SYCOR INC

Model 340 Intelligent Communications Terminal

Programmer's Manual

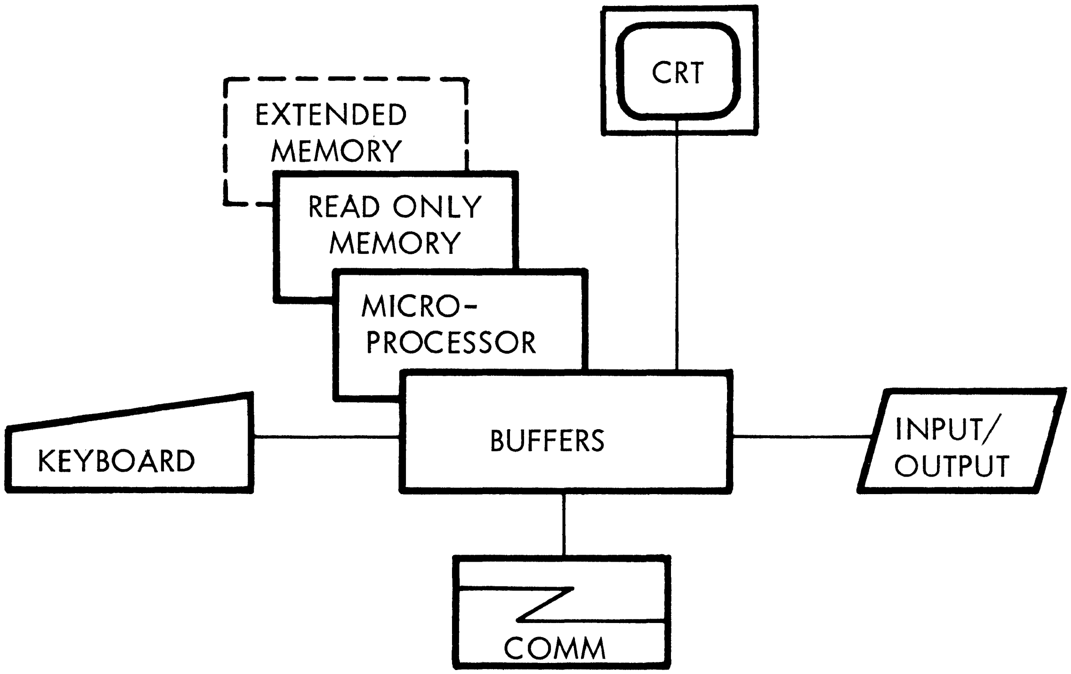
SYCOR, INC.

Model 340
Intelligent
Communications Terminal
PROGRAMMER'S MANUAL

Prepared at Ann Arbor, Michigan
Revised September, 1974

INTRODUCTION

The basic Sycor Model 340 Communications Terminal consists of control logic and peripherals. The control logic incorporates a powerful microprocessor which executes instructions stored in a high-speed read only memory and extended memory. In addition, a high speed random access memory is provided to allow buffering of data transfer operations between peripherals. The basic terminal is shown in block diagram form below. The extended memory option may be thought of as an expansion of read only memory allowing the customer to choose additional features from the Sycor program library.



MODES
OF
OPERATION

FORMATTING

TAL
PROGRAMMING

PERIPHERAL
DRIVERS

PROGRAM
GENERATION

APPENDIX

LIST OF ILLUSTRATIONS

FORMATTING

PROGRAM GENERATION

MODES
OF
OPERATION

INTRODUCTION

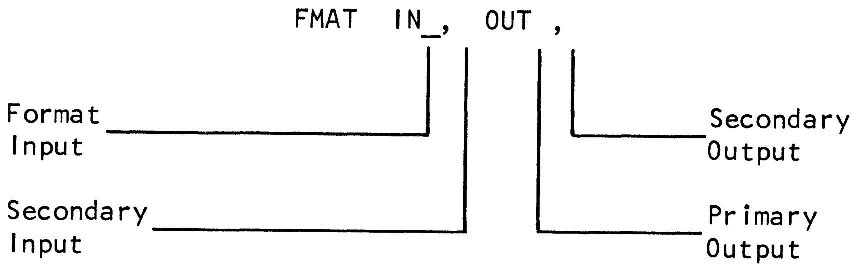
All Model 340 Communications Terminal operations have been divided into four classes or modes: format, search, batch and auto communication. The terminal is placed in these modes by a job selection sequence. The job selection sequence also establishes the normal input and output devices to be used on a particular job. A fifth class, program, has been defined and will be discussed in the extended memory section of the manual.

Job selection sequences are initiated by depressing JOB SELECT followed by an F for format, an S for search, a B for batch, an A for auto, or a P for program. Input/output codes are then entered as follows:

INPUT		OUTPUT	
Cassette one	1	Cassette one	1
Cassette two	2	Cassette two	2
Communications	L	Communications	L
Extended memory input	C	Printer	P
None	Space	Extended memory output	C

FORMAT MODE

Depressing job select followed by an F causes the status line* of the screen to display:



---

*The status line is the top line of the display and contains the job selections, status messages, and accumulators.

with the cursor positioned in the location reserved for the format input. The implied primary input in format mode is always the keyboard or an accumulator. As legitimate I/O device codes are entered, the cursor moves to the next input or output location. When all locations have been filled with codes or spaces, depressing ENTER completes the job selection sequence. The Format Mode sequences described in detail in the Operator's Manual are:

Task	Job Selection Sequence
Enter data	FMAT IN1, OUT2,
Enter data and print	FMAT IN1, OUT2,P
Update	FMAT IN ,1 OUT2,
Update and print	FMAT IN ,1 OUT2,P

If other format operations are used, we suggest writing special operating instructions similar to the standard instructions given in the Operator's Manual.

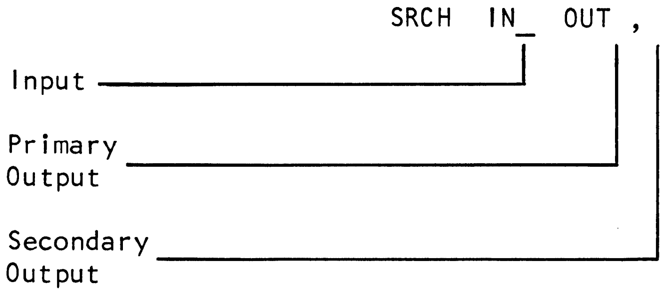
The Format Mode is primarily used to collect source data under format control. For this application the program control (PROG CTL) switch is on and all input and output devices are controlled by the job selection sequence, the format, the ENTER key and the NEXT FORMAT key. The auto operation (AUTO OPRT) switch is also active and determines whether the ENTER key will be required to initiate data output.

Another aspect of Format Mode is format program preparation. When program control is off the I/O controls specified by the job selection are not in effect and the format programmer can create a format page on the CRT display. Each format page is then written on tape (using the tape control keys) and used later, with the PROG CTL switch on, to collect data.

Format Mode with program control off may be thought of as a "free form" mode where all display and tape controls are performed manually.

SEARCH/EDIT MODE

Depressing JOB SELECT followed by an S causes the status line of the screen to display:



with the cursor in the location provided for the input device. As I/O codes are entered, the cursor moves to the next location. Depressing ENTER after all the job selection locations are used completes the job selection sequence and conditions the terminal to accept a search identifier. The search mode sequences described in detail in the Operator's Manual are:

Task	Program Control	Job Selection Sequence
Search 1	off	SRCH 1 OUT ,
Search 2	off	SRCH 2 OUT ,
Search and copy	off	SRCH 1 OUT2,
Search and print	off	SRCH 1 OUTP,
Edit and copy	on	SRCH 1 OUT2,

If other search or edit operations are to be used special operating instructions similar to the standard ones given in the Operator's Manual should be used.

The PROG CTL switch is used in Search Mode to copy either matching records (Search) or non-matching records (Edit). The AUTO OPRT Switch is used to determine if the copy operation will be done automatically or only after ENTER is depressed.

PROG CTL AUTO OPT	OFF	ON
OFF	Skip non-match Stop on match Depress ENTER to output match and continue Stop on file separator	Stop on non-match Depress ENTER to output non-match and continue Skip match Stop on file separator
ON	Skip non-match Output all matches Stop on file separator	Output all non-matches Stop on match Depress ENTER to skip match and continue Stop on file separator

Search Identifiers

A Search Identifier may be up to 256 characters in length and, like all records, must be followed by a record separator. Search identifiers may be continuous or compound. Compound Search Identifiers may be used to find a record when the unique, identifying characters are not next to each other in the data stream. For example, a last name and first name might be separated by an unknown number of spaces, and since spaces are recognized in a search identifier, the record would be difficult to find using a continuous Search Identifier. The compound Search Identifier to find a record on Mary Jones would be:

MARY:JONES■

Compound Search Identifiers must be written in the same order as the characters in the data stream because the 340 terminal finds a match with the first component of the Search Identifier and then searches the remainder of the record for a match with the second component. If the record contained names in reverse order, the search identifier would be written:

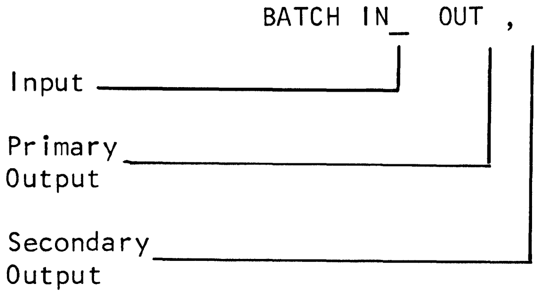
JONES:MARY■

A Search Identifier may be broken into any number of components so long as its total length (including colons) does not exceed 256 characters.

The operator depresses the LOAD ID key to store the Search Identifier and start the search. The Search Identifier is entered right after the job selection sequence. The Search Identifier may be changed after a match is found or after an end of file is encountered.

BATCH MODE

Depressing JOB SELECT followed by a B causes the status line of the screen to display:



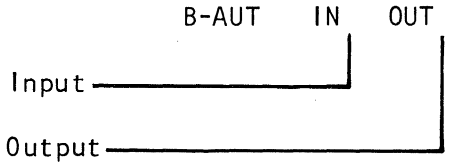
with the cursor in the location reserved for the input device. As I/O codes are entered, the cursor moves to the next location. Depressing enter after all locations are used completes the job selection sequence and starts the batch operation. The Batch Mode sequences described in detail in the Operator's Manual are:

Task	Job Selection Sequence
Batch printing	BATCH IN2 OUTP
Batch copying	BATCH IN1 OUT2
Receiving	BATCH INL OUT1
Transmitting	BATCH IN1 OUTL
Unattended Comm	B-AUT IN OUT

Batch printing is illustrated in the Operator's Manual using cassette recorder two (2) as an input because batch printing often follows formatted data entry and the data tape to be printed is already on cassette recorder two (2). If other batch operations are used, we suggest writing special operating instructions similar to the standard instructions given in the Operator's Manual.

AUTO COMMUNICATION

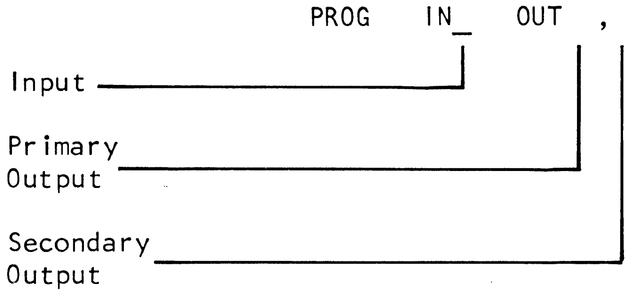
Depressing job select followed by an A causes the status line of the screen to display:



with the cursor not displayed and the keyboard locked. The remote master station has control of further entries in the status line and can select one input and one output for collecting from or transmitting data to the 340. If cassette one is selected as input or output cassette two will be automatically selected when end of tape is sensed on cassette one before sensing the end of file. Both cassette are rewound after data has been collected from an unattended terminal. The cassettes do not rewind, however, after data is received on the unattended terminal.

PROGRAM MODE

Depressing JOB SELECT followed by a P causes the status line of the screen to display:



with the cursor positioned in the input position. Program mode operation is a function of a software loaded in extended memory and will be discussed in that section.

Additional Notes

FORMATTING

INTRODUCTION

This part of the manual provides instructions for planning, designing and preparing formats for the Model 340 Communications Terminal. Formats can utilize 340 features to simplify data entry and detect certain operator errors. Model 340 features contained in the read only memory can control:

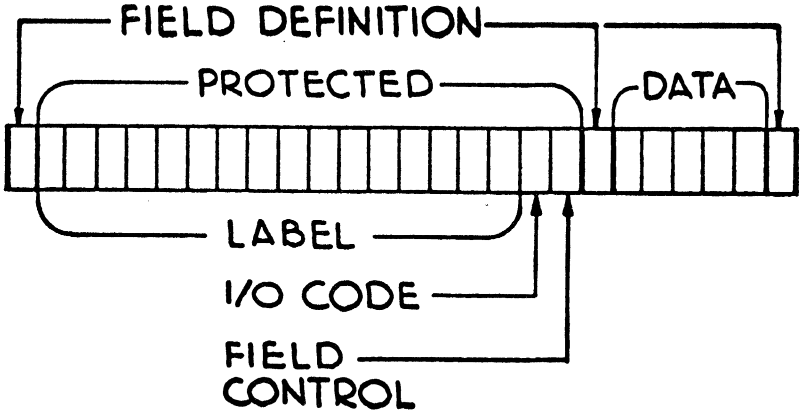
- field and record length
- input/output devices: cassette tapes, communications, printer
- character mode: mixed, alphabetic, numeric
- automatic duplication
- data field compression
- data omission
- data capacity
- right justification (left space fill)
- accumulators (add/subtract)

In addition to the read only memory features, the extended memory option allows expansion of the Model 340 capabilities to include additional input/output devices, format control routines and communication routines. Extended memory features are described in the latter sections of the Programmer's Manual.

FORMAT CHARACTERS

The first step in designing a format is to consider the form of the input data (cards, cassette tape, etc.) and the form the output will take (cassette tape, printed forms, etc.). A job selection sequence establishes the normal inputs and outputs and the format can be used to make exceptions for specific data fields.

A format program divides the 512 characters of the display into protected fields and data fields. The protected fields are separated from the data fields by field definition characters. Protected fields contain labels (or instructions to the operator), I/O controls, and data field control characters. Under program control, the operator fills in the data fields while the terminal automatically controls format and input/output operation. Entry in each data field is controlled by a preceeding protected field.



Typical Format Sequence

As illustrated above, a protected field may contain labels, I/O controls, and data field controls. However, if labels or instructions are not necessary, the protected fields may be reduced to just I/O control and data field control: [4M], for example is a legitimate format control field. If I/O control is to be as specified in the job selection sequence, the format control field may be reduced further to a single protected data field control character such as [M].

All field definition characters, I/O control characters, and data field control characters are defined in the pages that follow.

FIELD DEFINITION CHARACTERS

Brackets [ ]

Brackets are the most common field definition characters. Brackets enclose the protected fields that define data mode and I/O control. Other field definition characters may enclose protected label fields or may stand alone as one character protected fields. A left bracket ends a data field and starts a protected field. A right bracket starts a data field and ends a protected field. Brackets must be used in pairs and must enclose at least one character (one of the field control characters defined later in this section).

Automatic Cursor Advance |

The vertical line symbol is used for automatic cursor advance. The vertical line defines the end of a data field and the start af a protected label field. When the cursor encounters a vertical line it moves at once to the first position in the next data field.

Display Tab Stop \

The display tab stop is translated into a horizontal tab character for the printer and acts as a kind of data field "ditto" in the format. A backslash defines the beginning of a new data field that has the same mode and I/O control as the previous data field. A backslash is written by the tab/skip key when program control is off.

The auto cursor advance and backslash are often used in pairs to partition the display into separate data fields. The sequence which follows, for example, defines four mixed entry data fields separated from one another by horizontal tab characters.

```
[M]       |\        |\        |\        ▮
```

New Line Symbol ¬

The new line symbol is translated into a printer carriage return/line feed code. The hook also behaves like a backslash in defining the data field following it as having the same mode and I/O control as the previous data field. The hook is written by the new line key when program control is off. When a hook appears in a format control program and program control is on, depressing the new line key will move the cursor to the first data field following the hook.

The auto cursor advance and the hook are often used together in much the same way as the auto cursor advance and backslash. In the example below, the data from the four mixed entry fields will be separated by carriage return/line feed codes.

```
     [M]          |⌉        |⌉        |⌉        ■
```

Record Separator ■

A record separator defines the end of a record. It is used to define the end of a search identifier (see Modes of Operation Section) and the end of each format page. The record separator automatically becomes the last character in each data record created under format control.

LABELS

A label is an optional part of a protected field. A label may follow a left bracket or an auto cursor advance. Labels can be used to identify the data field that follows or to give special instructions to the operator. Labels may be any length or may be eliminated completely.

Some labels are illustrated in the four sample formats below:

```
     [NAME            M]      ⌉|
     [ADDRESS         M]      ⌉|
     [CITY, STATE     M]       |
     [ZIP             9 ]   ⌉■

        [NAME M]            |
        ADDRESS⌉            |
     CITY, STATE⌉           |
         [ZIP 9]   ⌉■
```

```
[   NAME          ADDRESS        CITY, STATE      ZIP

M]              ┐              ┐            [9]     ┐■

|   NAME          ADDRESS        CITY, STATE      ZIP

[M]             ┐              ┐            [9]     ┐■
```

All four format samples would create identical data records which if printed would have the form:

Mary Smith
100 Phoenix Drive
Ann Arbor, Michigan 48104

When TAL Programming is used, labels within bracketed protected fields take on additional meaning.

DATA FIELD CONTROL CHARACTERS

The last character before the right bracket "]" in a protected field is the data field control character. Thirty data field control characters are defined for the Model 340 Communications Terminal. Many of these 30 characters are used to implement the special optional functions and may not be present in a particular Model 340. The data field control character controls the mode of the data field that follows it.

Normal Data Entry M,A,N

No special option is required to use these field control characters.

[M] Mixed entry data field. Alphabetic, numeric and special symbols are legal entries.

[A] Alphabetic entry data field. All letters, space, comma, and period are legal entries.

[N] Numeric entry data field. All numbers, minus, comma, period, and space are legal entries.

Tab Compression 1,2,3

The tab compression option provides varible length in mixed, alphabetic, or numeric data fields. When the operator depresses tab/skip to exit a tab compression field, a backslash appears in the data field and a horizontal tab character is inserted in the data in place of the remaining spaces in the data field.

[1] Mixed, like M above but spaces may be replaced by a horizontal tab character.

[2] Alphabetic, like A above but spaces may be replaced by a horizontal tab character.

[3] Numeric, like N above but spaces may be replaced by a horizontal tab character.

Omission Detection 4,5,6

The omission detection option allows you to define data fields that must be used by the operator. At least one legal character must be entered in an omission detection data field or the keyboard will lock and a TAB error message will appear on the top line of the screen.

[4] Mixed, like M above but at least one character must be entered.

[5] Alphabetic, like A above but at least one character must be entered.

[6] Numeric, like N above but at least one character must be entered.

