy Goldsmith
40	JWF	Jim Forgie
41	MLG	Mason Groves
42	SPE	System Prog. Exec.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

4 November 1964

TO: TX-2 Users
FROM: T. Stockham (via Alx)
SUBJ: EDIT and LOOK Meta-commands

Two new features have been incorporated in the drum version of MK4. They are the EDIT and LOOK meta-commands, and they behave essentially as TDIR and TYPE respectively except that they display on the line-drawing scope instead typing Lincolnwriter output. In addition, EDIT provides a facility for making changes in the displayed directive and automatically reflects these changes in the stored directive when the meta-command is terminated. Roughly EDIT is a combination of TDIR and REPLACE. LOOK cannot be used for editing and cannot affect the stored directive in any way. It provides a very swift partial listing.

To use EDIT, type `≡EDIT` AA→BB

or any similar address range, but do not use AA→ alone. (The scope has room for 20 lines.)

To use LOOK, type `≡LOOK` AA→BB

or any similar address range, including AA→ (if the RC block is short enough). (The scope has room for 20 lines.)

To terminate either command, strike the BEGIN key.

If one uses EDIT and wishes to make changes in the displayed directive, one merely types the desired modifications before terminating the command by striking the READ IN key. The rules for typing are explained in the following paragraphs.

The EDIT command has two modes:

- a. EDIT
- b. AUGMENT

When the command is first entered it is in the AUGMENT mode. The color shift keys are used for changing modes. RED gives EDIT mode; BLACK gives AUGMENT mode. All other keys except

- a. WORD EXAM
- b. NO
- c. YES
- d. BEGIN
- e. READ IN

are used for placing characters into and out of the directive. Characters are removed by means of the DELETE and STOP keys.

In the AUGMENT mode, characters are entered at the end of the displayed directive as they are typed. DELETE removes the character just typed and STOP removes all characters back to, but not including the last carriage return. READ IN terminates the command and, provided some changes have occurred, causes the stored directive to be modified. BEGIN terminates the command without modifying the stored directive. WORD EXAM, NO, and YES have no effect in the AUGMENT mode.

In the EDIT mode a marker is displayed within the text in the shape of a box. As they are typed, characters are entered into the directive just to the left of the box. DELETE removes the character within the box, and STOP removes that character and the rest of the line. The marker may be moved in several ways. YES moves it one character to the right, and NO moves it to the left similarly. WORD EXAM moves the marker to the beginning of the next string of visible characters. BEGIN moves the marker to the previous line beginning, and READ IN moves the marker to the next line beginning. Note that one must return to the AUGMENT mode before the meta-command may be terminated.

While editing, it is the users responsibility to keep track of all invisible characters except case change characters. MK4 keeps track of case with a case bit. It associates such a bit with each character so that when one deletes upper case characters imbedded in lower case surroundings it is not necessary to worry about deleting case changes. Such diligence is necessary while editing a paper tape.

A bit is not used for super and sub-script however. In this context it is sometimes difficult to tell whether the marker box is before or after script change characters, tabs, and carriage returns. Nevertheless, if the text being displayed looks correct after editing has taken place, all is well since all redundant invisible characters are removed by MK4 when the READ IN key is pressed.

When EDIT or LOOK is typed, M4 transfers programs in from the drum after saving the appropriate core on the drum. Upon completion of these commands core is restored as it was. Even if Codobo is hit to M4 during a display, core is restored so that one need not worry about changing memory or about the drum noises which will be heard.

TS/jk

Olps

TO: TX-2 Distribution List
FROM: L. G. Roberts
SUBJECT: MK4 Off Drum Package

7 January 1965

With new memory, SPAT, and the drum available now, a drum package has been prepared which reads-in system programs, operates M4 under SPAT control, and can be used for drum swaps via subroutine calls. First the package and its intended use will be described, then the use of the machine and M4 with SPAT on, and third, the drum subroutine calls.

