B. GRAD File
(Selden)
1401, PP

1 Oct. 61
Contents

1401 T. P.

Implementation Manual

This document follows the organization of T. P. Reference Manual, but discusses material from the inside out. A numbering scheme is established of the following form:

NN	for sections
.	decimal point
N..N	room for insertion
A	the final character will be alphabetic to indicate replacement

WS:go

T. P.
June 27, 1961

10 Basic Language Elements

11 Character set: H, 48 symbols (including blank b)

ABCDEFGHI
JKLMNOPQR
STUVWXYZ0
123456789
+-*/(),.$
= '

12 Standard name:

the set:		is represented by
Numbers (0-9)	:	N } M
Letters (A-Z)	:	L }

$LM_1M_2M_3M_4M_5$

1 - 6 characters of which the first must be a letter and each of the remainder either a letter or a number.

Must be left justified in fields where they appear.

Standard names are delimited by position or separated by blanks from surrounding text.

The name field may be left blank if the systems engineer chooses. Thus six blanks, while not a legal name, are valid entries in the name field for all data types. Note that such items cannot be referenced, and six blanks is not a valid operand. It is impossible to "fill in" such names by programming. *

Names must be unique; they cannot be repeated in more than one record. *

21 Literals

Numeric:

the set:		is represented by
numbers (0-9)	:	N
signs (+,-)	:	S
decimal point (.)	:	D

Signed:

S N_1 N_2... N_{12} DN_{13}...N_{18}	Signed number
S N_1 N_2... N_{12}	Signed integer
S D N_1.... N_6	Signed decimal
Unsigned: N_1 N_2 .... N_{12}DN_{13}...N_{18}	Unsigned number
N_1 N_2 ... N_{12}	Unsigned integer
D N_1 ... N_6	Unsigned decimal

Unsigned numeric literals are assumed positive.

Alphameric:

numbers (0-9)	: N
letters (A-Z)	: L
symbols)(.,-+*/#$blank character	
set H less'	: Y

Any combination of NLY in any order delimited by quote marks. In the character set available quotes are represented by'.

Hence ' N...L...Y...' is an alphameric literal. It cannot be used as a numeric value.

Length:

Literals in logic tables may not exceed six characters not counting the ' with alphameric, nor the S and D in numeric.

Literals in data tables may not exceed 999 characters with the same exclusions.

Literals used in arithmetic may not exceed 12 integer and~~/or~~ six decimal positions.

22) Constants:

A constant is a named operand whose value is assigned in the Data Description and cannot be changed by the Procedure Description. Hence a constant is a fixed alue operand. A constant name may be used to represent any literal value; a constant can represent either a numeric or alphameric literal. A constant has a standard name.

In the Data Description a constant is defined by the following information:

Card Col	Field Name	Possible Contents
1- 3	Table no.	000
4- 6	Line no.	nnn
7- 8	Column no.	nn* (00 except for continuation)
9	Operand class	c
10-15	Name	Standard name (see p. 10)
16-18	Integer length	000-012 for numeric; 001-999 for alphameric
19-20	Decimal length	00-06 for numeric; 00 for alphameric
21	Value type	A, for alphameric, N for numeric
22	Continuation	blank, or C*
23-72	Value	numeric or alphameric literal (see p. 21)

on length for numeric - do not count sign or decimal point, only numbers.
on length for alphameric - do not count quote marks, but do count all other characters including blanks.

* Continuation occurs when the value of the constant exceeds 48 characters in length. This is indicated by C in column 22 of all cards but the last, and consecutive ascending numbers in column 7-8, beginning with 00. The ' will appear in column 23 of the first card and the appropriate column of the last card. It will also appear in columns 23 and 72 of any intermediate cards. Columns 10-22 of all cards after the first will be ignored and should be left blank to reduce confusion.

The length will be determined by the number of characters between the ' and the count checked with the integer and decimal fields. If there is a discrepancy, this will be noted as an error.

23) Variables:

Variables are the key operands in any information processing system. It is their value determination that is the purpose of the process. Variables may be inputs, intermediates or outputs depending upon how their value is developed. The same variable may serve as all three.

There are three types of variables, each of which serves a particular function:

unassigned, preassigned, and masked.

Unassigned variables are used when the value is developed through an input or program synthesis. Preassigned variables give the opportunity to introduce common constants, and set initial values. Masked variables are used for outputs of numeric data to provide for zero suppression, dollar amount formation, and other editing.

Col.	Field Name	Possible Contents		
		Unassigned	Preassigned	Masked
1- 3	Table No.	000	000	000
4- 6	Line No.	nnn	nnn	nnn
7- 8	Column No.	00	nn	00
9	Operand Class	v	v	v
10-15	Name: standard	yes	yes	yes
16-18	Integer length:	numeric alphameric 000-012,001-999	numeric alphameric 000-012,001-999	alphameric or 001-48
19-20	Decimal length:	00-06, 00	00-06, 00	00
21	Value type:	N , A	N , A	M
22	Continuation:	blank	blank or C	blank
23-72	Value	blank	numeric alphameric literal literal	alphameric literal using special masking symbols only

Names must be unique; they cannot be repeated in more than one record. A masked variable may not have a preassigned value.

Continuation occurs when the value of the variable exceeds 48 characters in length. This is indicated by a c in column 22 of all cards but the last, and consecutive ascending numbers in columns 7-8 (starting with 00). The ' will appear in column 23 of the first card and the appropriate column of the last card, and columns 22 and 72 of any intermediate cards. Columns 10-22 of all cards after the first will be ignored and should be blank to reduce confusion.

In the case of unassigned variables, the length will be determined strictly by the integer and decimal fields. In the case of preassigned variables the length will be determined by the number of characters between the ' and the count checked with the integer and decimal fields. If there is a discrepancy this will be noted as an error.

Masks - Department of fuller explanation. A masked variable has no preassigned value. It is treated as an alphabetic field of length sufficient to include the entire mask. The mask will be loaded into the printout area. (Should there be any word marks there, they will be wiped out. This could occur only in the case of improper use of overlapping.) Subsequent to this, some quantity which will be assumed to be numeric and of proper length will be placed therein by the use of Move and Edit command. It is assumed that the programmer will properly adjust for all decimal points. As Autopoint will not be effective with respect to masks, this technique offers an opportunity for scaling printed results. The exact editing possible will depend upon special features and RPQ's ordered for the object machine, and processor will make no checks. As masks cannot be continued, they have a maximum length of 48 characters.