Capacity Control 7,8,9

The capacity control option allows you to define data fields that the operator must fill to capacity or encounter a TAB error.

[7] Mixed, like M above but must be filled completely.

[8] Alphabetic, like A above but must be filled completely.

[9] Numeric, like N above but must be filled completely.

Right Justified Numbers R

Every Model 340 Provides this feature, a [R] specifies a numeric entry field where the numbers are automatically right justified (left space filled) when the operator tab/skips to the next field.

Constant Data C

A [C] data field may be set-up to use the same data over and over. The data for constant fields may be a part of the format tape, or may be entered from the keyboard with program control off. Once program control is turned on, a constant data field is protected, cannot be typed over and becomes a part of each data record created with the format.

Automatic Format Paging *

If [*] appears anywhere in the format page, a new page of format will be automatically displayed on the screen as soon as the current data is on the way to its outputs. This data field control character also does double duty by defining the next field as a constant data field like [C] above.

Accumulator Fields +,T,<,&,S,>,$

A ten-digit plus sign total accumulator and a ten-digit plus sign sub-total accumulator are available as an optional function. The following data field control characters define numeric, right justified (left space filled), accumulator data fields. Each accumulator data field should be long enough to accomodate the number plus a sign (tab/skip for addition, minus for subtraction).

[+] Add or subtract in the total accumulator.

[T] Move the total accumulator value to the data field and clear the total accumulator.

[>] Move the total accumulator value to the data field but do not clear the total accumulator.

[&] Add or substract in the sub-total accumulator.

[S] Move the sub-total accumulator value to the data field and clear the sub-total accumulator.

[<] Move the sub-total accumulator value to the data field but do not clear the sub-total accumulator.

[$] Move the sub-total accumulator value to the data field, add the sub-total value to the total accumulator, and clear the sub-total accumulator.

With the exception of subtracting in [+] or [&] data fields where the minus key is used, all accumulator operations are activated by tab/skip, new line or enter.

Extended Memory I,J,K,L,W,X,Y,Z

The extended memory field control characters are paired to provide mixed entry or numeric entry fields for up to four groups at a time.

The actual functions provided are described in the TAL PROGRAMMING Section of the Programmer's Manual.

INPUT/OUTPUT CONTROL CHARACTERS

The next to last character in a protected field is an I/O control character. If the I/O control character is omitted, input is from the keyboard or the accumulator and output is as defined in the job selection. When an I/O control character is used it selects from the I/O devices defined in the job selection sequence by calling for primary or secondary input or output only. The I/O control character controls the data field that follows it. The format I/O control characters are defined below:

None	Keyboard or accumulator input, outputs specified selection by the job slection sequence.
1	Secondary input, outputs specified by the job selection sequence.
2	Keyboard or accumulator input, primary output.
3	Secondary input, primary output.
4	Keyboard or accumulator input, secondary output.
5	Secondary input, secondary output.

6 Keyboard or accumulator input, no output.

7 Secondary input, no output.

For example [5A] calls for alphabetic data to be brought into the data field from the secondary input device and to be sent to the secondary output device only.

PRINTER CONTROL

In addition to the hook "⌉" and backslash "\", special control characters are provided for designing printer control programs. The additional printer control characters are not protected and must be enclosed in constant data fields. The printer control codes are as follows:

VT	Used to set a vertical tab record.
HT	Used to set a horizontal tab record and to correct data in batch or search modes. To write formats, use the backslash, .
FF	Form feed, the form is advanced to channel A (top of form).
CR	Carriage return, the carriage is returned and the form is advanced one line. Use this character to correct data in batch or search modes. To write formats, use the "hook", .
LF	Line feed, the form is advanced one line.
ESC	Special Model 340 code to identify a printer control sequence.

The printer control codes listed above are non-displayed characters created with the shift key and the numeric keys 4, 2, 5, 3, 6 and 7 respectively. The printer control codes appear to be blanks or spaces on the display except that when the cursor shares the location of a non-displayed character the cursor itself disappears. A line feed may be used only after a carriage return/line feed sequence and may not be used in the middle of a print line.

Horizontal Tab Record

The horizontal tab record is a special record which makes use of the ESC and HT characters to set the horizontal tabs.

For example:

```
          E
           H
        * S 12X45678901X3456X89X █
           T
          C
```

sets horizontal tabs in character positions 3, 12, 17 and 20. The numbers in the example are included only for convenience and may be replaced by spaces or any non-X alphanumeric characters. The backslash character "\" is used in subsequent format pages to tab to the locations set by the horizontal tab record.

Vertical Tab Record

The codes in the Printer Vertical Tab Code Table (Appendix A) are used in conjunction with the ESC and VT characters to set the vertical tab record for the incremental printer. Each vertical tab record consists of an ESC, a VT and seven, two-letter, alphabetic codes. These codes define the number of lines on a form and the line numbers where vertical tabs A through F are set. For example:

```
          E
           V
        * S DBPKAGBBBEBJCD █
           T
          C
```

defines the vertical tab record for a form 66 lines long (DB) with vertical tab channel A at line 11 (PK), channel B at line 23 (AG) and so forth. The alphabetic codes may be calculated if the Printer Vertical Tab Code Table is not available. The scheme is hexadecimal (base 16) with A through P representing the numbers 1 through 15 and zero. Line 58 for example is CJ.

```
      3 ──────────────┐
  16 ⟌58              │
     48             C J
     ──
     10 ──────────────┘
```

The following sequences are used in subsequent format pages to tab to the locations set by the vertical tab record:

E [C]SA C	Used to cause the printer to return the carriage and tab to vertical tab channel A. (Line 11 in the example above).
E [C]SB C	Used to cause the printer to return the carriage and tab to vertical tab channel B. (Line 23 in the example above).
E [C]SC C	Used to cause the printer to return the carriage and tab to vertical tab channel C. (Line 34 in the example above).
E [C]SD C	Used to cause the printer to return the carriage and tab to vertical tab channel D. (Line 37 in the example above).
E [C]SE C	Used to cause the printer to return the carriage and tab to vertical tab channel E. (Line 42 in the example above).
E [C]SF C	Used to cause the printer to return the carriage and tab to vertical tab channel F. (Line 52 in the example above).

Setting Tabs

The horizontal and vertical tab records discussed above are stored in a special buffer and used by the terminal to interpret tab commands (HT, ESCA, etc.) that occur in the print data.

The terminal will store the HT and VT records only if the job selection includes the printer as an output device. In format mode, the VT and HT formats are used to set the tabs:

FMAT IN1 OUT2,P

[*] $\substack{E\\S\\C}\substack{H\\T}$12X4567890345 6X89X■

and

FMAT IN1 OUT2,P

[*] $\substack{E\\S\\C}\substack{V\\T}$DBPKAGBBBEBJCD■

In batch mode, the HT and VT data records are used to set the tabs:

BATCH IN2 OUTP

$\substack{E\\S\\C}\substack{H\\T}$12X4567890345 6X89X■

and

BATCH IN2 OUTP

$\substack{E\\S\\C}\substack{V\\T}$DBPKAGBBBEBJCD■

FORMAT PROGRAMS

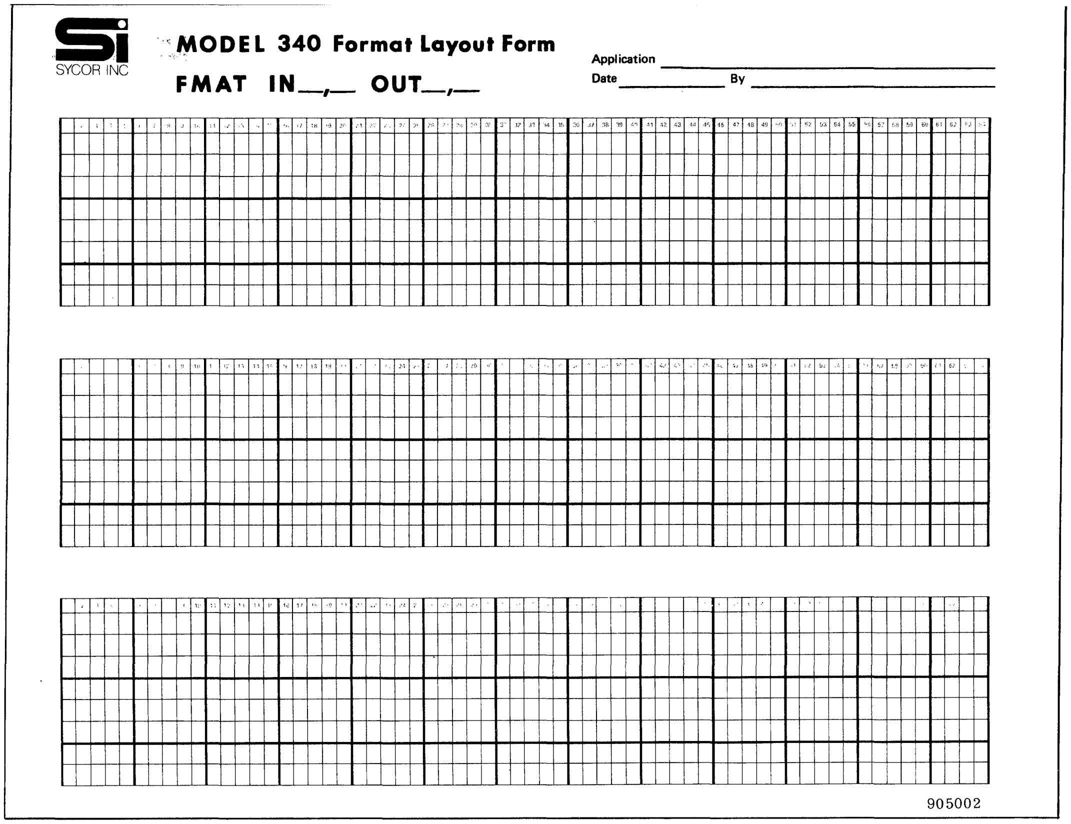
Sycor has designed a special Model 340 Format Layout Form to aid in planning format programs (refer to next page). Each of the three sets of lines on the format sheet represent a format page. Each page consists of eight (8), 64 character lines for a total of 512 possible characters. All Model 340 format program characters are defined on the back of the Format Layout Form for easy reference.

The first character on the first line of a format page must be either a left bracket or a cursor advance character ([or |) and each format page must end with a record separator. The strategy in writing formats is to make the display on the screen resemble the source data while using the I/O controls, constant data fields, and printer controls to structure the data and the printed forms.

The operator should be able to relate what he sees on the screen to the source documents he is using for data entry and should not have to think about the form of the output data.

One Page Format, No Printing

The major consideration in this application is to construct data fields that are compatible with the requirements of the central processor, and to design a format that detects and prevents common operator errors. A payroll data collection format is shown as an illustration. The data collected for computing a payroll must be processed before the checks can be printed. Therefore, the printing is done from the processed data and not from the source data. Alphabetic, numeric, omission detection, capacity control and numeric right justified are used in this format illustration. The format is shown on a format layout sheet in Figure 1. A sample data record is shown below the format as it would appear on the display in search or batch modes. The non-displayed HT and CR (LF) characters are indicated with arrows.



SYCOR INC

MODEL 340 Format Layout Form

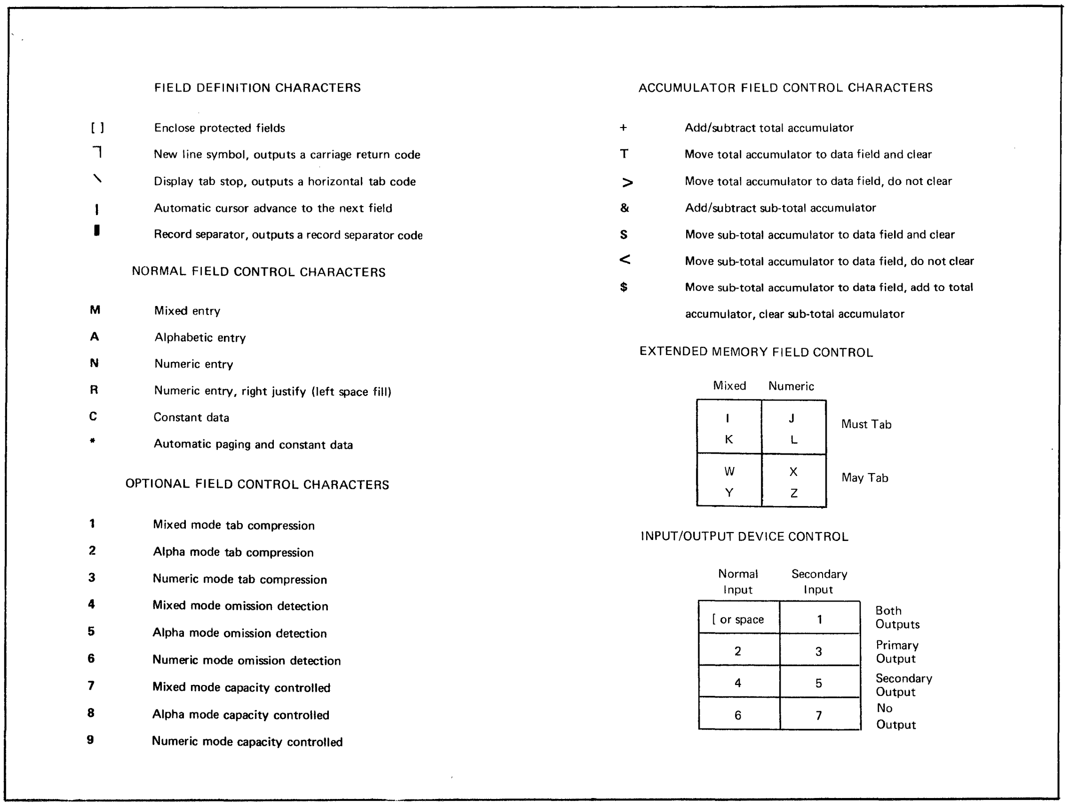
FMAT IN__,__ OUT__,__

Application ____________________

Date __________ By ____________________

905002

Format Layout Form (Front)



FIELD DEFINITION CHARACTERS

[ ]	Enclose protected fields
┐	New line symbol, outputs a carriage return code
\	Display tab stop, outputs a horizontal tab code
|	Automatic cursor advance to the next field
▮	Record separator, outputs a record separator code

NORMAL FIELD CONTROL CHARACTERS

M	Mixed entry
A	Alphabetic entry
N	Numeric entry
R	Numeric entry, right justify (left space fill)
C	Constant data
*	Automatic paging and constant data

OPTIONAL FIELD CONTROL CHARACTERS

1	Mixed mode tab compression
2	Alpha mode tab compression
3	Numeric mode tab compression
4	Mixed mode omission detection
5	Alpha mode omission detection
6	Numeric mode omission detection
7	Mixed mode capacity controlled
8	Alpha mode capacity controlled
9	Numeric mode capacity controlled

ACCUMULATOR FIELD CONTROL CHARACTERS

+	Add/subtract total accumulator
T	Move total accumulator to data field and clear
>	Move total accumulator to data field, do not clear
&	Add/subtract sub-total accumulator
S	Move sub-total accumulator to data field and clear
<	Move sub-total accumulator to data field, do not clear
$	Move sub-total accumulator to data field, add to total accumulator, clear sub-total accumulator

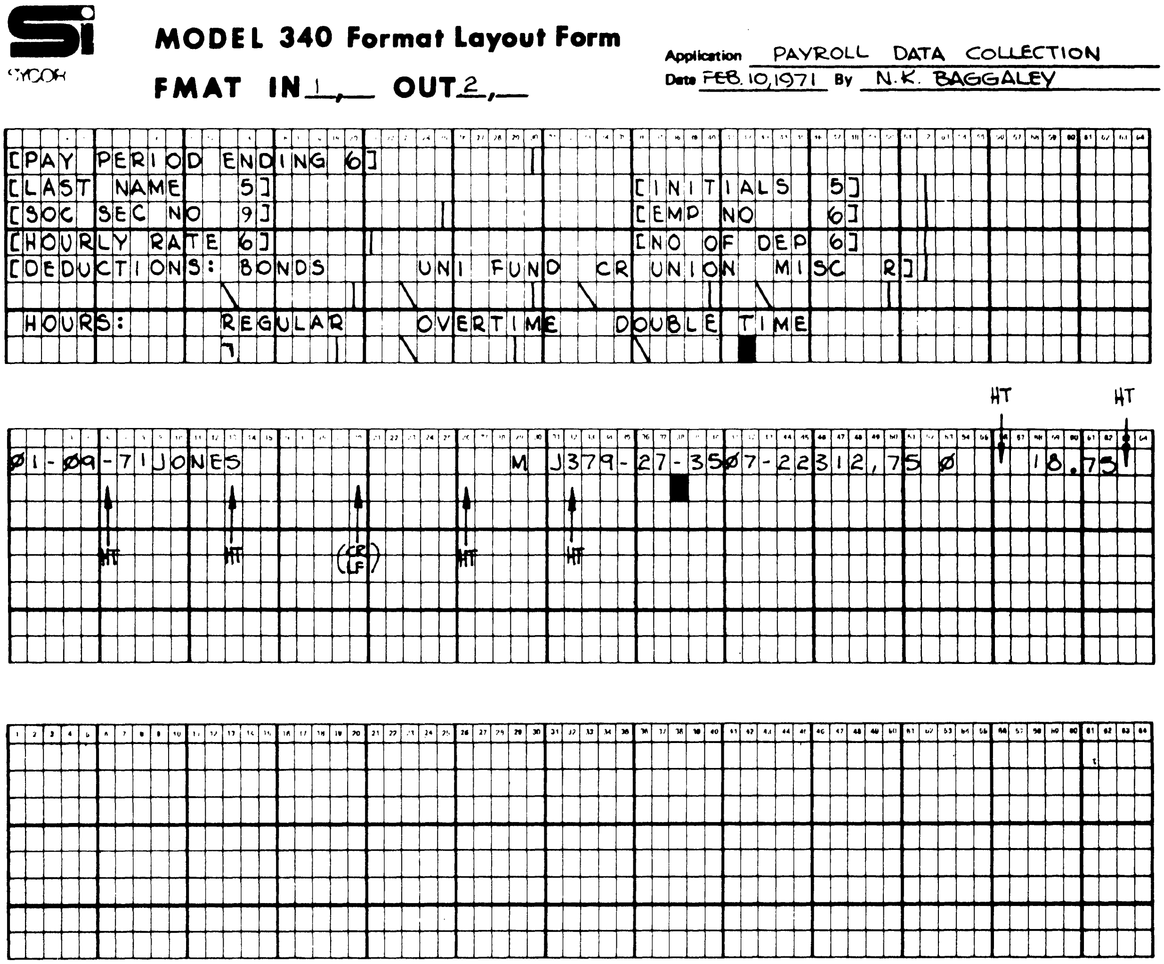
EXTENDED MEMORY FIELD CONTROL

Mixed	Numeric	
I K	J L	Must Tab
W Y	X Z	May Tab

INPUT/OUTPUT DEVICE CONTROL

Normal Input	Secondary Input	
[or space	1	Both Outputs
2	3	Primary Output
4	5	Secondary Output
6	7	No Output

Format Layout Form (Back)



SYCOR

MODEL 340 Format Layout Form

FMAT IN 1, **OUT** 2,

Application PAYROLL DATA COLLECTION

Date FEB. 10, 1971 By N.K. BAGGALEY

```
[PAY PERIOD ENDING 6]
[LAST NAME     5]                [INITIALS    5]
[SOC SEC NO    9]                [EMP NO      6]
[HOURLY RATE 6]                  [NO OF DEP 6]
[DEDUCTIONS: BONDS    UNI FUND   CR UNION   MISC   R]
 HOURS:       REGULAR    OVERTIME    DOUBLE TIME
```

```
01-09-71JONES                    M J379-27-3507-22312,75 0     18.75
```

Figure 1 Payroll Format and Data

Entering Data for Printing and Processing

Many Model 340 applications will involve data that is to be printed directly on a prepared form. An order form and invoice (Figure 2) have been chosen to illustrate this application. The format program, shown in Figure 3, offers a choice of different job selection sequences. The job selection sequence FMAT IN1, OUT2,P calls for the invoices to be printed as the data is being entered. If batch printing is preferred, the data could be entered with the job selection sequence BATCH IN2, OUTP.