The brown reel is now a binary tape of the new package which reads into memory at 274000 - 275100. However, the last 4K of memory (274000 - 300000 at the moment) is hereby reserved for extension of this package. When the paper tape is read in, an automatic JFQ₀ 274000 is performed. This sets up SPAT memories, reads M4 off the drum and jumps to M4. If SPAT P & Q OFF switches on the console are lit up, then the machine will operate as it normally has and the only effect of loading SPAT will be to clear out SPAL errors in the special memories. The reasons for using SPAT with M4 are to speed it up and avoid MPALS from S memory. Operating with SPAT on reduces M4 compile and bin time to 60 percent of what they used to take.

DRUM PACKAGE

- A. If the machine has been turned off:
 1. Load the Brown Reel.
 2. Suppress MPAL, XPAL, and SPAL.
 3. Make sure P & Q OFF switches are lit. (SPAT off.)
 4. Codabo to 377770.
This will clear all memory registers, set SPAT memories, and read M4 off the drum.
- B. Now if you wish to use M4 with SPAT, turn SPAT on (i.e., Lights off) and execute a JFQ₇ 274000. (Explained further later on.)
- C. Once the package has been read in, there should be NO reason to clear memory and read it in again unless there has been an MPAL in upper U memory (274000 - 300000) or unless this area has been destroyed by a wild program. The package restores SPAT on every entry and upon entry #7 it will clear all memory below it.
- D. Entry procedure:
All calls are made by a JFQ_x 274000.
The index 'x' indicates the entry desired.
The JFQ may be held or not as necessary.

Entries:

x = 0	Read drum M4 off drum, go to 160000.
= 1	" drum " " " " " 160001.
= 2	" tape " " " " " 160000.
= 3	" tape " " " " " 160001.
= 5	Go to 377750 to read paper tape.
= 6	Read CORAL off drum, and dismiss.
= 7	Clear (with #) lower memory (0-274000) and then read drum M4 off drum and go to 160000.
= 10	Read binary record off drum. *
= 11	Write binary record on temporary drum positions. *

* Paragraph on drum routines below.

SPAT UTILIZATION

A. Facts about SPAT:

1. SPAT is ON when P and Q SPAT OFF switches are NOT LIT.
2. Sequence Zero commands are never transformed by SPAT.
3. Addresses 377400 - 377777 are not transformed.
4. PAM memory has addresses 374000 - 376000.
Bits 1.1 - 1.9 are real core page #, rest of bits = 0
5. LBM memory has addresses 376000 - 376100.
Bits 1.1 - 2.1 are the Relocation #, (R)
Bits 4.1 - 4.7 are the Bound #, (B)
6. Four LBM registers are used for each group of sequences.
Presently sequences are grouped by pairs. (High order two bits of K ignored.) All book zero registers are first in LBM, then book 1, etc. However, the SPAT setup used by the drum package makes all sequences have the same setup. That is LBM has all books full and all sequences the same.
7. PAM is loaded to transform memory as shown on the chart on page (5) placing M4 in T memory and its data in U memory. This speeds up M4 so that it takes only 60 per cent of the time it used to. Lower S memory, old T memory and upper U memory are not relocated so that if SPAT is turned off, these areas of memory will still be where they appeared.
The second half of PAM is loaded with a pure count or a straight transformation but is not used.

B. Using SPAT

1. Do Not Codabo to programs directly, since sequence zero is not spated. Set a RFD₇₀ 160001 or other appropriate sequence change in the special Codabo togs and you will get to your program, not some queer memory location.
2. Do not try to use a SKM loop or run programs in togs in sequence zero for the same reason as above.
3. Either run your program with SPAT on when you are through with M4 or store everything you want in the non-relocated memory areas.
4. If alarms occur, P and Q indicate the transformed addresses. So for debugging you need not worry about SPAT. However, the "real" memory addresses are shown in PA and QA on the right below the real time clock if you want to know the memory that failed.
5. If you use SPAT, clear memory with the command JPQ₇ 274000, since this will put the apparent address instead of the real address in all registers.
6. SPAL errors may occur after the initial clearing of memory indicating failures in SPAT or its memories. Record these like any other machine failure.
7. If you want to change PAM or LEM for your own purposes, do so at the risk of being incompatable with the executive when it arrives. Although the present packages will remain, the exotic users will have less time available.