I. General

This material is directed to the processor rather than the language. This section will redefine various parts of the language as their translators are written, and is source material for deeper understanding of T. P.

II. Processing T. P. will be a three stage operation:

T. P. Processor	Autocoder	T. P. Object Machine

The T. P source language will be translated to legal Autocoder language which will be assembled. The resultant deck will become the T. P. object program.

Machine specification: See Section 90

III Data:

Names of all items will be entered in the symbol table, together with type, length, decimal position, etc. If the item in the symbol table is longer than eight digits, the item will be placed in consecutive disc locations and the disc address only will be put in the table.

Variables become DCW's with the literal remaining in place for alpha and inserted for numeric literals. Sign and decimal point are cleaned up .

The Autocoder cards are punched immediately. When overlapped, the last defined record is established.

Constants are treated identically except that the name always goes into symbol table, and the statement goes into disc instead of punching.

Expression names are in symbol table; the compiled routine which will evaluate the expression goes on disc.

Record names go in symbol table with count of number of fields and routine to set all work marks as required by SETUP as operator. Note that in the not overlapped case SETUP is redundant and string need not be put on disc; but, it suffices to make a count of the number of fields (for move record operations).

Working storage is treated as record areas.

Files are compiled into routines to read and write, as the case may be, and corresponding cards punched.

IV. A point of order:

Expressions must follow definitions of all names used in them and logic tables follow definition of all names used in them. We will specify the easier rule variables and constants first, then expressions, then tables. A two pass approach could free up this restriction, but does not seem worth while.

It is illegal to put constants or expressions between variables within a single record or between records in one file, it is possible to put constants between files.

Please note the usage of reassigned variables.

Tables are converted into rules containing the tests and actions specified for each rule (column). See section 87.1 for details.

The comments associated with each table header are listed to produce a table of contents of the program.

Object program tables will be assigned to disk in consecutive locations as encountered. The Autocoder will through EQU statements handle table addresses.

24) Expressions

A combination of allowed operators and operands which define a value dynamically, e.g., an expression is evaluated at the time of its use. As of now only certain arithmetic expressions are permitted. Generally, expressions are formed following FORTRAN rules.

operands:	positive numeric literals (no unary minus) numeric constants numeric variables	
operators:	+ , -	low
	* , /	high
	* *	highest
delimiters	- ( )	

An expression consists of operands, operators and delimiters properly sequenced by the expression formation rules. Operands and operators alternate with an operand in both initial and final position. There must be an equal number of left and right parentheses.

Blanks within an expression (or even within words) are ignored as in FORTRAN. Teaching manuals will prescribe usages for blanks. Words must not be broken at the end of a card, rather insert blanks.

* * is allowed in the scan, will cause transfer to a non-existant routine; at some later date a routine may be added.

No single expression may contain more than 26 operands.

Expressions are entered in T. P. as follows:

Columns	Field Name	Value
1-3	table number	000
4-6	line number	nnn
7-8	column number	nn
9	operand class	E
10-15	name	standard name
16-18	integer length	000-012 for result value
19-20	decimal length	00-06
21	value type	N
22	continuation	blank, C
23	value	actual expression

Continuation occurs when the expression will not fit on one card. This is indicated by C in column 22 of all cards but the last, and consecutive ascending numbers in columns 7-8 (starting with 00). The expression should be written (leaving blanks) as necessary to avoid breaking names or literals.

The conversion of expressions into machine code to carry out the function takes place in three stages.

I. PRESCAN

The source statements are converted to the form A⊙B⊙⊙C⊙D.... where ⊙ is any actual operator including parentheses and the A, B, C are literal values. Also, a table is produced equating A, B, C, etc., with the name or literal.

II. SCAN

An extran scan reduces the expression to a properly ordered list of three address operations coupled with calls to the proper sub-routines. A label lookup replaces the single with three character names.

III. POST SCAN

The Post Scan program takes the properly ordered list produced by Scan and produces a series of T. P. language macro-instructions. These are placed on the disk.

IV. MACRO-INSTRUCTION GENERATOR

When it discovers that a name is an expression name, looks up the macro-instructions on the disk and punches out the 1401 Autocoder instructions to evaluate the expression.

NOTE: Certain ways in which expressions in T. P. differ from Fortran:

1) There are no functions (use logic tables).
2) There is no exponentiation (**) operation.
3) All operations assume autopoint. Neither integer nor floating point operations are automatic. Floating point is impossible and integer is possible only if the values are defined as integer.
4) Results are truncated, not rounded. Rounding may be added later, as it was specified in the manual.
5) There is no unary minus legal in T. P.
6) The sequence A(B&C) is illegal, use A*(B&C). This restriction should be removed when we have time.

10/3/61
WS:np

FILES

A file is a referenced collection of records all on a single storage device. A file may have from 1 to 9 records (and hence the same number of formats.) The purpose of a file is to associate an area in high-speed data memory (core) with a particular file medium, one or more types of records, and the variables contained in each record type. The system engineer defines a file as follows:

Card-Column	Field Name	Possible Contents
1-3	Table No.	000
4-6	Line No.	nnn
7-8	Column No.	00
9	Operand class	F
10-15	Name	Standard name.
16-18	Integer length	000-999; the maximum length of the longest record in this file - 000 represents 1000. In the case of RAMAC files the length must be 200 or 000 in the move mode and 176 or 880 in the load mode.
19-20	Decimal Length	Ib for input only. b0 for output only. I0 for input and output.
21	Value type	L for load mode; M for move mode.
22	Continuation	Blank
23	Description	1-9 (number of record types.)
24	Description cont.	
25-	Description cont.	(Medium name which must be.) CARD, n for reader or punch; n is the stacker number 1,2, 4, or 8. The , and the n are both omitted if the feature is not used. PAGE for printer MAGTAPE for tape MEMORY for core (working storage)

Card Column	Field Name	Possible Contents
		RAMAC, nnnnn-nnnnn for 1405 disk storage. The nnnnn-nnnnn indicates the lower and upper bounds of the RAMAC region occupied by this file.

Record descriptions for the records in a file must follow the file. Enough memory is assigned to each file to accommodate the longest record in it. All internal data areas (all variables) including those for working storage of intermediate values must be within MEMORY files. Any file whose type is CARD or PAGE must have the value type M. If description is CARD integer length must be 080. If description is PAGE integer length must be less than 133. For MAGTAPE with Load mode the integer length is the actual number of data characters (not including work marks.)