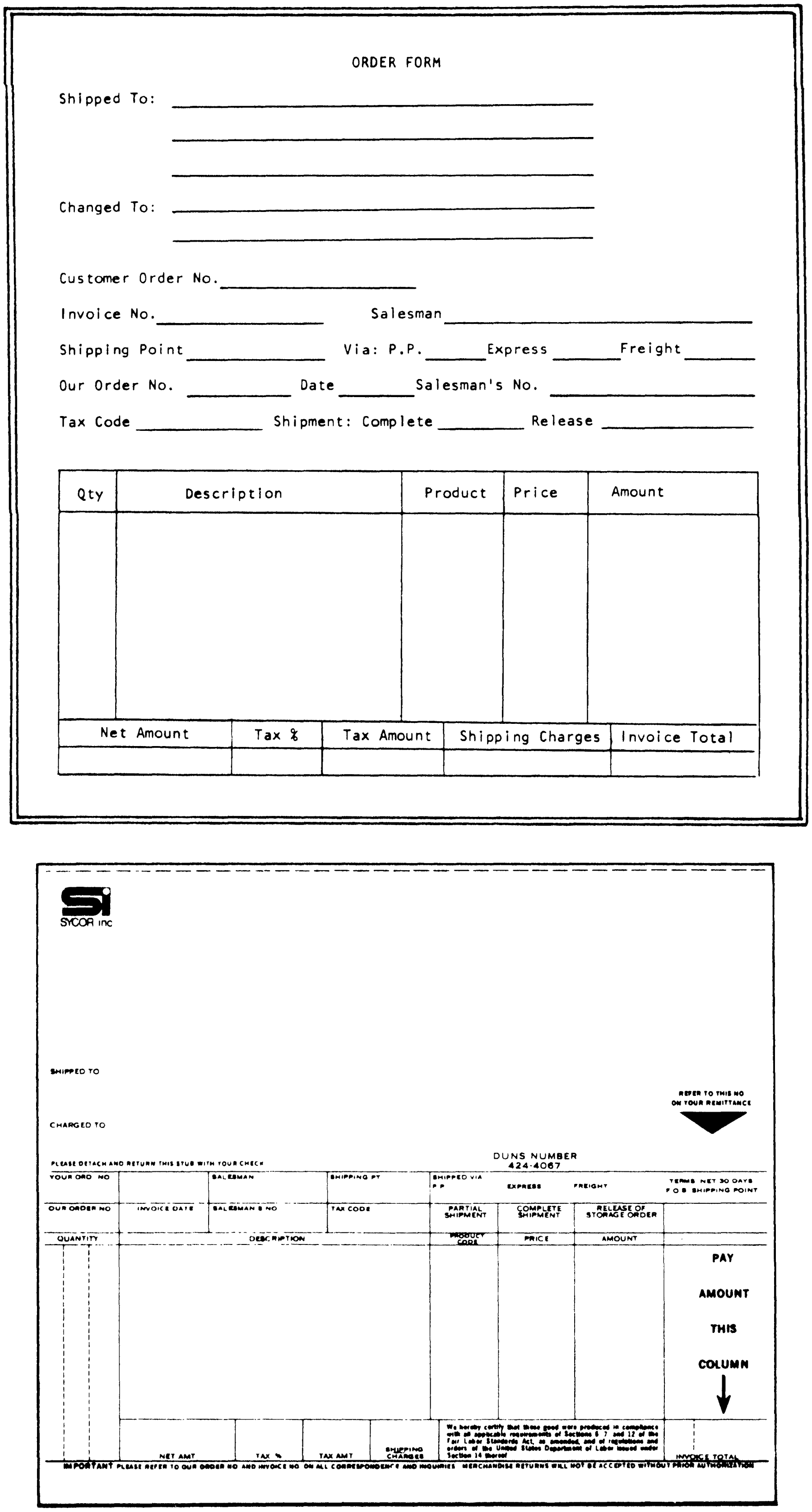
The printer control format program example in Figure consists of six format pages: a horizontal tab sequence, a vertical tab sequence, an address page, a header page, an item page and a trailer page. Notice that the trailer page advances the printer to the first line of the next form:

```
   E
[C]SF¬ █
   C
```

The strategy in writing formats of this type is to make the display on the screen resemble the source data sheet while using printer control codes to make the output data fit the preprinted form.

Preparing Separate Data for Printer and Central Processor

In the previous example the data contained special printer control codes, horizontal tabs, vertical tabs, and carriage return codes. For this reason it was possible to elect to print during formatted data entry or in a batch mode after data entry was complete. However, the central processor would have to have been programmed to ignore the printer codes. An alternate approach to combining printing and processing is to write the original Model 340 format in a form which sends the printer codes to the printer only and not to the cassette tape that will contain the data to be processed (see Figure 4). In this case, since the tape will not contain the printer control codes, the printing must be done during data entry and not as a separate batch process. However, the central processor need not be specially programmed to ignore the printer control codes. The job selection sequence, then, for data entry will be FMAT IN1, OUT2,P. The printer control codes are kept out of the data stream on tape 2 by the use of a constant data field with secondary output only, [4C]. The format illustrated is exactly the same as the one used in the previous example. A comparison of the data tapes from these two examples would show that the only difference is the presence or absence of printer control codes.



ORDER FORM

Shipped To: ______________________

Changed To: ______________________

Customer Order No.__________

Invoice No.__________ Salesman __________

Shipping Point __________ Via: P.P.______ Express ______ Freight ______

Our Order No. __________ Date ______ Salesman's No. __________

Tax Code __________ Shipment: Complete ______ Release __________

Qty	Description	Product	Price	Amount

Net Amount	Tax %	Tax Amount	Shipping Charges	Invoice Total

SYCOR inc

SHIPPED TO

CHARGED TO

REFER TO THIS NO
ON YOUR REMITTANCE

PLEASE DETACH AND RETURN THIS STUB WITH YOUR CHECK

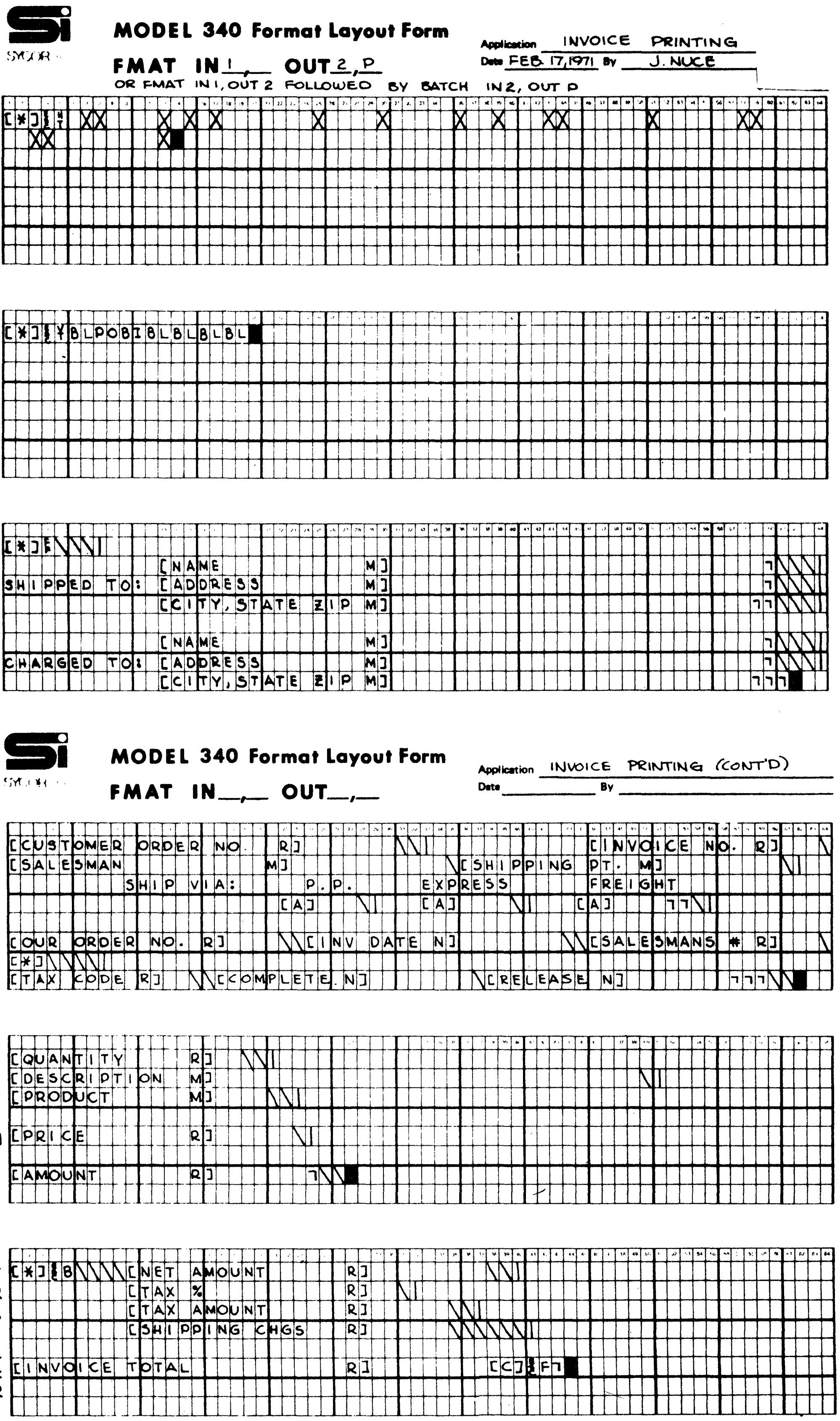
DUNS NUMBER
424-4067

YOUR ORD NO		SALESMAN	SHIPPING PT	SHIPPED VIA P P	EXPRESS	FREIGHT	TERMS NET 30 DAYS F O B SHIPPING POINT
OUR ORDER NO	INVOICE DATE	SALESMAN S NO	TAX CODE	PARTIAL SHIPMENT	COMPLETE SHIPMENT	RELEASE OF STORAGE ORDER	
QUANTITY	DESCRIPTION			PRODUCT CODE	PRICE	AMOUNT	PAY AMOUNT THIS COLUMN

NET AMT	TAX %	TAX AMT	SHIPPING CHARGES	We hereby certify that these good were produced in compliance with all applicable requirements of Sections 6 7 and 12 of the Fair Labor Standards Act, as amended, and of regulations and orders of the United States Department of Labor issued under Section 14 thereof	INVOICE TOTAL

IMPORTANT PLEASE REFER TO OUR ORDER NO AND INVOICE NO ON ALL CORRESPONDENCE AND INQUIRIES MERCHANDISE RETURNS WILL NOT BE ACCEPTED WITHOUT PRIOR AUTHORIZATION

Figure 2 Order Form and Invoice



MODEL 340 Format Layout Form

FMAT IN 1, **OUT** 2, P

OR FMAT IN 1, OUT 2 FOLLOWED BY BATCH IN 2, OUT P

Application INVOICE PRINTING

Date FEB. 17, 1971 By J. NUCE

HORIZ TAB

```
[*]}HT  XX      X X X      X     X    X  X   XX      X        XX
   XX           X█
```

VERT TAB

```
[*]}¥BLPOBIBLBLBLBL█
```

ADDRESS

```
[*]}\\\|
              [NAME                M]                                 ┐\\\|
SHIPPED TO:   [ADDRESS             M]                                 ┐\\\|
              [CITY, STATE ZIP     M]                                ┐┐\\\|

              [NAME                M]                                 ┐\\\|
CHARGED TO:   [ADDRESS             M]                                 ┐\\\|
              [CITY, STATE ZIP     M]                               ┐┐┐█
```

MODEL 340 Format Layout Form

FMAT IN__, __ **OUT**__, __

Application INVOICE PRINTING (CONT'D)

Date ________ By ________

HEADER

```
[CUSTOMER ORDER NO.     R]          \\|             [INVOICE NO.  R]  \
[SALESMAN               M]             \[SHIPPING   PT.  M]          \|
            SHIP VIA:      P.P.          EXPRESS        FREIGHT
                        [A]     \|        [A]     \|    [A]    ┐┐\|

[OUR ORDER NO.  R]      \\[INV DATE N]          \\[SALESMANS #  R]    \
[*]\\\\|
[TAX CODE R]  \\[COMPLETE.N]               \[RELEASE N]        ┐┐┐\\█
```

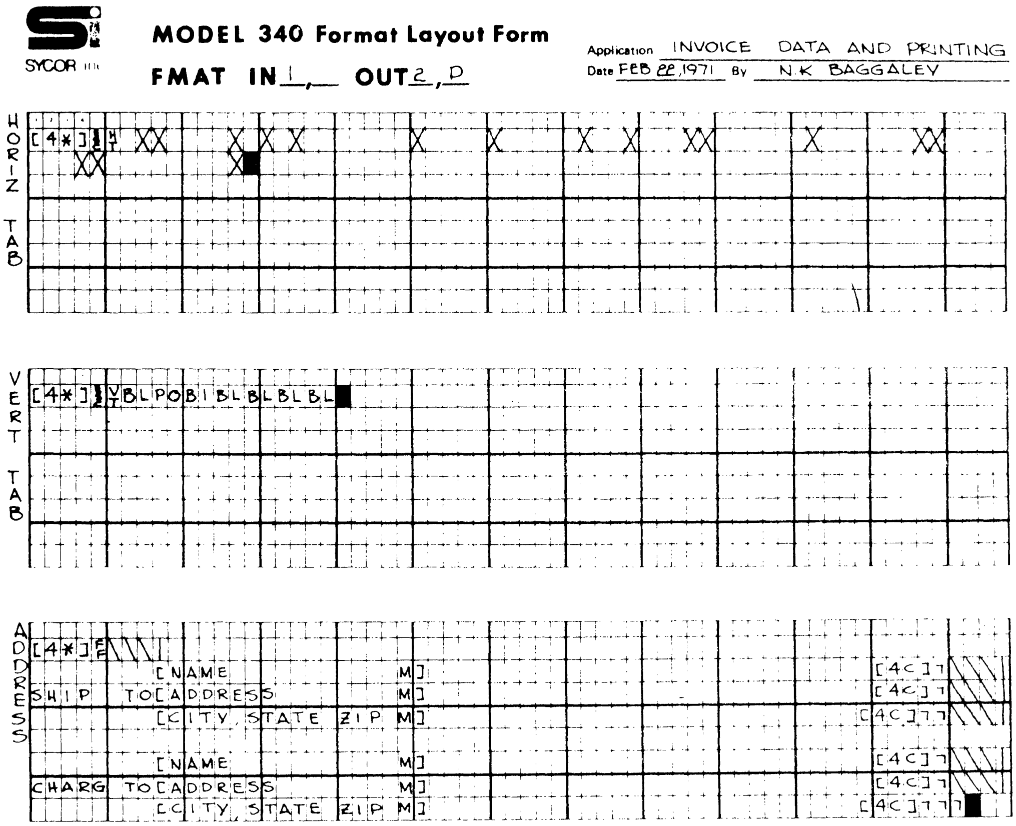
ITEM

```
[QUANTITY          R]    \\|
[DESCRIPTION       M]                                   \|
[PRODUCT           M]      \\|

[PRICE             R]        \|

[AMOUNT            R]          ┐\\█
```

TRAILER

```
[*]}B\\\\[NET AMOUNT          R]          \\|
         [TAX %               R]    \|
         [TAX AMOUNT          R]       \\|
         [SHIPPING CHGS       R]       \\\\\\|

[INVOICE TOTAL                R]          [C]}F┐█
```

Figure 3 Invoice Printing Formats



MODEL 340 Format Layout Form

FMAT IN 1 , **OUT** 2 , D

Application INVOICE DATA AND PRINTING
Date FEB 22,1971 By N.K BAGGALEY

HORIZ TAB

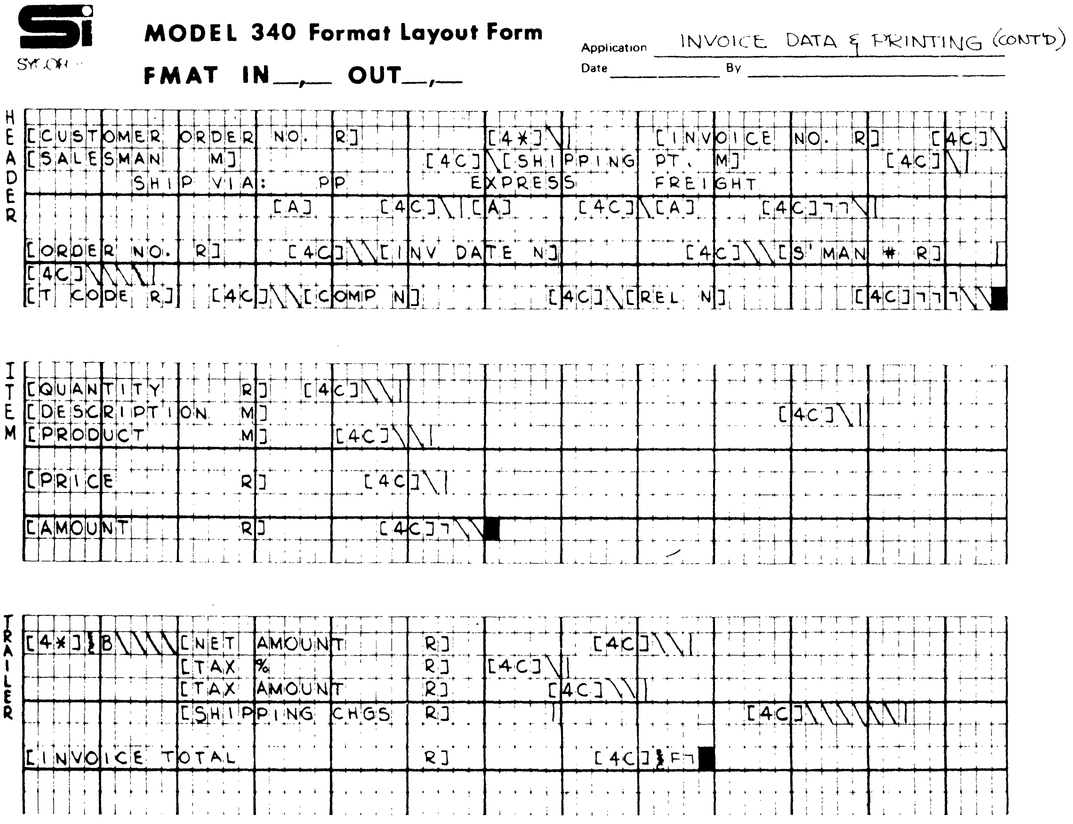
```
[4*]}HT XX      X X X      X      X      X X   XX      X      XX
      XX            X■
```