DRUM ROUTINES

A. To Read Binary Records off Drum:

Any record written by M4 with a ~~W~~ #W can be dumped back into memory with a subroutine call to the package. If the record was written at block BLK under the name NAM, use:

JPQ₁₀ 274000

0, 34, 20, 35

BLK

JPQ ERROR

** Lincoln Writer Codes for NAM

** Block # written at.

** Normal return (in sequence 70)

This call can be used from any user program or togs (in any sequence) but you must not multi-sequence with the drum. The AE and indexes 35, 36, 37 are used and not restored by the routines. Upon a normal return, A contains the addresses where the record was stored;

A = FIRST , , LAST (exclusive)

B. To Write a Binary Record on Temporary Positions -

Two drum positions of 1400 blocks each (135 and 136) have been assigned to the use of this write command. There is no facility for you to write in your own or anyone else's drum position. Very similiar to a ~~W~~ /W, the call requests the first and last binary addresses (exclusive) and the block number to write at. Allocate space yourself remembering that there are 200 octal words in each block.

JFQ₁₁ 274000

First ,, Last

Block No.

JFQ ERROR

** Binary Range, Exclusive









** Bit 3.1 = 1 second drum area

** Error Return

** Normal Return (in sequence 70

The block number may be from 0-1400 for the first area and from (1,,0) to (1,,1400) for the second drum position.

C. To Read data written by (B) give call (A) with zero (0) as a name and the same block number as was used to write it.

Mk 4 Spatted Address in Oct. Thousnds	Memory Used: S-Pur T-Red U-Grn V-Blk	Spatted Blks, Not inclusive, in Octal Thousands	Actual - i.e. Non-Spat Block Addresses, in Octal thou, not inclusive.	Comment
0		0 - 124	0 - 124	S Memory - This part Not re-allocated. Same as when spat is off.
40				
60				
100				
120				
124		124 - 130	234 - 240	T Memory
130		130 - 160	240 - 270	U Memory
140				
160		160 - 174	220 - 234	T Memory
174		174 - 200	270 - 274	U Memory
200		200 - 220	200 - 220	T Memory (Not re-allocated)
220				
240				
260		274 - 300	274 - 300	U Memory (Not re-allocated)
274				
300		300 - 320	300 - 320	VTU Memory (Not yet in.)
320				

TO: TX-2 Distribution List
FROM: J. M. Frankovich
SUBJECT: CHANGES TO SKX INSTRUCTION

25 January 1965

INTRODUCTION

Although SKX is the most powerful single TX-2 instruction for manipulating index registers, it suffers from several shortcomings which limit its use. An attempt to improve the instruction will be made with some TX-2 modifications on Monday, 25 Jan.

The changes to SKX are indeed changes, and not additions. Frequency studies of instruction use by Larry Roberts and attestments by other TX-2 users indicate that no known TX-2 programmers use the variations of SKX which are being changed.

A. Zeros

A skip SKX instruction is not supposed to alter the content of the index register. Nevertheless, zero index registers would frequently be complemented. Although this zero changing followed a well-defined rule, it was too complex to be useful to the ordinary programmer. This situation is now changed so that all skip SKX's do not disturb the index register in any way.

B. Raise Flag

No observed programmer both raises a flag and skips with the same SKX instruction. The instruction is now changed so that this is impossible. That is, the instruction -

cf_{SKX_jy}

will raise flag j only when bit $N_{4.7}$ (also called cf_4) is a one, and bit $N_{4.6}$ is a zero. You can no longer request skipping and a flag raise together. This opens up four codes - identified by these two bits ($N_{4.7}$ and $N_{4.6}$). See the chart on the next page.

C. New SKX Skip Instructions

Two of the new SKX variations are left subject to future redefinition. The current actions are indicated in the attached table, but they are subject to change at any time.

The two new skip SKX's are for $cf = 16$ and 17 (with or without dismiss), (i.e., Bits $N_{4.7}$ to $N_{4.4}$ set to 1110 and 1111.) and the actions taken are given in the table. These new variations give a complete set of "greater than" and "less than" SKX skips. Note that in all cases the skip occurs when the strict inequality situation occurs.