W. Selden

WS/nbr

1401 T.P.
14 July 1961
Section 26

RECORDS

A record is a collection of variable fields which is useful to consider as an entity. Each record contains a definite number of fields. If a file contains more than one record they are overlapped; that is to say only one record may be present at one time in the file area. The system engineer defines records as follows:

Card Column	Field Name	Possible Contents
1-3	Table No.	000
4-6	Line Number	nnn
7-8	Column No.	00
9	Operand class	R
10-15	Name	Standard Name
16-18	Integer Length	000-999, number of characters per record. 000 represent 1000. 01-99 number of variables per record.
19-20	Decimal Length	
21-		Blank

A record must be followed by the description of the variables in the record. Every variable must be a member of some record. (Note that it is not necessary to define any internal variables if the variables are not going to be addressed. Constants and expressions will not be defined to be within the record no matter where placed by the programmer.)

The description of a Record in a File must follow the definition of the File and precede the definition of any other File. The definition of each record is immediately followed by the definition of the variables within that record. Note that if the system engineer does not wish to refer to any of the variables in a particular record he need not define them, but if he defines any variables within a record he must define all of them.

Constants and expressions will not be defined to be within a record no matter where placed by the programmer. Note, however, that preassigned variables will fall within the record.

The GO and DO operators refer to a table rather than a unit of data. The Table is deemed to be defined by the appearance of the logic table. The form of this operand is:

TABnnn (where nnn is any number 000 through 999.)

001

GF/DO cont.
"Any open procedure table may GO TO itself."

Read and Write stand as defined in the language definition memorandum dated 6/25/61, page 19 and 20.
Except: Page 19, seven lines from bottom replace "CORE" by "MEMORY"

(3) CLOSE - an operation which rewinds magnetic tape units and writes tape marks if required.

form - CLOSE file-name.

analysis The input magnetic tape unit referenced in the data description for that file name is rewound. A tape mark is written on the output tape referenced in the data description and it is rewound.

RAMADD

USE

The use of an assignment operator SET, MOVE, SET - etc., placing data in RAMADD will cause a seek to that address. RAMADD must not be defined by the System Engineer, but will be defined by the T. P. system. RAMADD will be a five character field; 2 digit face, 2 digit track and 1 digit sector number. Arm and Unit will be capable of being changed by system control cards. All data in one T. P. problem will have to be on one unit and accessed by one arm, all tables will be on one unit and accessed by one arm, but either the unit or the arm or both may be different for data and tables.

MECHANIZATION

RAMADD will exist within a 9 character field ending in a Work mark, Group mark character. The name of the high order position of the field will be RAMAD$. Thus Processor System Housekeeping will punch EQU cards defining RAMADD and RAMAD$, will place an initial entry in the symbol table establishing RAMADD as a 5 integer numeric. Macro Generator will punch a SD to RAMAC$ following each assignment into RAMADD, Object System Housekeeping will establish the initial values for the remaining digit positions of RAMAD$.

W. Selden/jb

GO/DO - Section 35

GO and DO stand as defined in the language definition memorandum dated as 6/25/61, page 27.

Insert an additional sentence in the analysis of GO TO.

RAMADD

The digits are interpreted:

Disk Face	Track	Sector
nn	nn	n

Special Actions Section 36.

SPACE remains as defined in the language definition memorandum dated 6/25/61, page 28.

1401 T. P.
14 July 1961
Section 60

RULE FOR PROCESSING LOGIC TABLES

1) Scan the condition stubs to locate limited entries. Replace each limited entry in a given row with a BCE to a character in a list. (Equal compare to a 1 for satisfied and 0 for not satisfied in the stub under the Y condition, reverse the logic for N.) Nothing for blank (NP).

2) Define a list of characters which will be tested by the limited entries.

3) Produce the macroinstructions which will make the tests specified in the limited entry stubs and store the results (1 for satisfied and 0 for not satisfied) in the list produced in 2. Write macroinstructions on Ram.

4) At this point, space consideration may dictate storing the table column wise on the ram retaining in memory only the stubs and one column.

5) Reform the extended entries so the stubs are consumed into the boxes ie. each column will stand on its own feet.

6) Copy out each box in the form specified for input to the macroinstruction generator.

7) The first entry at the top of each column will be supplied a generated label.

8) Each condition will be followed by a conditional branch to the head of the next column, except that the tests in the last column will branch to error table routine. See table on page 3 of "T. P. Implementation."

9) For these and other purposes generated labels may be of the form Lnnnn$, where L is the letter indicative of the phase A for 1 B for 2 etc. nnnn is generated $ is $ (a rose is a rose).

10) Notice that at object time in a given table as soon as the first action is encountered there will be no more condition and no more columns considered.

11) If a table GOES or does itself it will be necessary to re-evaluate all formulas and re-assess the limited entry conditions.

W. Selden
Project Co-ordinator (Pro Tem)

WS/pm

B Q 72.1

JUL 20 Rec'd

Section 72.1
Logic Table Header

1ST CARD

Column	Name	Contents
1-3	table header	3 numeric digits
4-6	line number	000
7-8	column number	00
9-13	identification	TABLE
14-15	conditions	2 digit number of condition rows
16-17	actions	2 digit number of action rows
18-19	rules	2 digit number of entry colums
20-22	next table	3 digit number of table to go to next
23-26	Error table	3 digit number of table to go to in case of error
27-46	Analyst	name of analyst
47-64	date	date
65-72	Systems name	name of system
73-80	Identification	identifications of program

2ND CARD

1-3	table header	3 numeric digits
4-6	line number	000
7-8	column number	01
9-13	identification	ORDER
14-72	column numbers	n n, n n, n n, etc. column numbers in the order to execute
73-80	Identification	Identification of program

SUBSEQUENT

ORDER cards may be used, each differing from the previous in having consequetive higher sequence number.