VERT TAB

```
[4*]}VT BL PO BI BL BL BL BL■
```

ADDRESS

```
[4*]FF\\\|
            [NAME                    M]                              [4C]⌐\\\|
SHIP   TO[ADDRESS                    M]                              [4C]⌐\\\|
            [CITY, STATE  ZIP        M]                             [4C]⌐⌐\\\|

            [NAME                    M]                              [4C]⌐\\\|
CHARG  TO[ADDRESS                    M]                              [4C]⌐\\|
            [CITY, STATE  ZIP        M]                             [4C]⌐⌐⌐■
```



MODEL 340 Format Layout Form

FMAT IN __,__ **OUT** __,__

Application INVOICE DATA & PRINTING (CONT'D)
Date ______ By ______

HEADER

```
[CUSTOMER ORDER NO.  R]          [4*]\|      [INVOICE NO.  R]      [4C]\
[SALESMAN      M]           [4C]\[SHIPPING PT.  M]            [4C]\|
        SHIP VIA:      PP          EXPRESS       FREIGHT
                   [A]       [4C]\|[A]      [4C]\[A]       [4C]⌐⌐\|

[ORDER NO.  R]       [4C]\\[INV DATE N]             [4C]\\[S'MAN # R]     |
[4C]\\\|
[T CODE  R]   [4C]\\[COMP N]             [4C]\[REL N]           [4C]⌐⌐⌐\\■
```

ITEM

```
[QUANTITY         R]   [4C]\\|
[DESCRIPTION      M]                                        [4C]\|
[PRODUCT          M]    [4C]\\|

[PRICE            R]        [4C]\|

[AMOUNT           R]         [4C]⌐\\■
```

TRAILER

```
[4*]}B\\\\[NET AMOUNT        R]           [4C]\\|
          [TAX %             R]   [4C]\|
          [TAX AMOUNT        R]       [4C]\\|
          [SHIPPING CHGS     R]       |              [4C]\\\\\\|

[INVOICE TOTAL               R]           [4C]}F⌐■
```

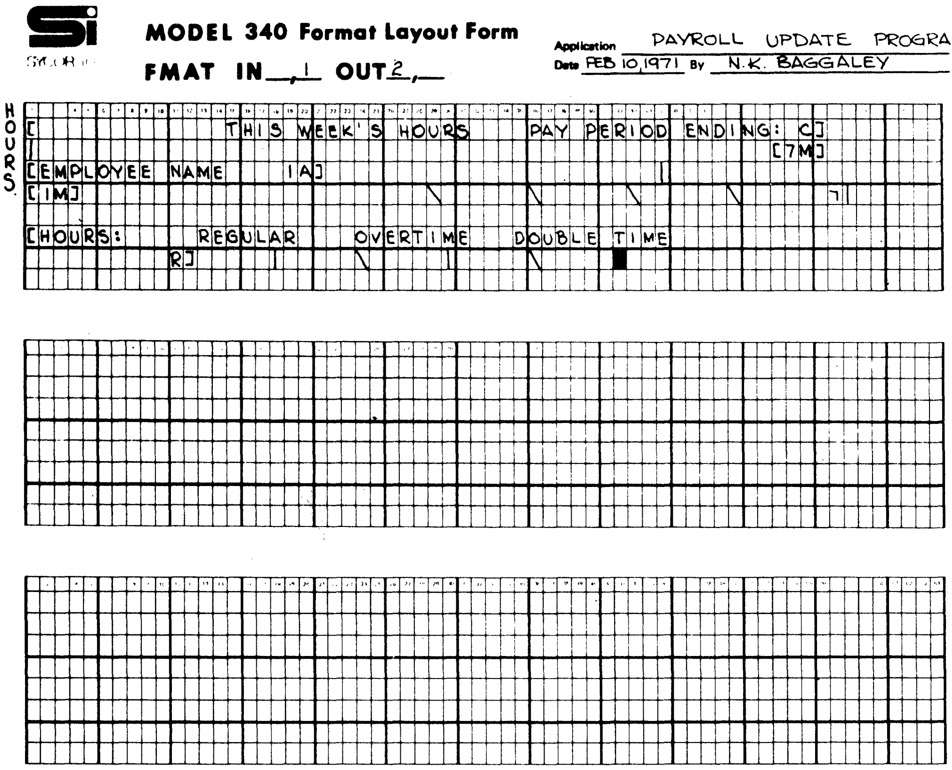
Figure 4 Invoice Data and Printing Formats

Updating Existing Data

An update program can be written for any existing data set. The ability to call data in from a secondary input (present in all Model 340 terminals) and a second cassette recorder make updating possible. An update format distinguishes between data that will change and data that will remain the same. Data that does not change is called in from the secondary input and used as is in the output data stream, [1M]. Data that will be replaced is called in from the secondary but with "no output" specified, [7M]. At this point in the update, new data is keyed in from the keyboard and entered in the output data stream. These operations are alternated where necessary throughout the update format.

Figure 5 is an update format program to update the payroll data collected with the format shown in Figure 1 . This update could be used regularily for every employee every week to enter the number of hours worked.

The update operation is very useful when all records in a data set are being changed. If you want to revise the data in a single record out of a group, you should use a combination of search and insert operations. Search for the data record, type over the data that needs to be changed, depress tape 1 or tape 2, and depress insert.



MODEL 340 Format Layout Form

FMAT IN ,1 OUT2,

Application PAYROLL UPDATE PROGRAM

Date FEB 10,1971 By N.K. BAGGALEY

HOURS

```
[            THIS WEEK'S HOURS     PAY PERIOD ENDING: C]
|                                                 [7M]
[EMPLOYEE NAME    |A]                      |
[IM]                     \      \      \      \      ¬|

[HOURS:     REGULAR    OVERTIME    DOUBLE TIME
           R]     |    \      |    \   ■
```

Figure 5 Payroll Update Format

Updating and Printing

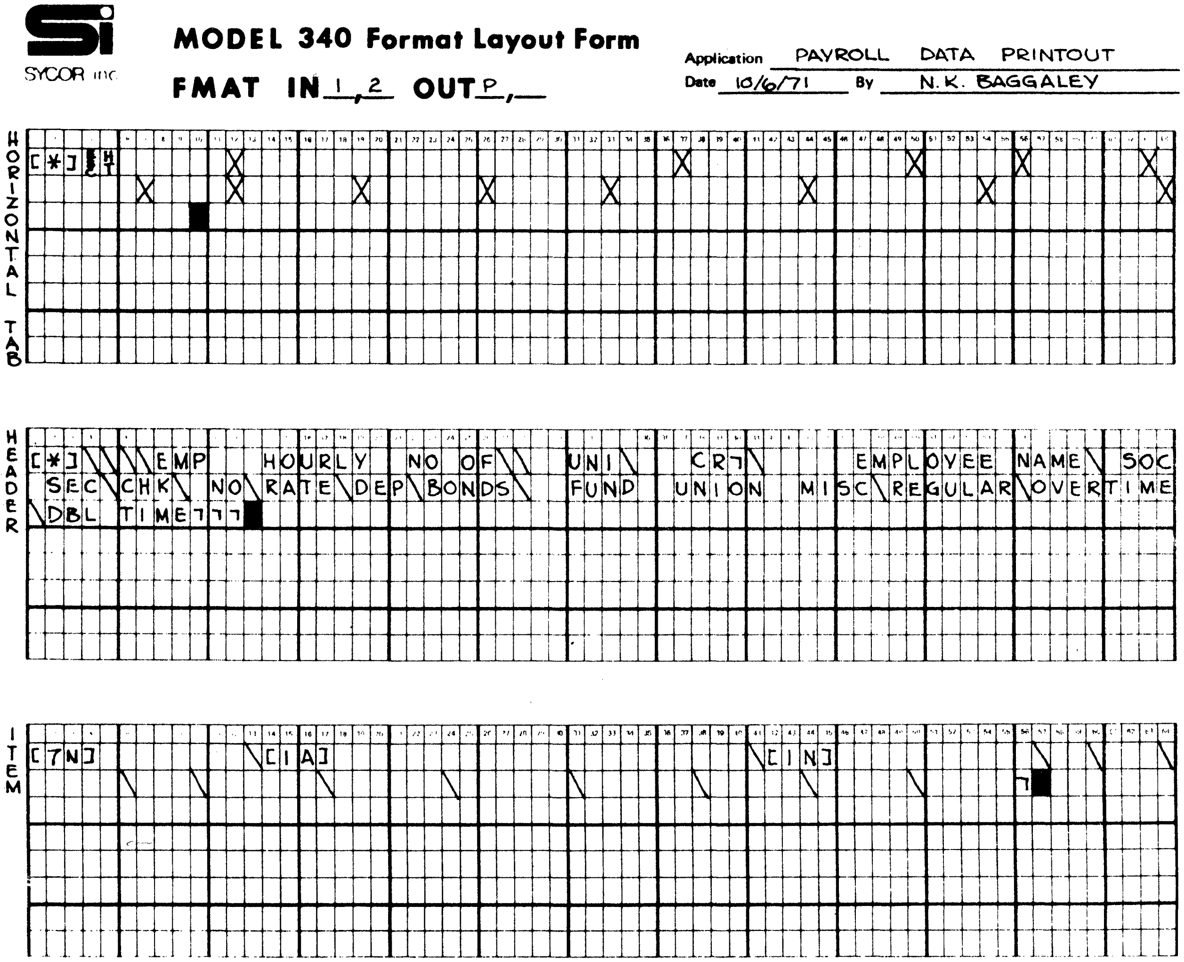
A feature of the Model 340 updating procedure is automatic removal of all printer control characters (ESC, HT, VT, LF, CR, FF) from the secondary data before bringing it into the data fields on the display. Update formats, therefore, generally contain the same printer control characters, hooks and backslashes as were used in the original data collection format (Figures 1 and 5 for example). However, this feature may be used to change the print format by using an update format program that contains different printer control characters. The original printer control characters will be removed and the output data will contain only the printer control characters created by the update format. This feature is extremely useful in situations where data files exist and a new or different print requirement arises.

If escape sequences have been used in the original data, (ESC, A through F) only the escape characters will be automatically removed when the data comes in as secondary input. To restore the original printer controls, only the escape character need be included as constant data in the update format. To eliminate the original printer controls, the A, B, C, D, E or F will have to be eliminated in a one character 7M data field.

Examples:

	ORIGINAL FORMAT	UPDATE FORMAT
Preserve Original Printer Control	E [C]SA | C	E [C]S[1M]_ | C
Eliminate Original Printer Control	E [C]SA | C	[7M]_ |
Change Original Printer Control	E [C]SA | C	E [C]SB[7M]_ | C

Below is a format that might be used to print the payroll data collected and updated in Figures 1 and 5.



SYCOR inc

MODEL 340 Format Layout Form

FMAT IN 1, 2 **OUT** P,

Application PAYROLL DATA PRINTOUT

Date 10/6/71 By N.K. BAGGALEY

HORIZONTAL TAB

[*] HT

HEADER

```
[*]\\\\EMP   HOURLY  NO OF\\  UNI\   CR¬\    EMPLOYEE NAME\ SOC
 SEC\CHK\  NO\RATE\DEP\BONDS\ FUND  UNION  MISC\REGULAR\OVERTIME
\DBL TIME¬¬¬■
```

ITEM

```
[7N]          \[IA]                        \[IN]              \    \    \
     \   \       \      \      \      \       \      \     ¬■
```

PREPARING A FORMAT TAPE

Format tape structure depends on the length of the format (number of pages) and on whether or not printing is being done. A single cassette tape may be used to store many one page formats but, because of the format tape rewind feature, a separate cassette tape must be used for each multiple page format.

Fill out a Sycor Format Layout Form with the format you have designed. Select a suitable format mode job selection and work with program control off. Key in the formats on the terminal and write them on tape manually (tape 1 or tape 2 followed by write). Turn program control on and test each format as soon as it is written on tape. When format errors occur, turn program control off, correct the format on the display and correct the format tape by doing an insert (tape 1 or tape 2 followed by insert).

One Page Formats

This is the simplest case. The format tape consists of a single record: the format, exactly as written on the format layout form. However, if several one page formats are to be stored on the same tape each format should be preceeded by a separate format title record. The format title record is used to locate a format in search mode without having the tape head positioned after the actual format record. A convenient form for a title record is:

```
[TITLE      6M]_▮
```

Using a format structure for the title record makes it possible to advance through the format tape using the next format key without turning program control off. This method may be used for locating formats near the beginning of the tape and saves the operator the trouble of changing to a search mode job selection to locate the correct format.

An index of titles may be recorded at the beginning of the tape to remind the operator of the exact search identifiers and of the order of the formats on tape.

```
[    TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE
     TITLE          TITLE       6M]_▮
```

When the data generated by a one page format is to be printed, a horizontal tab format and a vertical tab format may preceed the format on the format tape. Format tape structure for a tape containing a mixture of one page formats with and without printing is illustrated in Figure 6. Automatic paging, [*], is used only for the horizontal and vertical tab formats.

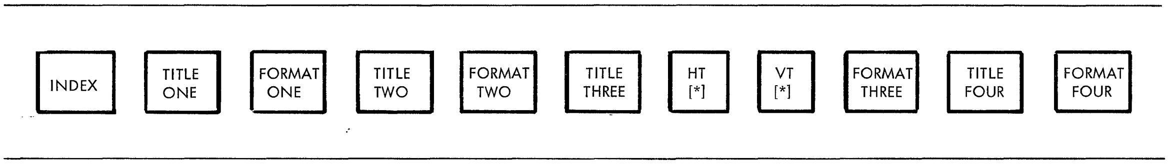


Figure 6 Format Tape Structure, One Page Formats

Multiple Page Formats

Multiple page formats generally consist of a combination of single use pages and item pages. The pages that are used once will contain a protected asterisk, [*], and will be cleared when data entry is complete. Item pages do not contain a protected asterisk and are used a number of times for each source document. The operator depresses next format to advance to the next format page when an item page is no longer needed.

When the Model 340 encounters a file separator on the format input tape, the tape automatically rewinds and the first format page is displayed again. To minimize the time spent waiting for the format tape to rewind, we recommend repeating the formats on the input tape so that a number of records may be entered before the tape rewinds. A typical format tape would contain format pages 1,2,3,4 . . . . N; 1,2,3,4 . . . N; 1,2,3,4, . . . N; . . . followed by a file separator. The format pages should be repeated enough times to allow an operator to complete a two hour work period without having the tape rewind.

When printing is combined with multiple page formats, repeated recordings of the format pages becomes even more necessary. A typical format tape for printing from a multiple page format would contain a horizontal tab record, a vertical tab record, format pages 1,2,3, . . . N; 1,2,3, . . . N; 1,2,3, . . . N; . . ., and a file separator. In the printing situation, the printer tabs are reset by the horizontal and vertical tab formats each time and format tape is rewound. Therefore, in the interest of the throughput, it is important to record many sets of the format pages on the format tape.

Figure 7 Format Tape Structure, Multiple Page Formats

An alternate method of arranging printer format tapes requires more operator set-up time but provides assurance that the HT and VT records will be used only once. If the alternate tape organization is used (Figure 8) the Model 340 operation instructions will have to be changed so that the operator positions the tape after the first file separator before turning program control on.

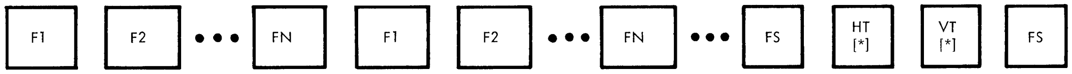
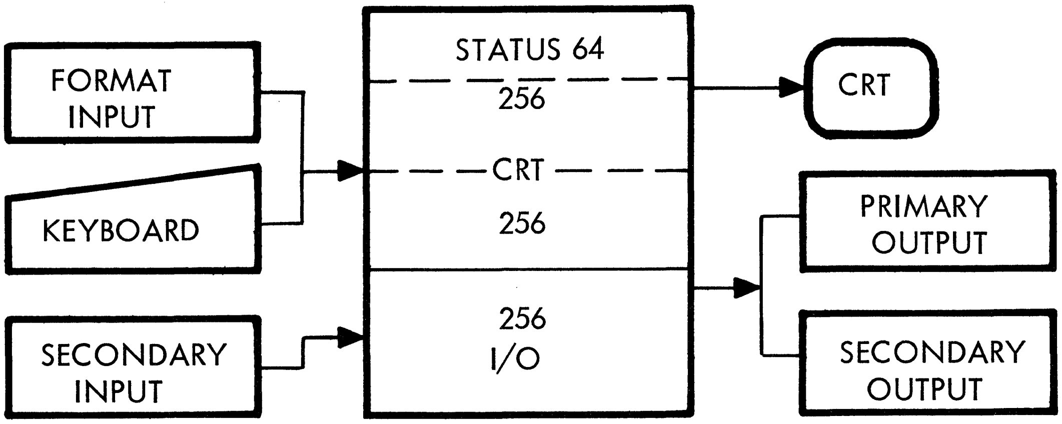


Figure 8 Alternate Format Tape Structure

A convenient way of advancing past a file of formats is to use a batch job selection with no output specified and with auto operation off. When EOF is reached the operator can turn auto operation and program control on and select a format mode job selection. The HT and VT records will be executed, the tape will rewind, and the first format will be displayed on the screen. Multiple copies of the format sequence are recommended to maximize tape life.

DATA FLOW IN FORMAT MODE

Model 340 format mode data flow is illustrated below. The CRT buffer contains both the format and data characters exactly as they appear on the CRT. When program control is on and the operator releases the data to the outputs, the format program is executed by the microprocessor in the following sequence.



1. Scan each character on the CRT and collect all primary output characters.
2. Store the primary output data characters in the I/O buffer as they are collected.
3. If a primary output is specified in the job selection sequence, transfer the contents of the I/O buffer to the primary output.
4. If the primary output data set is greater than 256 characters, repeat steps two and three for the additional characters.
5. Scan each character on the CRT again and collect all secondary output data characters (only if the secondary output set is not identical to the primary output set).

6. Store the secondary output data characters in the I/O buffer as they are collected.

7. If the secondary output data set is greater than 256 characters, repeat steps 5 and 6 for the remaining characters.

8. Check for auto paging etc., and prepare for the next data entry operation.

9. If a secondary output is specified in the job selection sequence, transfer the contents of the I/O buffer to the secondary output.

   NOTE: Step 8 and data entry may overlap step 9.