New SKX Variations
(25 January 1965)

M; Abbr.	cf	Function	Comment
SKX REX SEX	0	$r \rightarrow X_j$	Set Index
¹ SKX	1	$-r \rightarrow X_j$	Set Index to Negative of
INX	2	$r+x_j \rightarrow X_j$	Increase Index
DEX	3	$-r+x_j \rightarrow X_j$	Decrease Index
SKD	4	Skip if: $r \neq x_j$	Skip if X differs
⁵ SKX	5	Skip if: $-r \neq x_j$	Skip if X differs from -r
SXL	6	Skip if: $x_j < r$	Skip if X is less
SXG	7	Skip if $x_j > -r$	Skip if X is greater than -r
RXF	10	$r \rightarrow X_j, rf$	Set Index and Raise Flag
-	10 - 13	-	Same as 0 - 3 with raise flag added.
-	14,15	-	Undefined - No effect on X_j
¹⁶ SKX	16	Skip if: $x_j > r$	Skip if x_j is greater than r
¹⁷ SKX	17	Skip if $x_j < -r$	Skip if x_j is less than -r

Terminology: r is the final address syllable after all deferring.

X_j is the specified index - x_j its contents.

Note: The restrictions imposed by overflow considerations (wrap-around of the interval) are still imposed.

2 February 1965

To: Apex et al.
From: Alx Vanderburgh
Subject: Scope Octal Printout Subroutine

What: We now have a subroutine that will display the contents of about 40 memory registers, and one index register - on Larry's scope (#56). The subroutine returns in Sequence 54 without changing the AE.

Controls: Knobs...

Index	Number of Registers	First Address
q4	q3	q 2,1

Requirements: Four Index Registers - (X1X, X2X, K, and G).
About 300 Memory Registers.
Sequences 54 and 56 - Interval Timer and Larry's Scope.

How to get it:

1. Read it from GUP area. (GUP 557 PRAE)
2. Borrow paper tape from Alx.
Presently located at ALXW = 10000.

Alx

TO: Interested Persons

5 May 1965

FROM: Ellin Foreman

SUBJECT: Distribution List

There has been an up-dated and formal TX-2 Distribution List compiled for the purpose of insuring that no one should be overlooked in the distribution of pertinent information.

It has been decided that a memo should be written to this effect, giving all speculative names -- broken down between "Designers" and "Users" and giving each person the chance to indicate whether he does or does not wish to be included in the distribution of any further information.

If you feel that you should not be included in either list would you please contact me in B-124 and I will delete your name.

NOTE: There are bound to be overlappings of the two lists and we are leaving it up to the authors' discretion as to whose name should be included in both lists.

TX-2 DESIGNERS LIST

A. Bowen	B-149	D. Malpass	B-122
J. Frankovich	B-155	J. Mitchell	B-159
R. Hudson	B-063	C. Norman	B-124
K. Konkle	B-147	L. Roberts	B-121
J. Laynor	B-153	S. Pezaris	B-151
C. Kurys	B-151	W. Sutherland	B-125
C. S. Jan	B-122	A. Vanderburgh	B-121
J. Lynch	B-124	F. Vecchia	B-124
T. Johnson	B-118	O. Wheeler	B-124

Possible Inclusions

R. Butt B-063

A. MacDonald P-053

TX-2 USERS LIST

Group 23

J. Feldman
P. Fergus
C. Forgie
J. Heart
B. Kantrowitz
C. Lo
M. Morrissey
T. Stockham

To be distributed by
Ellin Foreman B-124

Group 25

J. Forgie
M. Groves
B. Harris
L. Klem
L. Martin
C. McElwain
R. Mitchell
C. Stewardson
A. Stowe
B. Thiele
M. Wendell
R. Weisen
D. Yntema

To be distributed by
Anne Green B-267

F. Belvin B-291
L. Fleck C-390
P. Fleck C-392
B. Gold B-330
C. Rader B-330

Possible Inclusions

J. Frankovich
T. Johnson
C. S. Lin
D. Malpass
L. Roberts
B. Sutherland
A. Vanderburgh
O. Wheeler