COMMENT CARD

1-3	table header	3 numeric digits
4-6	line number	000
7-8	col. number	2 numeric digits
9-13	NOTE 5	NOTE
14-72		Comments

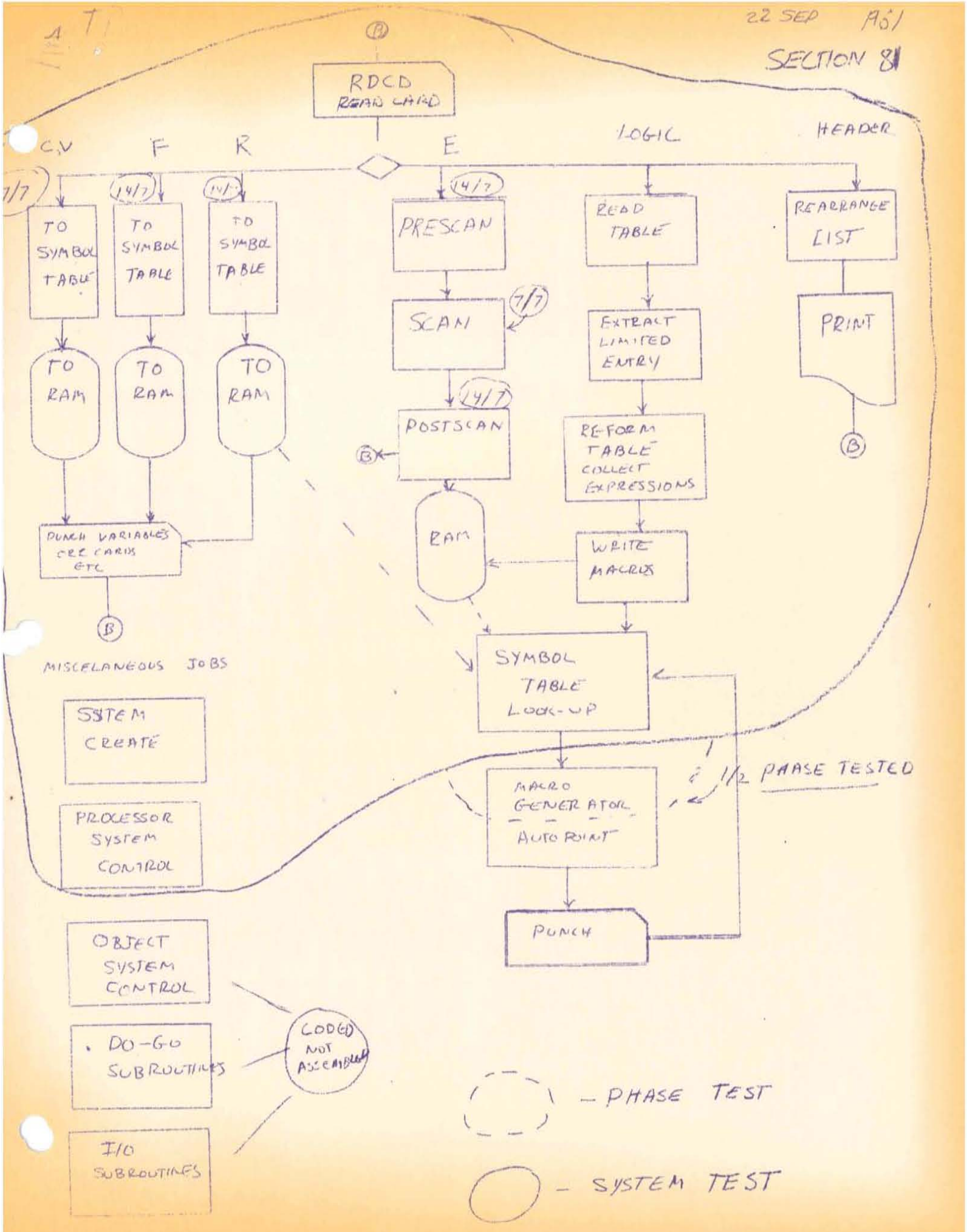
Subsequent cards are identical except ascending numbers are punched in columns 7, 8.

Table headers will be printed as a table of contents of the job.

dtw
7/12/61

22 SEP A51

SECTION 8



SYMBOL TABLE

A) Item Description:

If Item Was:	Col 1-5	6-8	9-10	11-16
Numeric Constant	Ramac Address	Integer	Decimal	Symbol
Alpha Constant	Ramac Address	Integer	bb	Symbol
Numeric Variable	bbbbb	Integer	Decimal	Symbol
Alpha Variable	bbbbb	Integer	bb	Symbol
Expression	bbbbb	Integer	Decimal (signed plus)	
Mask	Symbol	Integer	**	Symbol
Record	Ramac Address	Integer	##	Symbol
File	Symbol	Integer	□ □	Symbol
"RAMADD"	bbbbb	005	$$	RAMADD

B) Table Description:

The symbol table contains 55 of these 16 character items. It is followed by a Word Mark and Group Mark character in position 881. It is preceded by either one of two things:

8 digit Ramac address and WDMGM character
6 character symbol or lookup preceded by two workmarks.

The symbol table normally starts in location 1240, however this may change. The symbol table is defined in the packet furnished, and this definition should be used if the table is to be in standard position.

C) Using the Symbol Table:

The symbol table is filled from the low order (right hand) location first working backward. Index register 1 is reserved during the data division to indicate the location of the next available space in the symbol table. The area in variable storage named SYMRAM contains the address of the next ram track for the symbol table, that is to say it contains the number of tracks already written.

To write the symbol table the Ramac Address should be placed in front of

SYMBOL TABLE

A) Item Description:

If Item Was:	Col 1-5	6-8	9-10	11-16
Numeric Constant	Ramac Address	Integer	Decimal	Symbol
Alpha Constant	Ramac Address	Integer	bb	Symbol
Numeric Variable	bbbbb	Integer	Decimal	Symbol
Alpha Variable	bbbbb	Integer	bb	Symbol
Mask	Symbol	Integer	**	Symbol
Record	Ramac Address	Integer	##	Symbol
File	Symbol	Integer		Symbol

B) Table Description:

The symbol table contains 55 of these 16 character items. It is followed by a Word Mark and Group Mark character in position 881. It is preceded by either one of two things:

8 digit Ramac address and WDMGM character
6 character symbol or lookup preceded by two workmarks.

The symbol table normally starts in location 1240, however this may change. The symbol table is defined in the packet furnished, and this definition should be used if the table is to be in standard position.

C) Using the Symbol Table:

The symbol table is filled from the low order (right hand) location first working backward. Index register 1 is reserved during the data division to indicate the location of the next available space in the symbol table. The area in variable storage named SYMRAM contains the address of the next ram track for the symbol table, that is to say it contains the number of tracks already written.

To write the symbol table the Ramac Address should be placed in front of

the table and the table written, SYMRAM incremented.

To read the symbol table proceed as above for write using Read command.

To search the symbol table (serially) place the symbol being looked for directly left of the table and branch to SEARCH.

WS/pm

replaces same section dated 18 July

CONVENTIONS:

The following conventions apply to the T. P. Processor. Please observe them.

1. No phase may contain a Groupmark Word Mark character. This character will be permanently kept at TOP currently 3999.

2. There will be a packet of cards which should be added to each phase as it is assembled. The basic function of this packet will be to provide communication between the phase as written and the operation of System Creation and System Control.