10. When the operator releases the next data page, the sequence is repeated beginning with step 1.

The I/O buffer is also used when the job selection sequence and the format call for secondary input. For example:

```
FMAT   IN1,C OUT2,P

[M]1234[1M]ABCDEFGHI[M]567 |

[1M]JKLMN[2M]8901[4M]234 |

[5M]OPQRSTUVWXYZ █
```

The operator enters data in the first field, [M]1234. When the cursor enters the next data field, [1M], the microprocessor executes the secondary input command, reads a record from the secondary input, stores the record in the I/O buffer and brings nine characters from the buffer into the data field on the CRT. The I/O buffer pointer is then left in position 10. The operator then completes the next data field, [M]567, and the next secondary input field is filled from positions 10 through 14 in the I/O buffer. This process can be repeated until the record in the I/O buffer is exhausted, a data page is released, a new format page is displayed or the tape control keys are used. After one of these four events, a call for additional secondary input data will cause a new record to be read into the I/O buffer.

The following events occur when the data in Figure   is released by the operator.

1. Primary data is collected and stored in the I/O buffer:

   1234ABCDEFGHI567JKLMN8901 ▮

2. Primary data is written on the cassette tape.

   1234ABCDEFGHI567JKLMN8901 ▮

3. Secondary data is collected and stored in the I/O buffer.

   1234ABCDEFGHI567JKLMN2340PQRSTUVWXYZ ▮

4. The data fields are cleared to prepare for additional data entry.

   NOTE: Step 4 and data entry may overlap step 5.

5. The secondary data is printed.

   1234ABCDEFGHI567JKLMN2340PQRSTUVWXYZ ▮

Since the I/O buffer is never cleared, it may also be used for auto duplication. In this case the job selection sequence should not specify a secondary input or a secondary output. For example, in the following format:

```
FMAT   IN1,  OUT2,

 [2N]  [1M]        [*] ▮
```

since no secondary input device is specified, the secondary input fields will be filled from the data left in the I/O buffer by the previous operation. This fact can be used to recall data from the I/O buffer and re-use it in the next format page.

For this example assume that the I/O buffer contained the letters A through H and a record separator before the preceeding sample format page was used.

FMAT IN1, OUT2,

[2N]12345[1M]ABCDEFGH[*] ▮

The operator enters data in the primary input field [2N] and the [1M] field is filled from the I/O buffer.

When the operator releases the data the following steps occur.

1. Primary data is collected and stored in the I/O buffer.

   12345ABCDEFGH ▮

2. Primary data is written on the cassette tape.

   12345ABCDEFGH ▮

3. Secondary data is collected and stored in the I/O buffer.

   ABCDEFGH ▮

4. A new format page is advanced onto the screen.

5. No secondary output is specified in the job selection so no output is executed but notice that the I/O buffer again contains the letters A through H in the first eight positions.

Other format pages in the sequence may not require any of these characters but they must be saved to be re-used when the first format page comes up again. The "saving" is accomplished by bringing in the eight characters in a [5M] field (secondary input, secondary output) at the beginning of each format page. For example, the format: [5M]ABCDEFGH[N]37915▮ would result in having 37915 ▮ written on cassette tape and ABCDEFGH37915 ▮ left in the I/O buffer after the format was executed.

By careful use of the format I/O control characters and the job selection sequence, almost any auto duplication task amy be accomplished.

Additional Notes

TAL
PROGRAMMING

INTRODUCTION

The Extended Memory Feature expands the capability of the Model 340 terminal. Customized data editing programs can be easily written by the user in Terminal Application Language (TAL). Additional input/output peripherals drivers and communications systems are preprogrammed.

A Program Generator is provided by Sycor to translate TAL programs into machine language instructions and can be used on any 340 Terminal with the Extended Memory Feature.

This same generator is used to select the peripheral drivers and communication systems from the Software Library and "bring it all together".

The end product of generation is a User Program which can be transmitted in any Model 340 communications mode. This object program is loaded into the terminal by a "loader". The loader is one record in length and is the only portion of the Software Library that consists of characters outside the graphic character set. This means that the loader must be distributed by copying cassettes and mailing them.

The communication systems, once loaded, run under their own control; while the data editing programs are loaded and run under format control. Therefore, the first task is to design a format and indicate which fields will require the more extensive TAL Programming; and which will require ROM features.

LOGICAL AND PHYSICAL RECORDS

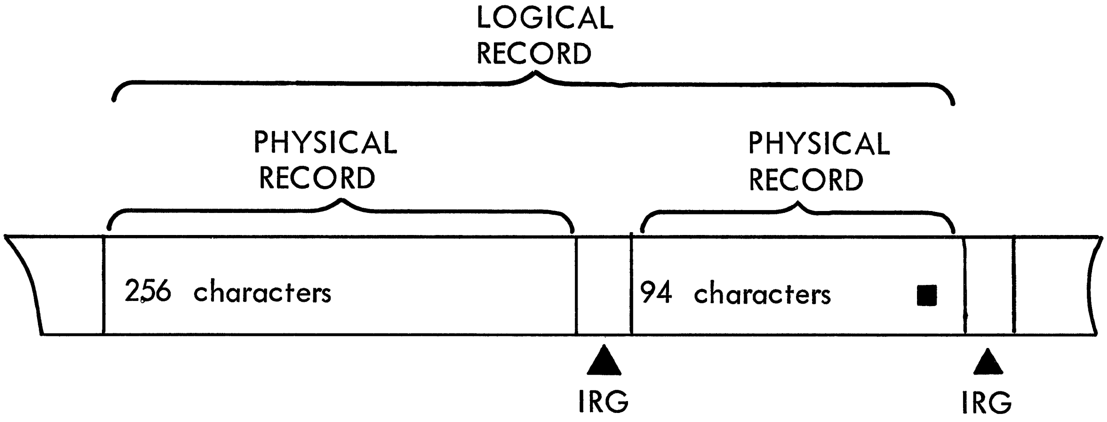
Direct control of the input/output devices through the use of certain TAL instructions is possible. Therefore, an understanding of the differences between a Physical Record and a Logical Record is necessary.

A Logical Record is a group of characters terminated by a RS (record separator). The maximum length of a logical record (including the RS) is 512 characters. A record that long is rare, however, since some of the 512 character-positions on the CRT are used to display the format.

A Physical record has a maximum length of 256 characters.

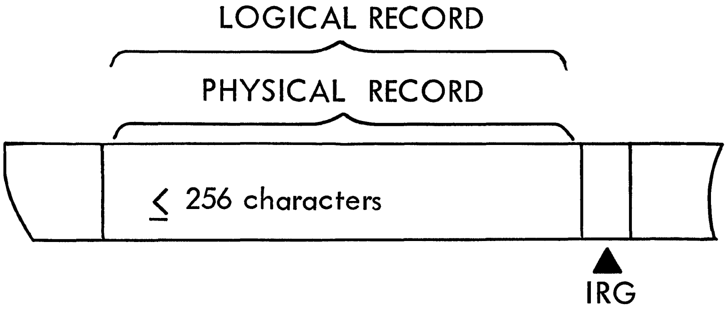
The I/O Buffer in the terminal is 256 characters long, meaning that a maximum of 256 characters can be input or output at one time. If the logical record is longer than 256 characters, it is input or output in two cycles or two physical records.

A logical record of 350 characters would be output through the I/O Buffer as two physical records, which the following diagram shows.

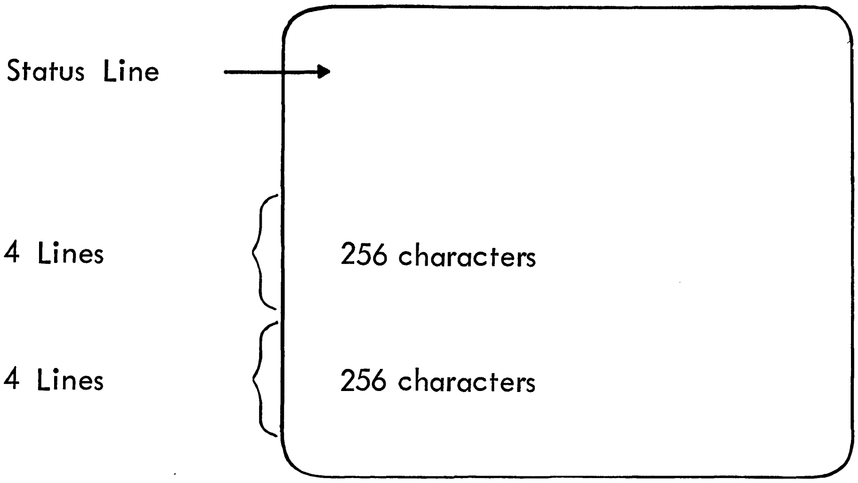


The "IRG" is an Inter Record Gap, a physical space on the tape between the two physical records. The IRG is required to allow for starting and stopping the tape over the read/write head of the recorder.

If the logical record is 256 characters or less, the length of the physical record is the same as that of the logical record. The following diagram illustrates this.



The CRT is divided into eight lines of 64 characters each. The first four lines consist of 256 characters; one physical record. If the format on the screen occupies any portion of the bottom half of the screen, it will require two physical records to output the format to the format tape.



Most formats will occupy at least part of the bottom half of the screen and will therefore require two write cycles (two physical records) to record the format on the format tape.

While the format usally requires two physical records, the data collected with the format may require only one physical record. This is because the protected fields are not output during data collection; only the contents of the data fields are output and this can be less than the 256 character limit of a physical record.

FIELD PROGRAMS

A field program must be written for each format field which has an extended memory field control character in the field control position of the protected field. A field program can consist of up to 63 instructions. Each instruction consists of an OP CODE and a TABLE NUMBER.

EXTENDED MEMORY FIELD CONTROL CHARACTERS

The main function of an Extended Memory field control character is that it informs the format processor that a field program has been written for the field in which the character resides. It has the effect of linking the format with extended memory. There are eight extended memory field control characters which can be used with a programmed field. The chart below shows these characters with their specific functions.

EXTENDED MEMORY FIELD CONTROL

Mixed	Numeric	
I K	J L	Must Tab
W Y	X Z	May Tab

Must Tab Field-Mixed (I, K)

A must tab field requires that the operator depress the TAB/SKIP key in order to execute the field program,whether the field is partially or completely filled by data.

May Tab Field-Mixed (W, Y)

A may tab field requires that the operator depress the TAB/SKIP key to execute the field program only if a field is partially filled with data. If the defined field is completely filled with data by keyed input or from other input devices, program execution is automatic.

Must Tab Field-Numeric (J, L)

Same as mixed with the exception that the rules of a numeric field applies.

May Tab Field-Numeric (X, Z)

Same as mixed with the exception that the rules of a numeric field apply.

TABLES

The operands for each TAL operation is written in a table format. Each table can consist of from one to n elements, limited only by the capacity of the CRT (512 characters). Some operations require fixed length tables whereas others allow a variable number of elements within a table. Elements of a table are separated by a backslash character " \ ". A record separator, immediately following the last element, defines the end of table.

Sharing Elements of a Table

An operation that requires one element in a table can utilize a table which has been defined with more than one element, assuming the first element of the multi-element table satisfies the requirements of that operation. An example of element sharing is shown below with the ADD and JUMP operation. The ADD operation requires a three element table which in this case specifies that Element 1, a constant of 3, is added to Element 2, register 0. and is stored in Element 3, also Register 0 the JMP operation requires a one element table which specifies the number of program steps to be skipped relative to the JMP instruction.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	1	2	3
	RNG	100	100		F	\	3	\	1	\	5	0	■
	ADD	200	200		3	\	R	0	\	R	0	■	
	JMP	200											
	ADD	300	300		1	\	S	A	\	S	A	■	
	SSW	400	400		8	\	1	■					
	GTO	300											

Since the table was defined with the ADD operation the table need only be referenced by its number, for the JMP operation. The execution of JMP instruction will result in skipping 3 program steps relative to that instruction. The JMP operation will ignore the last two elements of the table. The GTO operation uses table 300 which utilizes only the first element of the table.

Sharing Complete Table

All of the elements or part of the elements of a table can be shared by an unlimited number of operations.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2
			A	D	D	2	0	0	2	0	0		F	\	R	1	\	R	9	■
				~																
			S	U	B	2	0	0												
			G	T	O	1	5	0	1	5	0		1	■						

The ADD instruction results in the contents of the field being added to Register 1 and the result stored in Register 9. The SUB instruction utilizing the same table, results in the contents of Register 1 being subtracted from the field with the result stored in Register 9.

Table 200 was defined once; and satisified the requirements of both the ADD and SUB operations.

TAL PROGRAM FORM

The Model 340 TAL Program Form is provided for the writing of field programs and for the definition of tables associated with the operations in the field programs. A copy of the program form is on the next page.

Program Name

Three positions are allocated for the specifying of a Program Name. This name will be used to call the program from the program library.

Switches, Registers and Accumulator Indicators

Three boxes are located at the top of the TAL Programming Form which allow the programmer to indicate the specific Switches, Registers or Accumulators used in the application program. This allows the programmer to account for those used.

68% p. 81-A

SYCOR INC

MODEL 340 TAL PROGRAM FORM

PROGRAM NAME ☐☐☐

APPLICATION ______ SHEET ______ OF ______

DATE ______ BY ______

SWITCHES*	8	7	6	5	4	3	2	1
X IF USED								

REGISTERS*	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
X IF USED										

ACCUMULATORS	SA	TA
X IF USED		

IJ KL WX YZ	FIELD PROGRAMS			TABLES
	FIELD LABEL	OP CODE	TABLE NO.	TABLE NO. 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4

OP CODES Circle those used	ACP ADD ANS BEL BSP CKA CK1 CK2 CK3 CK4 CK5 CK6 CK7 CK8 CK9 CK0 CMP CRC CVB CVD DIV DUP EBC EDT ERR ETC EXF GET GTI GTO INS IXR JMP LMA LOD MAP MOP MPY MVE MVM NFT NXF NXT OVL PTI PUT RDX REG* REW RNG RTJ SCN SHF SIR SMA SRH SSW STR SUB TCP TLU TSC TSW VFY WRT #AP #AR #CP #CR #CT #CO #MA #MG #MH #MN #MS #MT #MX #M7 #M9 #PT #ST

*** the REG op code must be specified if the Registers or Switches are used**

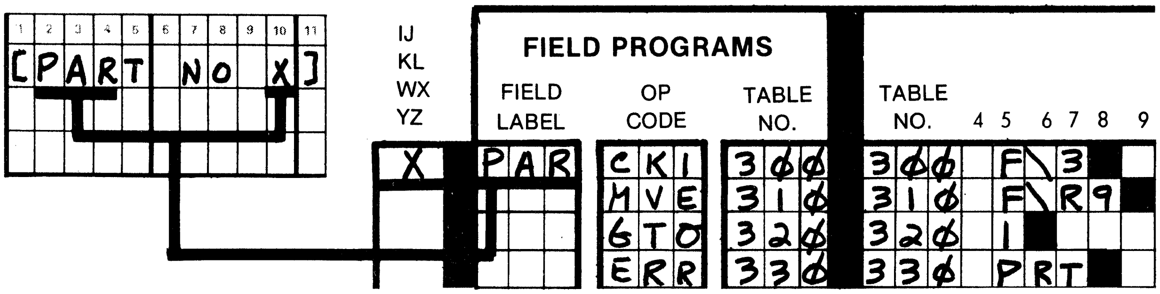
Extended Memory Field Control Character

The first column on the left is labelled IJ,KL,WX,YZ. This is the Field Control Character column. Each field program is associated with an Extended Memory Field Control character and should be entered in the column next to the field program. During program generation these characters specify the sequence in which the field programs will be presented to the generator. The field for which the field program is written will also have this character associated with it in the format.

Field Label

Each field program must be identified by a Field Label which is the first three characters of the label identifying that field in the format.

The following diagram shows the linkage between the format and the field program. The field program is linked with the format by a combination of the extended memory field control character and the field label.



During the program execution the specific field program will be located by the first three characters of the field label and the associated field control character.

OP Code

For each TAL instruction an OP Code must be entered from the list of available OP Codes at the bottom of the TAL form.

Table Number

For each TAL instruction a three character Table Number must be entered which references a table defined on the right side of the TAL form. The first character must be a numeric, followed by any two characters.

Table Definition

Tables are to be defined on the right hand side of the TAL form under the section marked Tables. Tables do not have to be defined in the same line as the OP code which references it; however, for clarity and convenience it is recommended that it be done that way. Tables are presented to the TAL assembler separately from the field programs. Software routines such as #PT do not require a table.

A table consists of a three character number(columns 1-3) and one or more elements(starting in column 5). Elements in a table are separated by a " " character and a RS must end the table. The operation determines the number of elements and their content.

OP Codes Used

A list of available OP codes in the TAL library is printed at the bottom of the TAL program form. It is the responsibility of the programmer to circle each OP code used in the program. Multiple use of the same OP code requires that it only be circled once. This information is used as input to the TAL generator.

TAL ABREVIATIONS

Abreviations are used in the coding of tables. What follows is a general description of the abreviations. Many instructions impose their own limitations on the elements of a table and will be explained in the section pertaining tp that instruction.

Registers (R0-R9)

There are ten Registers available in TAL. Each Register consists of 11 positions, ten for data and one for sign in arithmetic operations. When using the Registers for unsigned data all positions can be used for storage. The Registers are allocated to 110 contiguous positions in the 340's memory(10 Registers, 11 positions each). To set aside this space in memory the REG OP code must be specified during generation. Registers 0-9 are referenced in TAL as R0-R9.

Index Register (IR)

This is a one byte register, made up of eight bits, which is not to be confused with registers 0-9. It is normally used to store a byte address for the I/O buffer. The Index Register is referenced as IR.

Accumulators (SA,TA)

The Subtotal and Total Accumulators which are accessible through the formatting feature of ROM can also be referenced by TAL. The physical makeup of the Accumulators are identical to the registers with the exception that they are displayable on the status line of the CRT. The Accumulators are referenced as follows: SA for the Subtotal Accumulator and TA for the Total Accumulator.

Switches (SW or 1-8)

There are eight programmable switches in TAL. The Switches can be set to ON (1) or OFF (0). The REG OP code must be specified during generation in order to allocate a position in memory for the switches. Switches 1-8 are referenced by a 1-8. All eight switches can be referenced at once by specifying SW, which refers to the byte where the Switches are located.

Fields (F)

Fields are defined in the program format but can be referenced through TAL. Field size is limited only by the capacity of the data entry portion of the CRT. By specifying F, a field can be referenced in a program. F always refers to the current field, i.e. where the cursor is located at the time of execution.

TAL CONVENTIONS

In addition to the abreviations covered in the previous paragraphs the TAL instructions use the following conventions.

Location (LOC)

Whenever LOC is specified as a table element the following abreviations may be used: F,SA,TA or R0-R9.

Constant (CON)

Whenever CON is specified as a table element the element may contain numeric, alphabetic or mixed information. Refer to the individual instructions for any restrictions.

Relationship (REL)

REL is used in branching instructions to denote Relationships such as =,≠,<,>,≤ and ≥ .

ADDITION

ADD

ADD LOC,CON\ LOC,CON\ LOC ■

The contents of Element 1 are added to the contents of Element 2 and the result is stored in Element 3.

NOTES:

- Commas and decimal points are ignored.
- Leading and trailing blanks are ignored.
- Embedded non-numerics cause truncation.
- The result is left-justified in Element 3, with insignificant zeros eliminated.
- Overflow occurs if the result exceeds the capacity of Element 3, or if Element 1, 2, or 3 exceeds 10 digits.
- If Element 1 or 2 is null, a value of zero is assumed.

EXAMPLES:

Addition of a field and a register, storing the result in another register:

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO. 4 5 6 7 8 9 1 0 1 2
	ADD	140	140 F\R1\R2 ■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	1	8	4	5		1

R1	2	0	1	6						
R2	A	7	5							

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	1	8	4	5		1

R1	2	0	1	6						
R2	3	8	6	1						

Addition of a constant to a register:

FIELD PROGRAMS												
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2
	ADD	165	165		1	\	R	9	\	R	9	

Before

R9 145

After

R9 146

Addition of a field to a register with embedded non-numerics:

FIELD PROGRAMS												
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2
	ADD	1AC	1AC		F	\	R	6	\	R	6	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	1	5	4	5		1

R6 12A742

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	1	5	4	5		1

R6 2287

SUBTRACTION

SUB LOC,CON\ LOC,CON\ LOC■

The contents of Element 2 are subtracted from the contents of Element 1, and the result is stored in Element 3.

NOTES:

- Commas and decimal points are ignored.
- Leading and trailing blanks are ignored.
- Embedded non-numerics cause truncation.
- The result in Element 3 is left justified with insignificant zeros eliminated.
- Overflow occurs if the result exceeds the capacity of Element 3, or if Element 1, 2, or 3 is more than 10 digits.
- If Element 1 or Element 2 is null, a value of zero is assumed.

EXAMPLES:

Subtraction of a register from a field and storing the result in another register.

FIELD PROGRAMS				
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 1 0 1 2
	SUB	140	140	F\R1\R2

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	F	I	E	L	D		X	]	2	1	8	7		1		

R1	2	0	1	6								
R2	3	7	5									

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	1	8	7		1

R1	2	0	1	6							
R2	1	7	1								

Subtraction of a register from a constant:

FIELD PROGRAMS

										1		
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2
	SUB	165	165		R	9	\	1	\	R	9	

Before

R9 144

After

R9 145

Subtraction of a field from a register with an embedded non-numeric:

FIELD PROGRAMS

										1		
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2
	SUB	1AC	1AC		R	6	\	F	\	R	6	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	F	I	E	L	D		X	]	1	2	A	7	4	2	1

R6 1545

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	F	I	E	L	D		X	]	1	2	A	7	4	2	1

R6 803

MULTIPICATION

MPY LOC,CON \LOC,CON \LOC \CON ■

The contents of Element 1 are multiplied by the contents of Element 2, and the result is stored in Element 3. Element 4, which is optional, specifies the number of digits to be rounded off in the product.

NOTES:

- Commas and decimal points are ignored.
- Embedded non-numerics cause truncation.
- The result is left-justified in Element 3, with insignificant zeros eliminated.
- Element 1 can not exceed 10 digits.
- If the third Element is a field it can be defined up to twenty-one positions, including the sign.
- If Element 1 or 2 is blank, a value of zero is assumed.
- Overflow occurs if the result exceeds the capacity of Element 3.
- Rounding does not take place, when Element 4 is not specified.

EXAMPLES:

Multiplication of a field and a register, storing the result in another register, while rounding 2 places.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.							1				
				4	5	6	7	8	9	0	1	2	3	4
	MPY	010	010		F	\	R	1	\	R	2	\	2	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	1	3	3	3		1

Before		After	
R1	3	R1	3
R2	0	R2	40

Multiplication of a register by a constant and storing the product in the total accumulator.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.							1		
				4	5	6	7	8	9	0	1	2
	MPY	020	020		R	3	\	6	\	T	A	

Before		After	
R3	12	R3	12
TA	0	TA	72

Multiplication of a field by a register with an embedded non-numeric:

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	1	2	3	4
	MPY	030	030		F	\	R	7	\	R	7	\	1	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	4	2	3		1

R7

2	A	3								

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	4	2	3		1

R7

7	2	7								

Additional Notes

DIVISION

DIV LOC,CON\ LOC,CON\ LOC\ LOC,CON ■

The contents of Element 1 are divided by the contents of Element 2, and the result is placed in the location specified by Element 3.

If Element 4, which is optional, is a constant it states the number of additional significant digits to carry out the calculation of Element 3 and rounding takes place on the last significant digit. If Element 4 is a location, the remainder is placed in that location.

NOTES:

- Commas and decimal points are ignored.
- Embedded non-numerics cause truncation.
- The result is left-justified in Element 3, with insignificant zeros eliminated.
- If Element 1 or 2 is blank, a value of zero is assumed.
- If Element 3 is a field, it can be defined up to twenty-one positions, including the sign.
- If Element 2 equals zero, overflow occurs.
- Overflow will occur if the calculation or remainder exceeds the capacity of Elements 3 or 4 respectively.

EXAMPLES:

Division of a field by a register and storing the result in another register while carrying out the calculation two extra places:

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	11	12	13	14
	DIV	010	010		F	\	R	1	\	R	2	\	2	■

Before		After	

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	0	0	0		1

Before		After	
R1	3	R1	3
R2	0	R2	66667

Division of a register by a constant and storing the remainder in the total accumulator.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	11	12	13	14	15
	DIV	020	020		R	3	\	3	\	R	3	\	T	A	■

Before		After	
R3	25	R3	8
TA	0	TA	1

Division of a field by a register with an embedded non-numeric:

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	11	12	13	14	15
	DIV	030	030		F	\	R	7	\	R	7	\	S	A	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	4	2	3		1

R7 2A3

SA 50

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	F	I	E	L	D		X	]	2	4	2	3		1

R7 847

SA 2

Additional Notes

ASCII COMPARE

A C P

ACP LOC,CON\ REL\ LOC,CON\ CON ■

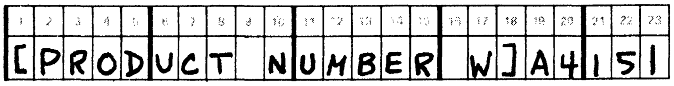
Element 1 is compared to Element 3, by the relationship in Element 2. If the comparison fails program control jumps the number of steps specified in Element 4.

NOTES:

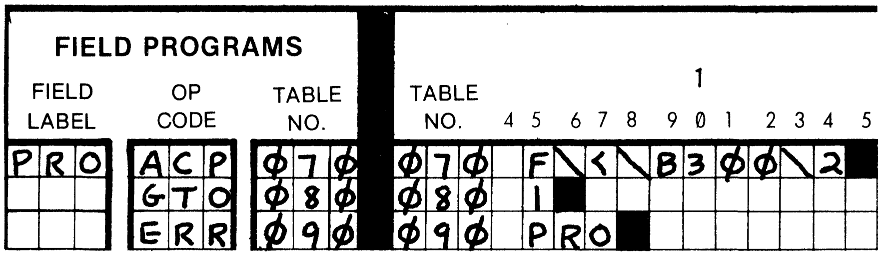
- Valid relationships for Element 2 are =, /=,>,<,> = and <=.
- "/=" in Element 2 represents an unequal relationship.
- Any ASCII character is valid; therefore all spaces are significant.

EXAMPLES:

Test the product number entered to make sure it is less than B300.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
[	P	R	O	D	U	C	T		N	U	M	B	E	R		W	]	A	4	1	5	1



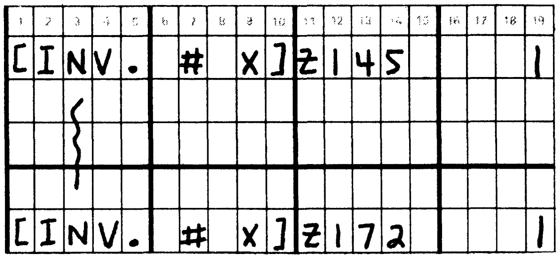
FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.							1					
				4	5	6	7	8	9	0	1	2	3	4	5
PRO	ACP	070	070		F	\	<	\	B	3	0	0	\	2	■
	GTO	080	080		1	■									
	ERR	090	090		P	R	O	■							

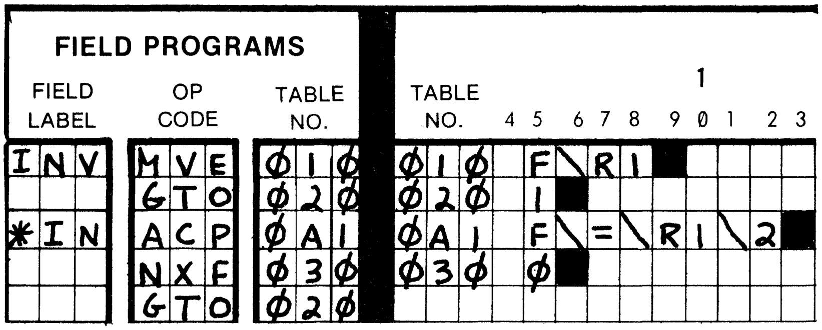
NOTE:

- Refer to the ASCII chart in Appendix.

If the invoice is repeated in two distinct places, call in the next format.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	I	N	V	.		#		X	]	Z	1	4	5					1
[	I	N	V	.		#		X	]	Z	1	7	2					1



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.		4	5	6	7	8	1 9	0	1	2	3
INV	MVE	010	010		F	\	R	1	■					
	GTO	020	020		1	■								
*IN	ACP	0A1	0A1		F	\	=	\	R	1	\	2	■	
	NXF	030	030		0	■								
	GTO	020												

NOTE:

- Since the Z145 entered in the INV. # field is unequal to the Z172 entered in the *INV. # field, program control jumps to the GTO 020 instruction.

CHECK DIGITS

CK_ LOC \ CON ■

Check digit verification is performed on data located in Element 1. If the check digit is not verified, program control jumps according to the value in Element 2.

NOTES:

- The check digit instruction is capable of performing different verifications, depending upon the last character of the instruction. These check digit OP CODES, and their algorithms, are listed below.

OP CODES	MOD	MULTIPLIERS right justified	Sum of	
			Digits	Products
CK0	10	7, 6, 5, 4, 3, 2		X
CK1	10	2, 1, 2, 1, 2, 1, 2	X	
CK2*	11	2, 7, 6, 5, 4, 3, 2		X
CK3	10	1, 3, 7, 1, 3, 7		X
CK4*	11	2, 8, 7, 6, 5, 4, 3, 2		X
CK5*	11	2, 7, 6, 5, 4, 3, 2		X
CK6**	7	NONE		N/A
CK7*	11	2, 9, 8, 7, 6, 5, 4, 3, 2		X
CK8*	11	2, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2		X
CK9**	NONE	2, 1, 2, 1, 2		X
CKA**	11	9, 8, 7, 6, 5, 4, 3, 2		X

* Base numbers resulting in a check digit of ten should be eliminated from the numbering system.

** These check digits have unique algorithms. An example of each is contained in the following section.

- Multipliers continue the series by repeating the basic factor. Example: Series 137. Number to be verified 1234567. Multiply factor 7137137.

METHODS OF CALCULATION:

Sum of Digits method of calculating a check digit using the algorithm for CK 1.

156742$\underline{9}$
121212
1+1+0+6+$\overline{1+4+4+4}$ = 21 ÷ 10 = 2 with remainder of 1

10 - 1 = $\underline{9}$

NOTE:

- Individual digits of the products are added in the algorithm.

Sum of Products method of calculation a check digit using the algorithm for CK 3.

475642$\underline{8}$
137137
4+21+35+$\overline{6+12+14}$ = 92 ÷ 10 = 9 remainder of 2.

10 - 2 = $\underline{8}$

NOTE:

- Individual products are added in the algorithm.

<u>Special method</u> of calculating the check digit for CK 9.

```
          1253462
          121212
     1+4+5+6+4+12 = 32
```

<u>Special method</u> for calculating the check digits for CKA.

```
          97532107
           765432
   63+42+25+12+6+2 = 150 ÷ 11 = 13 remainder of 7.
```

NOTE:

- If the remainder is ten or eleven do not precede it with a zero.

Additional Notes

EQUAL TABLE COMPARE

ETC LOC \ CON \ CON .... ■

Data in Element 1 is compared to elements 3 through N (if used). If no match is found, program control will jump the number of instructions specified in Element 2.

NOTES:

- The table may contain as few as 3 elements or as many elements as needed, limited only by the size of the screen.
- Trailing blanks in the table are not allowed.
- Leading and trailing blanks contained in Element 1 are ignored.

EXAMPLES:

Insuring that the operator enters county one, two, or six.

1	2	3	4	5	6	7	8	9	10	11	12
[	C	O	U	N	T	Y		X	]	9	1

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4
C	O	U	E	T	C	7	1	0	7	1	0		F	\	2	\	1	\	2	\	6	■
			G	T	O	7	2	0	7	2	0		1	■								
			E	R	R	7	3	0	7	3	0		C	T	Y	■						

NOTE:

- Since the operator entered a value unequal to Elements 3, 4, or 5 of Table 010 program control jumps two positions forward. The ERR 030 command is then executed.

Insuring that the operator enters either OHIO or MICHIGAN for the state:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	S	T	A	T	E		W	]	O	H	I	O					1

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2
S	T	A	E	T	C	9	4	0	9	4	0		F	\	2	\	O	H	I	O	\	M	I	C	H	I	G	A	N	·
			G	T	O	9	5	0	9	5	0		I	■																
			E	R	R	9	6	0	9	6	0		S	T	T	■														

JUMP

JMP CON ■

Move program control forward or backward, by the value in Element 1, relative to the current instruction.

NOTES:

- Maximum jump is 99 instructions.
- To jump backwards, use the minus symbol after the number, i.e. 2-.
- The JMP instruction operates only within the same extended memory field control groups, 1 and J, K and L, W and X, or Y and Z, i.e. jumping from group 1 to group L is not allowed.
- It is not possible to completely jump over a field program. The procedure is to move program control into the field program and then out via another jump.
- When jumping into another field program within the same extended memory field control group a count of one must be added for the field label.

EXAMPLES:

Assuming that information for some labels has already been stored in the I/O buffer multiple copies of the label can be produced as follows:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
[	N	O	.	O	F		C	O	P	I	E	S			6	X	]							1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.							1				
				4	5	6	7	8	9	0	1	2	3	4
NO.	SUB	101	101		R	1	\	R	1	\	R	1	■	
	CMP	102	102		R	1	\	<	=	\	F	\	4	■
	WRT	103	103		P	■								
	ADD	104	104		1	\	R	1	\	R	1	■		
	JMP	105	105		3	—	■							
	GTO	104												

Jumping into another field program to continue an editing operation.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.							1				
				4	5	6	7	8	9	0	1	2	3	4
	JMP	060	060		7	■								
	TSC	061	061		S	W	\	4	\	T	\	F	E	■
	MVE	062	062		*	*	*			\	R	4	■	
	JMP	063	063		4	■								
SP1	TSC	101	101		S	W	\	4	\	T	\	U	V	■
	MVE	102	102		*				*	\	R	4	■	
	PTI	103	103		R	4	\	5	■					
	JMP	104	104		4	0	—	■						
	GTO	105	105		1	■								

NUMERIC COMPARE

CMP LOC,CON\ REL\ LOC,CON\ CON ■

Element 1 is compared to Element 3, by the numeric relationship specified in Element 2. If the comparison fails, program control jumps the number of steps, forward or backward, specified in Element 4.

NOTES:

- Valid relationships for Element 2 are =, /=, > , < , >= and <=.
- "/=" Element 2 represents an unequal relationship.
- If Element 3 is a blank, only the "=" and "/=" relationships are valid.
- Commas and decimal points in Elements 1 and 3 are ignored.
- Embedded non-numerics cause truncation.
- Comparison is made from right to left and truncates when it reaches either an alpha character or a start of field.

EXAMPLES:

When the total accumulator equals 24 go to the next format; otherwise, go to the next field.

TA	2	4									

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4
T	A	X	C	M	P	4	1	0	4	1	0		T	A	\	=	\	2	4	\	2	■
			G	T	O	4	2	0	4	2	0		1	■								
			N	X	F	4	3	0	4	3	0		0	■								

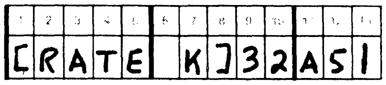
When Register 3, which contains total hours worked for the week, is less than or equal to 40.00 skip the overtime calculation.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
[	R	H	R		X	]						[	O	T	H		X	]					[	D	T	H		X	]					1

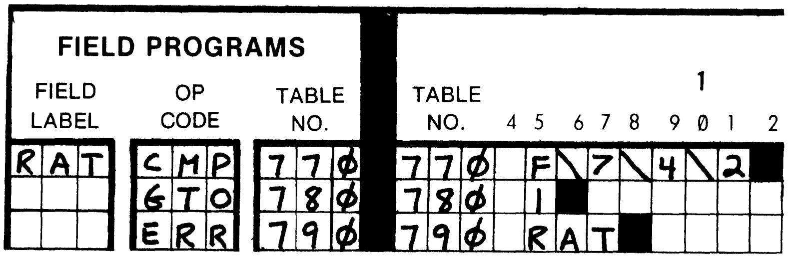
FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4	5	6	7
R	H	R	M	V	E	1	1	0	1	1	0		F	\	R	3	■								
			C	M	P	1	2	0	1	2	0		R	3	\	<	=	\	4	0	0	0	\	2	■
			G	T	O	1	3	0	1	3	0		2	■											
			S	U	B	1	4	0	1	4	0		R	3	\	4	0	0	0	\	R	4	■		
			G	T	O	1	5	0	1	5	0		1	■											

Comparing a field having an embedded non-numeric.



1	2	3	4	5	6	7	8	9	10	11	12	13
[	R	A	T	E		K	]	3	2	A	S	1



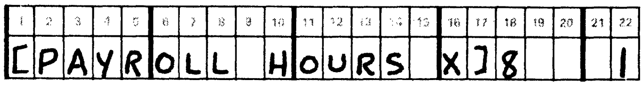
FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	10	11	12
R	A	T	C	M	P	7	7	0	7	7	0		F	\	7	\	4	\	2	■
			G	T	O	7	8	0	7	8	0		1	■						
			E	R	R	7	9	0	7	9	0		R	A	T	■				

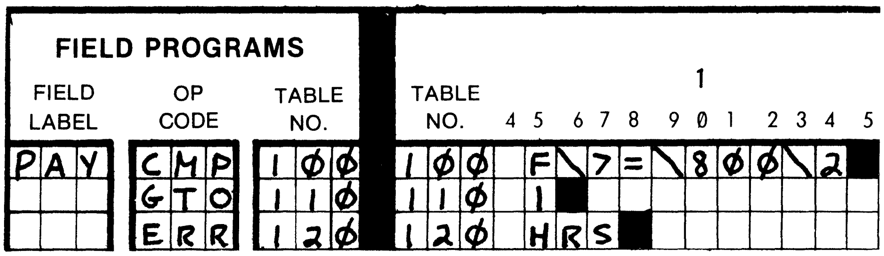
NOTE:

- The CMP instruction treats 3A5 as a 5, because it stops comparing the number when it comes to a non-numeric character. The GTO 080 command is then executed.