This packet must be used. To repeat it is strictly forbidden to refer to any portion of core outside your phase by absolute addresses. The obvious reason for this rule is that when the system control program is modified in some way it will be merely necessary to alter the packet containing EQU cards and reassemble all phases rather than having to closely examine and recheck each phase.

3. The beginning point of each phase is the first character thereof.

4. There will be a communication region defined where information required by more than one phase will be placed. These items will be defined in the packet menioned above.

5. The phases cannot assume that storage will be cleared prior to their operation. If clear storage is required positively ensure it by use of DCW or dynamically by execution during processor housekeeping of Clear Storage command.

6. During the phases dealing with the data division Index register 1 is reserved for the location of the next available space in the symbol table.

GRAD

PROCESSOR INPUT/OUTPUT

Use of the standard package and system control will make available a RAMAC reading/writing and checking routine. The programmer will need only use a Read or a Write command and a Branch to RAMAC. The system will seek the address and, in the case of writing commands, issue a write check. The routine will reset the nop and will reseek and repeat the operation in the case of error.

Restrictions

The RAMAC operation specified in line will not be executed there but will be moved into the routine, therefore it is always necessary that the Branch to RAMAC immediately preceed the I/O operation. The B address of the I/O operation will be the high order position of the disk record address preceeding the I/O area in core.

Upon successful completion of the specified operation control will be returned to the next instruction following the I/O operation.

EXAMPLES

```
B       RAMAC
WD      label

B       RAMAC
RDTW    label
```

The permissible operations are: RD, RDT, RDW, RDTW
WD, WDT, WDW, WDTW

The Write check is illegal (it is performed by the routine) and its use by the programmer will gum up the works.

replaces section 84.1 and 84.2 of 18 Jul

Operation of System Control, Housekeeping, etc.

The program identified as above comprises several quite distinct sub programs.

Package The first few positions (currently 1 060 to 2 160) consist of the package which is present in all phases, controls certain system functions, and ties the system together by EQU statements. While all phases will show DCW's in the listing of their section the processor will employ the constants assigned in System Control Program. This will permit independent phase testing using different values in certain instances but should be watched for a possible source of error.

RAMAC The second item (currently 000 through 480) is RAMAC a generalized RAMAC handling routine explained in Section 83. 2.

SYSCTL The third section (currently 10 010 through 10 999) accepts certain control information and reads in a phase consisting of one or more 880 character tracks and branches to the beginning of the phase. SYSCTL uses RAMAC.

RDCD The fourth section could be considered a part of each of the phases in the data division because it reads in the data division cards (after the header) and, if necessary, reads in the proper phase. It is overlapped by the logic division program.

System Housekeeping The fifth section (currently 13 010 through 13 060 located at 1175) sets Index registers and branches to begin processing. This section is overlapped after it is executed once.

System Creation The sixth section (currently 1 05 through 3 20) is the system creation program which writes phases on the RAMAC.

Primer Mention will be made of Primer. This is a separate program which is frequently shown as the last sheet of the listing for System Control and Creation, etc. Primer is the item identified on the system flow chart as "process card". Its function is to "prime the pump", i. e. read the system control program into core from the RAMAC and branch to it.

OPERATIONAL DESCRIPTION OF SYSTEM CREATION AND OPERATION

The reader will do well to read carefully Section 84. Material here describes the steps to create and run a processor, including operating instructions and also describing functions of the 1401.

System Deck

A system deck is obtained from the union of all phases as they come from Autocoder. The first must be the System Control, Creation, Housekeeping, etc. phase and the last be the Primer. Otherwise order is immaterial, but phases must be kept together. The System must be preceeded by the clear memory and bootstrap cards, the other decks may include these, but need not. The END card will branch To SYSCRE. The first few cards of each phase will contain certain control information which should be checked for consistency, particularly the material loaded into PROFTS which includes the PROcessor Arm, Unit Face Track Sector designations. The track must differ from all other tracks used by the processor. The rest of the numbers except arm must be the same for all phases. Changing values here will change the face required, but the change must be made for all phases in an identical manner.

Place cards in card reader and push card load button. Memory will be cleared and System creation et al read in. The END card of System creation will branch to write the first phase, (System Control, etc.) on the RAMAC. Detailed procedure follows:

> The value in PROFTS determines the RAMAC Address used for the first track for this phase. The value in NOTRAX is the number of tracks required for this phase and the high order of the 8 character address preceeding the first character of the phase is in PHASAD. All of these values should be established by DCW's. The writing procedure places word mark group mark characters where needed and places the eight character track address addresses where needed. It also conserves the characters within the program that were destroyed by these items. However, other characters outside the phase itself may be destroyed.

POSITION

System Control uses RAMAC to write the phase (one track at a time) and then branches to read more cards. It skips over cards serial 001, 002, and 003 if present and uses the normal loading procedure for the rest.

Subsequent phases are written using the new values loaded into NOTRAX, PHASAD, and PROFTS by these programs.

Primer is used to start processing the first program or to restart processing subsequent programs so long as T. P. remains undisturbed on the RAMAC. It contains the normal clear storage and bootstrap cards and reads in the System Control Program from track 00. The unit and face must agree with the similar assignment in writing the program on RAMAC. Arm need not be the same but normally will be. Primer contains its own Read and Check (the RAMAC routine will not be in core at this time as Primer clears storage).

A glance at the listing will show that the values initially assigned to PHASCT and TRACK will read in the System Housekeeping phase (this feature not yet installed.) In addition, LOCPHA (X3) contains the address of the high ORDER (left hand) character of the 8 character RAMAC address preceeding the phase. This reduces the requirements on Primer and the housekeeping portion of this program. As these are the values that will exist initially, each Phase as it desires to read in another phase must dynamically (not through DCW's) put the correct values in place. Note particularly that this is directly opposed to the procedure for the control of creation where the values must be placed by DCW's and may not be placed dynamically.

An example of this feature may be found in the instructions starting at SYX and extending to but not through CLEAR1 (currently 12 020 through 12 070) in System Creation et al phase. The clearing of memory is not in general necessary, but is needed if the phase being replaced might create word mark group mark characters inside itself.

The process which reads phases from the RAMAC restores any characters displaced within a phase but it will leave a word mark top and bottom of the phase which may interfere with reading phases into other locations in this program.