Forcing operator to enter hours to two decimal places.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	P	A	Y	R	O	L	L		H	O	U	R	S		X	]	8				1



FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	10	11	12	13	14	15
P	A	Y	C	M	P	1	0	0	1	0	0		F	\	7	=	\	8	0	0	\	2	■
			G	T	O	1	1	0	1	1	0		1	■									
			E	R	R	1	2	0	1	2	0		H	R	S	■							

NOTE:

- The eight entered in the field does not equal the "800" called for; therefore, ERR 120 is executed.

Here the digits have been entered correctly; but the required decimal point has been omitted.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	C	A	R	D		H	O	U	R	S		X	]	8	0	0			1

FIELD PROGRAMS										1					
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5
CAR	CMP	130	130		F	\	<	\	8	.	0	0	\	2	■
	ERR	140	140		C	R	D	■							
	GTO	150	150		1	■									

NOTE:

- Since decimals are ignored, the "8.00" equals "800" and no error results.

RANGE CHECK

RNG LOC \ CON \ CON \CON .... ■

Data in Element 1 is compared to pairs of elements, starting with the pair Element 3 and Element 4. If the value of Element 1 does not fall within these designated inclusive ranges, program control jumps the number of steps specified in Element 2.

NOTES:

- The first element of the pair must have a smaller value than the second i.e., Element 3 must be less than Element 4, Element 5 must be less than Element 6, etc.
- Element pairs can be negative numbers as long as each successive element is larger in value than the previous one.
- Comparison goes from right to left, until coming to an alpha character, a value that falls out of the range check, or the end of comparison, for that pair of values.
- All decimal points and commas are ignored.
- The table may contain as few as 1 pair of ranges or may contain as many as needed, limited only by the size of the screen.
- The negative sign is entered after the number, i.e. 2-.

EXAMPLES:

Insuring that the operator enters a tax rate between 1 and 4 or between 8 and 9:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	T	A	X			R	A	T	E		X	]	5	■

FIELD PROGRAMS										1						
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6
TAX	RNG	010	010		F	\	2	\	1	\	4	\	8	\	9	■
	GTO	020	020		1	■										
	ERR	030	030		T	A	X	■								

NOTE:

- Since the operator entered a value that does not fall within the parameters of the range check, program control advances 2 positions forward. The ERR 030 command is then executed.

Range check involving negative numbers.

FIELD PROGRAMS										1						
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6
	RNG	040	040		F	\	2	\	5	0	-	\	4	2	-	■

Range check involving positive and negative numbers:

FIELD PROGRAMS										1			
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3
	RNG	051	051		F	\	2	\	1	-	\	2	■

Range check involving embedded non-numerics:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	P	R	O	D	U	C	T		N	U	M	B	E	R		W	]	2	A	B	■

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
PRO	RNG	161	161		F	\	2	\	3	\	4	0	■
	GTO	162	162		1	■							
	ERR	163	163		P	R	O	■					

NOTE:

- Since comparison is made from right to left and stops upon reading an Alpha Character, the 2A3, in the field, will be read as a 3.

Additional Notes

SCAN THE I/O BUFFER

SCN CON \ CON ■

Starting at the byte address indicated by the index register, the I/O Buffer is scanned for a match with any character of Element 1. When found the index register is set to the relative byte address of the character in the I/O Buffer. A jump is taken according to Element 2 if a RS or no match if found. If no match is found and no RS encountered, the index register is set to 256.

NOTES:

- Element 1 can be any displayable string of characters except backslash or RS.
- Element 2 may be positive or negative.
- If a RS is encountered the index register is set to the position of the RS; then the jump is taken according to element 2.

EXAMPLE:

Scan the I/O Buffer for the letters A, B, or C.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	10
F	I	N	S	I	R	1	0	0	1	0	0		1	■				
			S	C	N	1	0	1	1	0	1		A	B	C	\	2	■
			G	T	O	1	0	2	1	0	2		1	■				
			G	T	O	1	0	3	1	0	3		2	■				

Before

IR	0	0	0	0	0	0	0	1

After

IR	0	0	0	0	1	0	1	1

I/O Buffer

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2
1	2	3	4	5	6	7	8	9	0	A	C	E	G	I	K	M	O	Q	S	Y	■

SET SWITCH

SSW

SSW CON\ CON ■

The switch specified in Element 1 is set to the value of Element 2.

NOTES:

- Element 1 is a constant from 1-8.
- Element 2 is a 0 or 1.
- When the program is loaded into extended memory, all switches are set to 0.
- A switch stays set until it is reset by another SSW instruction or an instruction using SW for a location.
- Include REG in the list of OP CODES when specifying the SSW command during program generation.

EXAMPLE:

If the operator keys in a tax rate greater than zero set Switch 3 to 1.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	10	11	12
T	A	X	M	V	E	6	1	0	6	1	0		F	\	R	0	■			
			C	M	P	6	2	0	6	2	0		F	\	>	\	0	\	2	■
			S	S	W	6	3	0	6	3	0		3	\	1	■				
			G	T	O	6	4	0	6	4	0		1	■						

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	T	A	X		R	A	T	E		X	]	4	.	5	1

	8	7	6	5	4	3	2	1
SW	0	0	0	0	1	0	0	1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	T	A	X		R	A	T	E		X	]	4	.	5	1

	8	7	6	5	4	3	2	1
SW	0	0	0	0	1	1	0	1

To set Switch 5 to one and the rest to zero with one instruction.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9
			M	V	E	2	0	0	2	0	0		4	\	S	W	■

Before

	8	7	6	5	4	3	2	1
SW	0	1	0	0	0	0	0	1

After

	8	7	6	5	4	3	2	1
SW	0	0	0	0	0	1	0	0

TABLE LOOK UP

TLU LOC \ CON \ CON \ CON .... ■

The data in Element 1 is compared to the first member of element pairs starting with Elements 3 & 4. If a match occurs, the second member of the element pair is placed into the location specified by Element 1. If no match occurs, program control jumps the number of steps specified by Element 2.

NOTES:

- Commas, decimal points, minus symbols and alpha characters are compared.
- Leading and trailing blanks in Element 1 are ignored.
- The table can contain as few as 1 pair or contain as many pairs as needed, limited only by the size of the screen.
- Overflow will occur, if the second member of the element pair exceeds the capacity of Element 1.

EXAMPLES:

If the operator enters a 1, 2, or 3 place "TAX", "FRT", or "MISC" in the field, respectively. If the operator enters anything else place "NONE" in the field.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7
CHA	TLU	560	560		F	\	2	\	1	\	T	A	X	\	2	\	F	R	T	\	3	\	M	I	S	C	■
	GTO	561	561		1	■																					
	MVE	562	562		N	O	N	E	■																		
	GTO	561																									

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	C	H	A	R	G	E		X	]	3				1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	C	H	A	R	G	E		X	]	M	I	S	C	1

NOTE: Since the operator Keyed in a code matching Element 7, Element 8 replaced the 3 in the CHARGE field.

As product codes are entered in the code field the corresponding price is placed in register 1 for use later on.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	P	R	O	D	U	C	T		C	O	D	E		X	]		1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7
PRO	TLU	150	150		R	1	\	2	\	1	\	1	7	5	\	2	\	1	0	5	\	3	\	2	5	0	■
	GTO	160	160		1	■																					
	GTO	170	170		0	■																					

TEST CHARACTER

TSC

TSC LOC\ CON \ REL\ CON ■

Compare each character in Element 1 to the list of characters in Element 4, according to the relationship in Element 3, True or False. If Element 3 is true, Element 1 can contain only characters specified in Element 4. If Element 3 is false, Element 1 must not contain any characters specified in Element 4. If the comparison fails, program control jumps the number of commands stated in Element 2.

NOTES:

- Element 3 is a T or F (true or false).
- Repeating characters in Element 4 is redundant. i.e., 2 is as effective as 22.

EXAMPLE:

Insure that the operator enters no commas or periods.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	A	M	O	U	N	T		X	]	1	,	5	0	0		1

FIELD PROGRAMS

											1			
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3
AMO	TSC	010	010		F	\	2	\	F	\	.	,	■
	GTO	020	020		1	■							
	ERR	030	030		A	M	T	■					

NOTE:

- Since the operator did enter a comma and the F in Element 3 means that Element 1 must not contain any of the characters in Element 4, program control jumps to the ERR 030 instruction.

Insure that the operator keys in all four digits of the rate and leaves no blank spaces or minus symbols.

1	2	3	4	5	6	7	8	9	10	11	12	13
[	R	A	T	E		X	]	4	1	2		1

FIELD PROGRAMS										1			
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3
RAT	TSC	640	640		F	\	2	\	F	\		—	■
	GTO	650	650		1	■							
	ERR	660	660		R	T	E	■					

Another method of insuring that the operator keys in all four digits of the field.

1	2	3	4	5	6	7	8	9	10	11	12	13
[	R	A	T	E		X	]	4	1	2		1

FIELD PROGRAMS										1										2	
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RAT	TSC	670	670		F	\	2	\	T	\	0	1	2	3	4	5	6	7	8	9	■
	GTO	680	680		1	■															
	ERR	690	690		R	T	E	■													

NOTE:

- Since a space is not included in table 070, ERR 090 is the next instruction executed. The operator either keyed in a space or failed to enter all four digits of the rate.

TEST SWITCH

TSW CON \ CON \ CON ■

The switch specified in Element 1 is tested for the value specified in Element 2. If equal, program control passes to the next instruction. If unequal program control jumps the number of instructions stated in Element 3.

NOTES:

- Element 1 is a constant from 1 thru 8.
- Element 2 is a constant of either 0 or 1.
- Program initialization sets all switches to zero.
- Include REG in the list of OP CODES entered during program generation.

EXAMPLE:

Switch 1 is set by the operator with a yes or no at the beginning of processing. As each invoice is entered this switch is checked before entering the tax calculation routine.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
[	A	R	E		I	N	V	O	I	C	E	S		T	A	X	A	B	L	E		W	]				1		Y	E	S		O	R		N	O	
[	G	R	O	S	S		A	M	O	U	N	T		X	]							1																
[	T	A	X											X	]					1																		
[	N	E	T		A	M	O	U	N	T				X	]							■																

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3
A	R	E	M	V	E	1	0	0	1	0	0		F	\	S	W	■				
			G	T	O	1	0	1	1	0	1		1	■							
				⋮																	
G	R	O	M	V	E	7	0	0	7	0	0		F	\	R	2	■				
			T	S	W	7	0	1	7	0	1		1	\	0	\	3	■			
			E	X	F	7	0	2	7	0	2		2	■							
			J	M	P	7	0	3	7	0	3		5	■							
			E	X	F	7	0	1													
			M	P	Y	8	0	0	8	0	0		R	2	\	5	\	F	\	2	■
			A	D	D	8	0	1	8	0	1		F	\	R	2	\	R	2	■	
			E	X	F	7	0	1													
			M	V	E	9	0	0	9	0	0		R	2	\	F	■				
			G	T	O	9	0	1	9	0	1		0	■							

CONVERT TO BINARY

CVB

CVB LOC ■

Converts the data specified by Element 1 to a binary number and places the result in the Index Register.

NOTES:

- Any positive decimal number between 1 and 256 can be converted to binary.
- Conversion takes place from right to left.
- Minus signs and spaces are ignored.

EXAMPLES:

Convert the contents of the quantity field to binary.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	Q	U	A	N	T	I	T	Y		X	]	2	1	0	-		1

FIELD PROGRAMS FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
QUA	CVB	550	550		F	

IR = 11010010

Convert the contents of Register 1 to binary.

R1 = AC050

FIELD PROGRAMS FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7
	CVB	663	663		R	1	

IR = 00110010

Additional Notes

CONVERT TO DECIMAL

CVD LOC ■

Converts the index register from a binary to a decimal number and places the result in the location specified by Element 1.

NOTES:

- The result is left justified in the location and padded with blanks if necessary in the specified location.

EXAMPLES:

Load Register 0 with the contents of the index register.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2
	CVD	150	150		R	0	■					

	Before	After
IR	00110010	00110010
R0	725	50

NOTE:

- The index register contains a one byte representation of the decimal number 50, but it cannot be used until it is converted and placed in a register.

EXAMPLES:

Convert the contents of the quantity field to binary.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	Q	U	A	N	T	I	T	Y		X	]	2	1	0	5		1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
QUA	CVB	550	550		F	

Before

IR

After

IR 11010010

Convert the contents of Register 1 to binary.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7
	CVB	663	633		R	1	

Before

R1 AC050

IR

After

R1 AC050

IR 00110010

DUPLICATE

DUP CON ■

The duplicate instruction can perform one of two functions; depending on the operator action. If data is keyed into the field it will replace Element 1 in the table. If the TAB/SKIP Key is depressed, instead, Element 1 will be replace the previous contents of the field.

NOTES:

- The data field must be the same size as the space allocated in the table.
- An OFL error occurs, when Element 1 is larger than the data field.

EXAMPLES:

The DUP instruction can be used to give the operator a choice between using the previous date entered or keying in a new date.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	D	A	T	E		W	]									

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
DAT	DUP	020	020										

Table after keying in a date.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
	DUP	020	020		1	0	-	1	7	-	7	4	

Table after depressing TAB/SKIP in position 1.

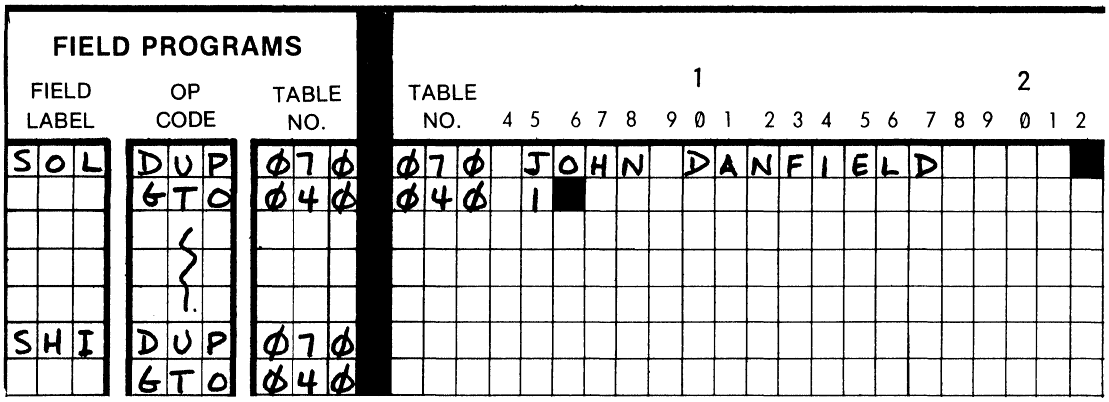
FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
	DUP	020	020										

NOTE:

- The table is empty because the original instruction is still intact. If a date had been keyed in, it would still be there.

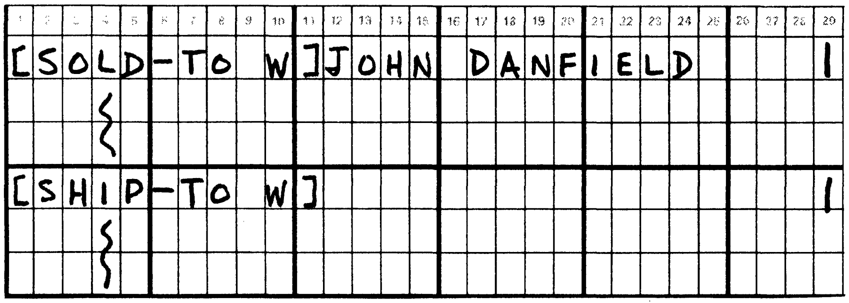
Displaying the data keyed into one field in a subsequent field.



FIELD PROGRAMS

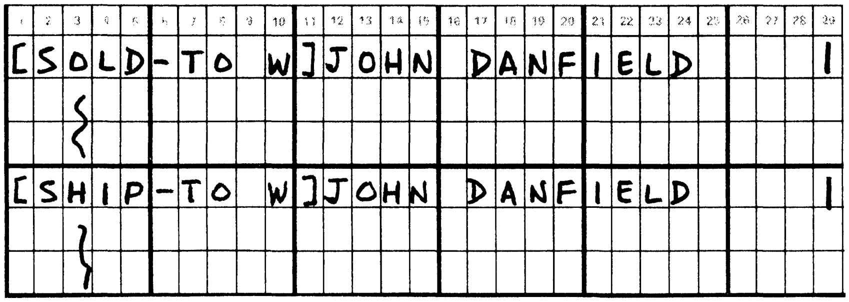
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2
SOL	DUP	070	070	JOHN DANFIELD ■
	GTO	040	040	1■
	{			
SHI	DUP	070		
	GTO	040		

Before depressing TAB/SKIP.



```
[SOLD-TO W]JOHN DANFIELD    I
   {
[SHIP-TO W]                 I
   {
```

After depressing TAB/SKIP.



```
[SOLD-TO W]JOHN DANFIELD    I
  {
[SHIP-TO W]JOHN DANFIELD    I
  {
```

Additional Notes

GET FROM BUFFER

GET

GET LOC\ CON\ CON ■

The numbers of characters specified in Element 3 is transferred from the I/O Buffer into Element 1 starting at the position specified in Element 2.

NOTES:

- The GET instruction terminates upon reaching a Record Separator in the I/O Buffer, or when the number of characters specified by Element 3 transfers to the location specified in Element 1.
- Element 2 and 3 must be between 1 and 256.
- The field pointer follows the data being transferred from I/O Buffer.
- If Element 1 is a data field, and Element 3 exceeds the capacity of the data field, data transfer continues into additional data fields.
- If Element 1 is the subtotal accumulator, and Element 3 is greater than 11, the additional data gets transferred into the total accumulator.
- If Element 1 is a register, and Element 3 is greater than 11, additional characters will transfer into the next register.
- Do not exceed the capacity of the ten registers or the total accumulator.
- All printer control characters, HT, CR, LF, LT, FF, and ESC, in the data, will be treated as normal data characters.

EXAMPLES:

Transfer the characters in positions 1 thru 5 of the I/O Buffer to the Previous Balance field.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
[	N	O	.		O	F		I	T	E	M	S		W	]	2	1	0	1	1
[	P	R	E	V	.	B	A	L	.		N	]						1		

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
NO.	EXF	005	005		1	■				
	GET	010	010		F	\	1	\	5	■
	GTO	005								

Before

	1	2	3	4	5	6	7	8	9	10
I/O Buffer	4	3	5	7	5					

After

	1	2	3	4	5	6	7	8	9	10
I/O Buffer	4	3	5	7	5					

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	P	R	E	V	.	B	A	L	.		N	]						1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	P	R	E	V	.	B	A	L	.		N	]	4	3	5	7	5	1

Transfer the contents of the I/O Buffer into Registers R0 thru R9.