SECTION 87

NOTE: In Source Languages
Stubs 26 Characters
Entries 10 Characters

Rule For Size In Source Languages:

(26 + 10* Nc) Nr < limit (2000 characters at present)

Where Nc is number of entry columns.
Nr is number of entry rows.

Logic tables will be read. Column transformation will be honored and the entries moved into the proper column and row.

Rule For Rearrangement:

1. Check row and column count for too great size by formula above. (As MPY and DIV are not specified) program by series of additions and shifts. If too large, message and reject.

2. Establish list showing machine address of the location of first stub and first entry in each column; the maximum number of columns is 20.

1-3 loc stub
4-6 loc entry 1
7-9 loc entry 2
10-12 loc entry 3
etc.

3. Rearrange this list according to the new order specified.

4. Move each field from the card to the position indicated by the table, and increment table by 10 (or 26 for stub).

5. Check for too many rows or columns.

WP/pm

B. Gead

JUL 24 Rec'd

1401 T. P.
18 July 1961
Section 87.1

After the Data Description table has been processed, System Control will have read into storage a Logic Table Header. Also as System Control has read into storage a Logic Header, it will then read into storage 'PROGRAM A' which is stored on the RAMAC.

'PROGRAM A' will perform the following:

1) ANALYZE LOGIC TABLE HEADER
2) TEST FOR AN ORDER OR COMMENTS CARDS
3) LOAD LOGIC TABLE INTO STORAGE
4) BREAK THE LOGIC TABLE INTO ITS RULES; WRITE EACH RULE ON THE RAMAC *
5) RETURN TO SYSTEM CONTROL

After 'PROGRAM A' has been executed, System Control will then read into storage 'PROGRAM B'.

'PROGRAM B' will perform the following:

1) READ INTO STORAGE EACH RULE, AND PROCESS EACH RULE. OUTPUT AS INPUT TO MGEN.
2) RETURN TO SYSTEM CONTROL.'.

* This will also change limited entry to extended entry.

WS:pm

1401 TP
Ref Man.

1401 T. P.
30 August 1961
Section 90

TO: T. P. List

SUBJECT: Machine Definition

Looking at the orders, it is clear most machines will have advanced programming and high-low-equal compare, about half will be 4K and half larger. Multiply/Divide will be rare. Considering all data, I declare the following machine requirements:

	Processor Step 3	Autoloder Step 4	Object Step 6
Core Size	4K	4K	4K
Additional core used if available	NO	YES	YES
1402	1	1	1
1403	1	1	3
1405	1	0	1
one arm	1	0	1
2nd arm	2	0	2
Tape	0	4	3
Compare feature	1	1	1
Advanced programming	1	1	1
MPY/DIV	0	0	3

legend
0 = not used, may be present
1 = required
2 = used if present, optional
3 = used if required by object program
4 = four tapes required

Earl has suggested that the initial overtures to establish experiments explore the customers with 8K or larger systems. Our initial system will work on a 4K machine, but until the mechanisms for overlapping and logic table break-up are complete the larger memory sizes will permit more attractive sample programs.

17 July 1961
Section 90

TO: T. P. List

SUBJECT: Machine Definition

You have seen the best available information concerning the 1401 Ramac systems to be delivered this year. While not complete nor 100% accurate, we have no choice but to come to a decision at this time.

Clearly most machines will have advanced programming and high-low-equal compare, about half will be 4K and half larger. Multiply/Divide will be rare. Considering all data, I declare the following machine requirements:

	Processor	Autocoder	Object
Core Size	4K or more	4K or more	4K or more
1402	yes	yes	yes
1403	yes	yes	yes
1405	yes	not used	yes
Arms	one or more	not used	one or more
Tape Units	not used	4	*
Compare	yes	yes	yes
Advanced Programming	yes	yes	yes
MPY/DIV	no	no	*

* Means this feature is required if used in source language. It is always possible for the system engineer to program multiplication and division using one or more tables.

This decision is an important one, and will effect T. P. in a number of ways. I request that any disagreements or suggestions be communicated to me.

Earl has suggested that the initial overtures to establish experiments explore the customers with 8K or larger systems. Our initial system will work on a 4K machine, but until the mechanisms for overlapping and logic table break-up are complete the larger memory sizes will permit more attractive sample programs. I heartily endorse this suggestion.

W. Selden

W. Selden
Project Co-ordinator (Pro-Tem)

WS/pm

1401 T.P.
6 Sept. 61
Section 91

System Control

Arms. The field ACOUNT contains either a 0 or 1. System housekeeping will add this number to the arms indicated for those system uses which can practically use the second arm. A similar function will use the second arm if available for tables in the object program.

Memory. The field X1TEMP contains the value for the CTL card to be used by the object program. This will control the maximum size of object program that Autocoder will compile.

Tapes The tape Error routine will exist as a Ramac Track. The tape reading routine in T.P. will use the DO mechanism to use this program as though it were a table.

W.Selden/jb

Ramac assignments

1	Face 00	T. P. Processor	(PROFTS)
		Track 00	System Control
		01, 2	Constants and Variables
		03, 4	Files and Records
		05, 6	Pre-Scan and Scan
		07	Post Scan
		08, 9	Logic Table Header and Housekeeping
		10	Processor housekeeping
		13-19	MGEN Control and Punch
			MGEN SET and MOVE
			MGEN Expressions with SET and MOVE
		20, 21	MGEN Input Output and Miscellaneous
		22, 23	MGEN Comparison
		24, 25	MGEN Expressions with comparison
		28	End of Table
		29, 30	Program A
		31, 32	Program B
		33,	Program C
		40, 41	Processor End of Job
		50, 59	Temporary Storage of MGEN
0	Face 01	Symbol Table	(SYMFTS)
0	Face 02	Constants	(NEWRAM)
1	Face 04	Output of Expressions	(MACFTS)
0	Face 05	Rules from Program "C" Rules	(RULFST)
1	Face 02	MACRO instruction output of C	MGNFTS)

W. Selden/ap

Ramac assignments.

1	Face	00	T. P. Processor	(PROFTS)
			Track 00	System Control
			01,2	Constants and Variables
			03,4	Files and Records
			05,6	Pre-Scan and Scan
			07	Post Scan
			08,9	Logic Table Header and Housekeeping
			10	Processor housekeeping
			15	MGEN Control and Punch
			16,7	MGEN SET and MOVE
			18,9	MGEN Expressions with SET and MOVE
			20,21	MGEN Input Output and Miscellaneous
			22,23	MGEN Comparison
			24,25	MGEN Expressions with comparison
			26,7,8	Program A
			29,30,1	Program B
			32,33	Program C
			40,41	Processor End of Job
			90,99	Temporary Storage for MGEN
0	Face	01	Symbol Table	(SYMFTS)
0	Face	02	Constants	(NEWRAM)
1	Face	03,04	Working Storage for Logic Tables	(TABFST)
0	Face	05	Input to MGEN from Program "C"	
1	Face	06,10	Output of Postscan	(MACFTS)

W Selden/jb

File NOITP Selden Reference

1401 I.P.
September 15, 1961
Section 91.2

CORE ALLOCATION

In the near future we may have to reallocate core to allow for LAYOUT. If this occurs the result will be

DATA DIVISION

ORG	1151		1151
DA	1x8		1151-1158
	Group Mark ˇ		1159
DA	2x880	Phases at 1160	1160-2919
	Group Mark ˇ		2920
DA	1x8		2921-2928
	Group Mark ˇ		2929
DA	1x176	Sector area	2930-3105
	Group Mark ˇ		3106
Not used			3107-3108
DA	1x8	(unchanged)	
ORG	3109		
DA	1x8		3109-3116
	Group Mark ˇ	Symbol Table	3117
DA	1x880		3118-3997
	Group Mark ˇ		3998
	Group Mark ˇ		3999

LOGIC DIVISION

As the logic division is structured so that each phase will go to only one other phase, space will be saved in System Control. We will ORG the phases to 900. If there is urgent need of it, we will consider ORG back to 850, however this will require moving some of System Control to the card read area, and it seems much safer to suggest that this area be used as temporary storage by the phases. If anyone does so please inform Bill Selden.

Change in System Creation

In order to permit utilization of the region above 3117 for program, system creation will be moved, for the logic division to ORG 650. This will be accomplished by the housekeeping phase. This will affect only the System housekeeping phase, the Branch to 3995 will remain in force.

An Important Point to Watch

Ronnie Gale has informed me that he uses the area between 2920 and 3000 as temporary storage, and that he does not use any area above 3000.

1401 T. P.
5 Sept. 1961
Section 93

Object Program system creation and system control.

The functions of creating and controlling an a TP object program are almost exact analogues of the similar functions for the processor. Similar programs will be used with the symbols identical with those for the system functions of the processor except that the sixth (or last) character of the names will be a $. Initially the programs will be the same as the processor, located in the same place etc. Prior to release of the system the object program system functions will be reassembled to utilize space more efficiently.

The Processor housekeeping and logic division housekeeping programs will punch out equals (EQU) cards to provide the locations for the following functions (and others as needed).

GOTO$	Branch here for GO TO
DO$	Branch here for DO
LAYOU$	Branch here for LAYOUT
etc.	

W. Selden/jb

Size Limits

There are a number of items in T.P. for which there is a limit of size, for example 6 character name length; or of quantity, for example there may be no more than 26 operands in an expression. We collect all of these size and quantity limitations here as a handy reference and as limits are most apt to change as T.P. develops with further programming.

The phrase unlimited is to be understood in a relative sense, as assembly and other programming are finite processes an infinite number of anything is impossible; rather we mean by unlimited that if n of anything is operable then there will be no specific bar to n + 1; however, this rule is not recursive.

There is a general limitation of 10 or 20 million characters in the size of RAMAC. As in general, all of any program (except the comments) has to stored on RAMAC, there is an obvious but very large limit on source and object programs.

Labels: 6 characters each: unlimited number.

Short Literals Used In Logic Table:
6 characters not counting the @ in alphabetic nor the . + or - in numeric literals, or restating, a total of 8 characters.

Long Literals Used In Data Table:
Numeric: 12 integer positions plus 6 decimal positions.
Alphabetic: 999 positions.

Data Division Items: Unlimited.

Number of Tables: 999

Size of Table:
This is restricted in several ways. No one of the following restrictions may be exceeded.

Rules:	no more than 13 on the coding sheet (more sheets possible)	
	no more than 30 if rearranged via header	
	no more than 64 in programming of T.P. processor	
Actions:	99	coding sheet
Conditions:	99	coding sheet

Overall restriction on the size of table is expressed in the following table:

Number of Rules	Number of Rows Actions Plus Conditions
1	54
2	42
3	34
4	29
5	25
6	22
7	19
8	17
9	15
10	14
11	13
12	12

Rules	Rows
13	11
14	11
15	10
16 - 17	9
18 - 19	8
20 - 22	7
23 - 25	6
26 - 30	5
31 - 37	4
38 - 47	3
48 - 64	2

Size Limits

There are a number of items in T. P. for which there is a limit of size, for example 6 character name length; or of quantity, for example, there may be no more than 26 operands in an expression. We collect all of these size and quantity limitations here as a handy reference and as limits are most apt to change as T. P. develops with further programming.

The phrase unlimited is to be understood in a relative sense, as assembly and other programming are finite processes, an infinite number of anything is impossible; rather we mean by unlimited that if n of anything is operable, there will be no specific bar to n + 1; however, this rule is not recursive.

There is a general limitation of 10 or 20 million characters in the size of RAMAC. As in general, all of any program (except the comments) has to be stored on RAMAC, there is an obvious but very large limit on source and object programs.

Labels: 6 characters each: unlimited number.

Short Literals Used In Logic Table:
6 characters not counting the @ in alphabetic nor the . +-or - in numeric literals, or restating, a total of 8 characters.

Long Literals Used in Data Table:
Numeric: 12 integer positions plus 6 decimal positions.
Alphabetic: 999 positions.

Data Division Items: Unlimited.

Number of Tables: 999

Size of Table:
This is restricted in several ways. No one of the following restrictions may be exceeded.

Rules:	no more than 13 on the coding sheet (more sheets possible)	
	no more than 30 if rearranged via header	
	no more than 64 in programming of T. P. processor	
Actions:	99	coding sheet
Conditions:	99	coding sheet

Overall restriction on the size of table is expressed in the following table:

Number of Rules	Number of Rows Actions Plus Conditions	Rules	Rows
1	54	13	11
2	42	14	11
3	34	15	10
4	29	16-17	9
5	25	18-19	8
6	22	20-22	7
7	19	23-25	6
8	17	26-30	5
9	15	31-37	4
10	14	38-47	3
11	13	48-64	2
12	12		

DO

There is a limit of 10 DO's in a nest at one time. There is no limit of the total number of DO's in a program. A table may appear at any number of level in a nest; it is up to the system engineer to figure out what he wants to happen.

Processor Track Assignments

The processor at present uses part or all of 10 tracks, 00-09. This imposes the following limitations:

Not over 2000 constants in a single program

Not over 1000 operands in all expressions in a single program.

1401 T. P.
3 October 1961
Section 95

Object Program Track Assignments

At present the object program produced by T. P. uses Faces 10-19 for the logic tables leaving the rest of the RAMAC free for data. (Note the processor uses faces 00-09.) Tracks are used consecutively on RAMAC and so short programs will not use the higher numbered faces. One million characters are available for storage of T. P. object program. Just how much is used by each program will depend upon the size of the program and in particular the degree that long constants and expressions are used.

Core Size

The object program will occupy core as follows:

First the system control program, including the standard hardware I/O areas, DO list, etc. Not more than 1700 characters, hopefully about 1200. Note we require exactly 1106 characters at present, but it is not fully checked out, and tape error will require about 35 more. We are allowing 1700 at the moment for convenience.

Next will come the files. These will require the length of the maximum record plus about 35 to 50 characters for each file (except core).

Finally will come logic table area immediately following the data division. As many tracks will be read in as are required to accommodate the table at 880 characters per track. I assume that a good rule of thumb will be that each box filled in will result in using 10 characters, but this is a guess, not even an estimate.

10/3/61
WS:np

Object Program Table Organization

Various phases will have certain functions with regard to object system organization. I will indicate herein the full scheme noting with * deferred features. The deferred features will improve running time by allowing for more than one table to be considered a block and handled at one time and also to operate on a single table which is too large to fit in core at one time. (The former is desirable if there are several small tables which constitute a 'loop'; and the latter is desirable if there are several long expressions and constants in one table.) We will at this time allow space (9 characters) for the information the deferred schemes will require. The space will be filled with ######### to allow for ascending compatability and to permit minimum disturbance to programs when these features are installed.

TABLE HEADER WILL CONTAIN:

col	size	item	filled in by
1-3	3	External table number	Program A
4-6	3	Normal Go To (Track No)	Program A
7-9	3	Error Go To (Track No)	Program A
10-11	2	Number of tracks to hold this block	Macro Generator
12	1	* Does this track contain end of this table?	Macro Generator
13-15	3	* External number of last table in this block	Program A
16-18	3	* Machine address of the header of the next table in this block	Macro Generator
19-20	2	* Number of Rues this track	Macro Generator

Program A

This phase will punch out the following cards where nnn represents the correct 3 digit number:

```
        ORG  TABLES

        DCW  nnn          External Table Number

        DCW  & TABnnn     Normal Go To

        DCW  & TABnnn     Error Go To

      * DCW  nnn          External Table number of last table this block or
                          for the present
```

In addition, this phase will transmit to Macro generator certain information.

*The External table number of the last table in this block will be put in the Communication zone. If this is already filled in, the ORG TABLES will not be punched.

*The external table number of this table will be put in the Communications zone.

*Program A will check to see that no "DONE" table contains a GO TO.

Macro-instruction Generator

This phase will keep track of the number of characters generated and will put the number as part of End of table operations.

```
         ORG TABLE$ & 9
         DCW nn                  number of track this table
         ORG NOTRA$
         DCW nn                  number of tracks this table
TABnnn   EQU mmm                 track this table begins at
         ORG                     PROFT$  2
         DC  mmm                 Track this table begins at
         EX  SYSCR$
         END SYSR$
```

File: Selden mail

1401 T. P.
22 September 1961
Section 99

T. P. System Operation

The operation of the T. P. System will be described with reference to the diagram previously distributed.

STEP 1

Step 1 consists of writing the program, both data description and procedure description, keypunching the cards, listing the cards on the 407, and visually checking for accuracy.

The important items to check for here are correct column alignment, correct spelling, correct usages of zero and "O", correct usage of 1 and I, and other normal keypunching problems. Special points to check for T. P. in the data division: There must be one header card for Table 000 which will come from the first sheet and the there may not be header cards from the second sheet. In the logic division, there must be one header card for each table. There may be, in addition, an "order of rules card" and a comment card. There may be more than one comment card. However, all of these cards must appear before the first line of the procedure division proper. If the procedure division has to be continued to a second page, you may not repeat the header on the second page. All cards, without exception, must be in ascending sequence, but except for continuation, they do not need to be in consecutive.

STEP 2

Step two consists of taking the processor deck furnished and loading the processor on the 1401. Upon completion of the loading, the computer will hang up attempting to read a card. Push START on the card reader to read the last two cards at which time the 1401 will hang up. The system is now loaded.

STEP 3

Place the primer program which is about six or seven cards in the card reader followed by the source language program prepared in Step 1. Press the CARD LOAD button and at the end of the process press the START button to read the last two cards. There is no

mechanism for an END card at the end of the source deck. During running of the processor, an Autocoder language program will be punched and certain material will appear on the printer. This material will form an index of the object program.

The deck of cards produced from T. P. will contain control card and job card. It is a wise precaution to mark the top of the deck or to interpret it so that it does not get out of proper sequence. The deck produced by the processor is ready to go to the Autocoder.

STEP 4

A normal Autocoder assembly will be performed on the cards produced in Step 3. The control card furnished is 661 to produce a system to run on a 16000 character 1401. This machine is available at Endicott; if the machine to run the Autocoder assembly is smaller, you should make a substitution of the control card. Autocoder system operation consists of mounting the system tape on Tape Unit 1, work tapes on units 4, 5, and 6, placing the card deck in the card reader and using the tape load procedure. A listing of the card deck will be produced by the Autocoder. Note should be made of the error messages connected to the Autocoder assembly. In general, the programproduced by the Autocoder will be in the same order as the program written in T. P. source language. An examination of the listing will demonstrate the source of duplicate symbols, undefined symbols, and object program too large to fit in memory. Two exceptions to the general rule are the constants and expressions will appear within each logic table where they are used rather than remaining with the data division where they were written.

STEP 5

Step 5 consists of loading the object program on the RAMAC 1401 combination. The procedure is analogous to Step 2 except that this time the program being loaded consists of the object control system deck furnished by our group followed by the output of the Autocoder assembly in the T. P. program. These decks are put in the card reader, card load is performed and again at the end of the cards the START button is pushed to feed the last two cards.

STEP 6

In step 6, the object program runs on the object machine. The object program primer deck of about 5 or 6 cards is placed in the card reader and card load executed. The object program will begin with the execution of table 001 and will operate in accordance with the instruction written in the source language program. If data cards are called for, they may be placed in the card reader following the primer. Good luck.

WS:np
9/22/61