FIELD PROGRAMS																					
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.									1			
												4	5	6	7	8	9	0	1	2	3
			G	E	T	0	2	5	0	2	5		R	0	\	1	\	1	1	0	■

I/O Buffer

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234
APPLES     ORANGES    GRAPES     PEARS      MELON      BANANA
  PLUMS      AVOCADO    GRAPEFRUIT PEACHES    ■
```

R0	A	P	P	L	E	S					
R1	O	R	A	N	G	E	S				
R2	G	R	A	P	E	S					
R3	P	E	A	R	S						
R4	M	E	L	O	N						
R5	B	A	N	A	N	A					
R6	P	L	U	M	S						
R7	A	V	O	C	A	D	O				
R8	G	R	A	P	E	F	R	U	I	T	
R9	P	E	A	C	H	E	S				

NOTE:

- Since Element 1 is Register 0 and Element 3 contains more than 11 elements, the transfer of data continues until all 110 characters are transferred into Registers 0 thru 9.

Moving the contents of locations 9 thru 30 of the I/O Buffer to both the Subtotal and Total Accumulators.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.
	GET	020

TABLE NO.	4	5	6	7	8	9	10	11	12
020		S	A	\	9	\	2	2	

I/O Buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
								2	5	9	5								5	0	5	3	5	-					

Before

SA											
TA											

After

SA	2	5	9	5							
TA	5	0	5	3	5	-					

Terminating the transfer of data from the I/O Buffer upon the recognition of an RS code.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.
AMO	GET	030

TABLE NO.	4	5	6	7	8	9	10
030		F	\	1	\	8	

I/O Buffer

1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6		2	5	3	3

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	A	M	O	U	N	T		X	]									|

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	A	M	O	U	N	T		X	]	1	2	3	4	5	6			|

GET FROM THE I/O BUFFER INDEXED

G T I

GTI LOC \ CON■

The number of data characters specified by Element 2 is transferred from the I/O Buffer, according to the position indicated by the index register, to a location named in Element 1.

NOTES:

- Element 2 must be a positive integer from 1 to 256.
- After the GTI is executed the index register is incremented by Element 2.
- The field pointer follows the data being transferred to the I/O Buffer.
- The location in Element 1 is <u>not</u> space filled if its capacity is greater than the number of characters in Element 2.

EXAMPLE:

Transfer the first 8 characters of the I/O Buffer to Register 5.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
	SIR	150	150		1	■							
	SUB	151	151		R	5	\	R	5	\	R	5	■
	GTI	152	152		R	5	\	8	■				
	JMP	153	153		6	–	■						

	1	2	3	4	5	6	7	8	9	1 0	1
I/O Buffer	4	3	2	7	3	3			9	7	5

<u>Before</u>

R5	J	A	C	K		S	M	I	T	H	

<u>After</u>

R5	4	3	2	7	3	3					

Additional Notes

INCREMENT INDEX REGISTER

IXR CON ■

The index register is incremented or decremented by the value in Element 1.

NOTES:

- Element 1 must be a numeric constant from 127- to 127 decimal.

EXAMPLE:

Put a carriage return at every tenth position of the I/O Buffer.

FIELD PROGRAMS																
										1						
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6
	SIR	601	601		1	■										
	IXR	602	602		9	■										
	PTI	603	603		C R	\	1	■								
	CVD	604	604		R	0	■									
	CMP	605	605		R	0	\	=	\	2	5	0	\	3	-	■
	GTO	606	606		1	■										

Additional Notes

LOAD FROM A TABLE

LOD LOC,CON \ CON ■

Transfer to Element 1 the number of characters specified in Element 2, that are located in Element 3. Element 2 must equal the number of positions in Element 3.

NOTES:

- If Element 1 is a data field, and Element 3 exceeds the capacity of the data field, data transfer continues into additional data fields in the CRT format, while the field pointer moves with the data. Data transfer will continue until the last data field on the screen has been filled or the number of characters specified has been transferred.
- If Element 1 is the subtotal accumulator, and Element 2 is greater than 11, additional characters are transferred into the total accumulator.
- Do not load more than 11 characters into the total accumulator.
- If Element 1 is a register, and Element 2 is greater than 11, additional characters transfer into subsequent registers.
- If Element 1 is the switch, and Element 2 is greater than one, additional data characters transfer into the registers, starting with R0.
- Do not exceed the capacity of the ten registers.
- All printer control characters, HT, CR, LF, VT, FF, and ESC will be treated as normal data characters.

EXAMPLE:

Key data into the "SOLD TO" field and redisplay the data in the "SHIP TO" field.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
[	S	O	L	D		T	O		W	]													7	1
[	S	H	I	P		T	O	6	W	]													7	1

FIELD PROGRAMS																		1										2		
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
S	O	L	S	T	R	0	1	0	0	1	0		F	\	1	2	\													■
			G	T	O	0	2	0	0	2	0		1	■																
S	H	I	L	O	D	0	1	0																						
			G	T	O	0	2	0																						

NOTE:

- Both the STR and LOD instructions use Table No. 010.

LOAD FROM MEMORY ADDRESS

L M A

LMA   LOC\ CON\ CON\ CON ■

The data at the memory location specified by Elements 2 and 3 is transferred to the location in Element 1. Element 4 specifies the number of bytes to be transferred.

NOTES:

- The field pointer follows the data being transferred from memory.
- Element 2 must be a numeric address from 0 thru 31, specifying the page location in decimal.
- Element 3 is the byte address, from 1-256, indicating the starting position of the data to be transferred in decimal.
- Element 4 must be a positive numeric constant indicating the number of characters to be transferred.
- Data stored in memory from previous programs may be accessed by following programs if the memory address has not been exceeded during Loading.

EXAMPLES:

Seven data characters stored in memory page 15, location 030 are to be loaded to a field.

FIELD PROGRAMS														
										1				
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4
	LMA	600	600		F	\	1	5	\	3	0	\	7	

An amount stored in page 14, location 100 is to be transferred to the Subtotal Accumulator.

FIELD PROGRAMS																	
										1							
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6	7
	LMA	601	601		S	A	\	1	4	\	1	0	0	\	1	1	

MOVE

M V E

MVE LOC,CON\ LOC ■

The data in Element 1 is moved to the location specified in Element 2. Data is moved one character at a time, starting with left most character. If necessary, Element 2 is filled with trailing blanks.

NOTES:

- Only the left-most character moves to Element 2 if Element 2 is SW.
- If Element 2 is a register or an accumulator, up to 11 characters from Element 1 may move to Element 2.
- If Element 1 is a literal, F-T, place a space as the first character of Element 1.

EXAMPLES:

Move the constant of 4.00 to the tax field.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1
			E	X	F	0	0	5	0	0	5		1	■					
			M	V	E	0	1	0	0	1	0		4	.	0	0	\	F	■
			E	X	F	0	0	5											

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	T	A	X		R	A	T	E		N	]						1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	T	A	X		R	A	T	E		N	]	4	.	0	0		1

Move the constant "ERROR" to the subtotal accumulator.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3
	MVE	020	020	ERROR\SA

Before

SA

After

SA ERROR

Moving more than 11 characters to Register 2.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20
	MVE	030	030	1234567890-9\R2

Before

R2

After

R2 1234567890-

MOVE MEMORY

MVM CON\ CON\ CON, IR\ CON \ CON, IR ■

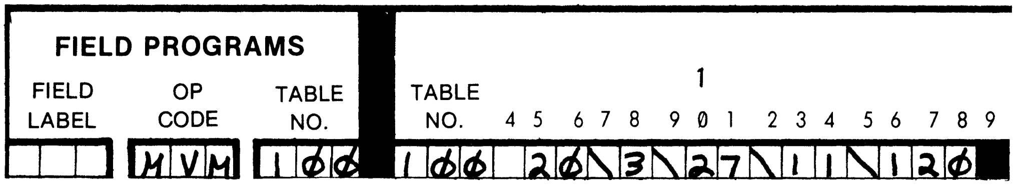
Starting at the memory location specified by elements 2 and 3, data is transferred to the memory location starting in elements 4 and 5 until the number of bytes in element 1 is satisfied.

NOTES:

- Element one is a numeric constant from 1-256 decimal.
- Elements 3 and 5 must be either a constant between 1 and 256 decimal, or the index register, specifying the byte address.
- Elements 2 and 4 must be a constant, between 0 and 31, specifying the page location in decimal.
- Where IR is specified as an element the contents of the index register are used as the address.
- The index register does not increment with the MVM instruction.
- Do not move into page 0 or page 16, bytes 0-32.

EXAMPLES:

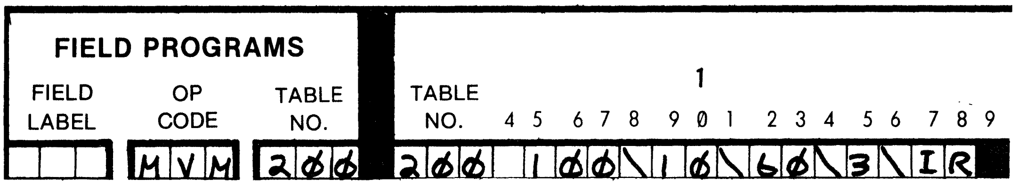
Move 20 bytes from page 3, bytes 27 thru 46 to page 11, bytes 120 thru 139.



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9
	MVM	100	100	20\3\27\11\120

Using the value of the index register as the receiving byte address, move 100 bytes from page 10, byte 60 to page 3, byte location specified by contents of index register.



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9
	MVM	200	200	100\10\60\3\IR

PUT TO BUFFER

PUT

PUT LOC,CON\ CON\ CON ■

Copy the contents of Element 1 into the I/O Buffer starting at the position in Element 2. Element 3 contains the number of characters to be copied.

NOTES:

- The PUT instruction terminates when the number of characters transferred exceeds the capacity of the I/O Buffer or when the number of characters specified in Element 3 is transferred into the I/O Buffer.
- Elements 2 and 3 must be between 1 and 256.
- The field pointer follows the data being transferred to the I/O Buffer.
- If Element 1 is a data field, and Element 3 exceeds the capacity of the data field, data transfer continues from additional fields. The field pointer follows the data.
- If Element 1 is the subtotal accumulator, the Element 3 is greater than 11, additional data is transferred from the total accumulator.
- If Element 1 is "RS", Record Separator, the single RS character moves into the I/O Buffer.
- If Element 1 is a one character literal, except F-T, place a blank as the first character of Element 1.
- If Element 1 is a register, and Element 3 is greater than 11, additional characters will be transferred from subsequent register.
- Do not exceed the capacity of the ten registers or the accumulator.
- Printer control characters HT, VT, CR, LF, FF, and ESC will be treated as normal data characters.
- The "¬" and "\ " are ignored when they appear on the CRT.

EXAMPLES:

Put the contents of Registers R0 thru R9 into the I/O Buffer.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
	PUT	010	010		R	0	\	1	\	1	1	0	■

R0	3	5	0	0							
R1	7	1	0	0							
R2	6	2	5	0							
R3	8	4	7	5							
R4	2	5	0	-							
R5	1	3	2	5							
R6	4	5	8	2							
R7	1	0	7	8	5						
R8	1	8	2	0							
R9	6	1	5	1	5						

I/O Buffer

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234
3500       7100       6250       8475       250-       1325     
  4582       10785      1820       61515      ■                 
```

NOTE:

- Since the field pointer moves with the PUT instructions and the registers are located next to one another in memory all the registers are put into the I/O Buffer with one PUT instruction.

Format with "¬" and "\".

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	Q	U	A	N	T	I	T	Y		X	]	1	0	0	¬	\	I				
[	D	E	S	C	R	I	P	T	I	O	N		M	]	B	O	X	E	S		■

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
QUA	PUT	020	020		F	\	I	\	S	■

I/O Buffer

1	2	3	4	5
1	0	0	B	O

NOTE:

- The PUT command ignores the "¬" and "\" when they appear on the CRT.

Format with CR and HT

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
QUA	PUT	030	030		F	\	I	\	S	■

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	Q	U	A	N	T	I	T	Y		X	]	1	0	0	[	C	]	CR	HT	I	
[	D	E	S	C	R	I	P	T	I	O	N		M	]	B	O	X	E	S		■

I/O Buffer

1	2	3	4	5
1	0	0	CR	HT

Place the literal "Invoice Number" with a Record Separator following in positions 10 thru 25 of the I/O Buffer.

FIELD PROGRAMS

										1										2						
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
	PUT	011	011			I	N	V	O	I	C	E		N	U	M	B	E	R	\	1	0	\	1	5	■
	PUT	012	012		R	S	\	2	5	\	1	■														

I/O Buffer

									1										2									
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
										I	N	V	O	I	C	E		N	U	M	B	E	R	■				

Moving the contents of both the Subtotal and Total Accumulator to the I/O Buffer.

FIELD PROGRAMS

										1		
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2
	PUT	123	123		S	A	\	1	\	2	2	■

<u>Before</u>

SA	1	0	0	2	5	-					
TA	5	1	1	3	7	5					

<u>After</u>

SA	1	0	0	2	5	-					
TA	5	1	1	3	7	5					

I/O Buffer

									1										2		
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
1	0	0	2	5	-						5	1	1	3	7	5					

PUT TO THE I/O BUFFER INDEXED

PTI

PTI LOC,CON \ CON ■

The number of data characters specified in Element 2 is transferred to the I/O Buffer, according to the index register, from the location specified by Element 1.

NOTES:

- Element 2 must be a constant from 1 to 256.
- Data transfer continues from the location in Element 1 until Element 2 is satisfied, or a record separator is detected.
- If Element 1 is a 1 character literal, except F thru T, this character will be in the I/O Buffer the number of times specified in Element 2.
- After the PTI instruction is executed the index register is incremented by Element 2.

EXAMPLES:

Fill the I/O Buffer with blanks.

FIELD PROGRAMS										
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
	SIR	301	301		1	■				
	PTI	302	302			\	2	5	6	■
	GTO	301								

Put the first 40 characters of the field into positions 10-49 of the I/O Buffer.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9
	SIR	401	401		1	0	■		
	PTI	402	402		F	\	4	0	■
	JMP	403	403		5	-	■		

SET INDEX REGISTER

SIR CON ■

The index register is set to the binary value of Element 1.

NOTES:

- Element 1 must be a numeric constant from 1 to 256 decimal.

EXAMPLE:

Put blanks into the I/O Buffer in positions 60-69.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9
	SIR	200	200		6	0	■		
	PTI	201	201			\	1	0	■
	GTO	202	202		1	■			

Before

IR 00110000

After

IR 00111100 = 60 decimal

Additional Notes

STORE AT MEMORY ADDRESS

SMA LOC,CON \ LOC\ LOC\ CON ■

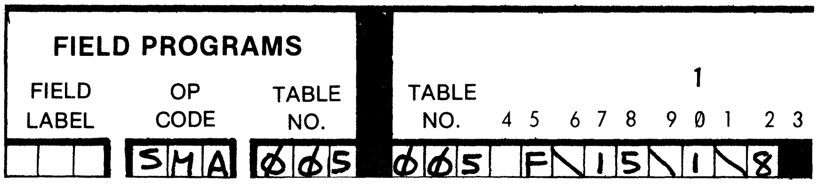
The number of data characters indicated by Element 4 is transferred from Element 1 to the memory address specified in Elements 2 and 3.

NOTES:

- Element 2 is a decimal page location from 0-31.
- Element 3 is a decimal byte address from 1-256 indicating the starting position where the data is to be stored.
- The data in Element 1 is stored starting from the left and data transfer continues until Element 4 is satisfied.
- Do not store information on page 16, bytes 1-32.
- An "RS" in Element 1 will indicate a record separator is to be stored in memory.
- Data can be stored in memory by one program and accessed by another as long as the memory requirement does not exceed the storage address.

EXAMPLES:

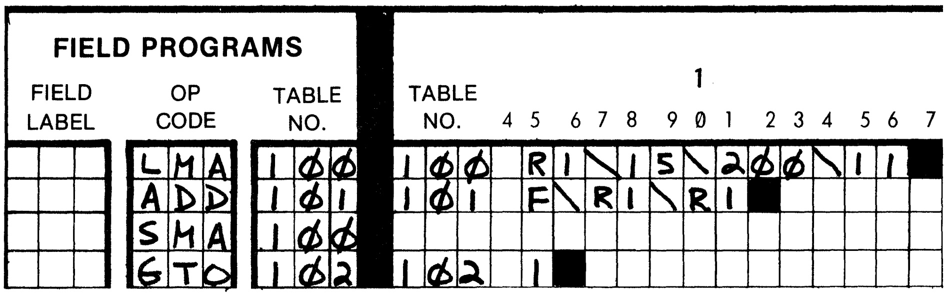
The eight data characters entered in the field are to be stored in pages 15, location 001.



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3
	SMA	005	005	F\15\1\8

A total is being stored on page 15, location 200. It can be updated with the contents of the field as follows:



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4 5 6 7 8 9 10 1 2 3 4 5 6 7
	LMA	100	100	R1\15\200\11
	ADD	101	101	F\R1\R1
	SMA	100		
	GTO	102	102	1

STORE IN A TABLE

S T R

STR LOC \ CON \ CON ■

Transfer the number of characters specified in Element 2 from Element 1 to Element 3.

NOTES:

- If Element 1 is a field, data is transferred intil the last data field or the number of characters specified in Element 1 has been transferred.
- If Element 1 is a data field, and Element 2 exceeds the capacity of the data field, data transfer continues from additional data fields in the CRT format. The field pointer moves with the data.
- If Element 1 is the subtotal accumulator, and Element 2 is greater than 11, additional characters are transferred from the total accumulator.
- Do not store more than 11 characters from the total accumulator.
- If Element 1 is a register, and Element 2 is greater than 11, additional characters will transfer from subsequent registers.
- If Element 1 is the switch, and Element 2 is greater than ONE, additional data characters transfer from the registers, starting with R0.
- Do not exceed the capacity of the ten registers.
- All printer control characters, HT, VT, CR, LF, FF and ESC will be treated as normal data characters, from the transferred data. "¬" and "\" are stripped.

EXAMPLE:

Store the product code in the field to the table.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
[	A	C	C	O	U	N	T		N	O		X	]	4	3	2	5	0	0	1

Before

FIELD PROGRAMS

																			1					
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3	4	5	
A	C	C	S	T	R	3	0	1	3	0	1		F	\	6	\							■	
			G	T	O	3	0	2	3	0	2		1	■										

Table 301 after execution

									1					
TABLE NO.			4	5	6	7	8	9	0	1	2	3	4	5
3	0	1		F	\	6	\	4	3	2	5	0	0	■

EDT LOC\ CON\ LOC ■

The data in Element 1 is edited according to the mask in Element 2 and place in the location specified in Element 3. The result in Element 3 is right justified.

NOTES:

- Mask control characters:

  9 Represents a data position in the mask.

  . Represents a decimal.

  , Represents a comma.

  * As the first character of the mask it causes left fill of leading spaces with an asterisk.

  $ As the first character it causes a $ to float just to the left of the first significant digit.

  \- As the last character of the mask it causes the sign of Element 1 to be placed in Element 3 as the last character. A space for a positive number, and a minus symbol "-" for a negative number.

- Zeros will follow to the right of the decimal, if necessary, up to the first significant digit.

- Overflow will occur if Element 1 exceeds the capacity of the mask, Element 2.

- If Element 2 is larger than Element 3 erroneous data will be placed in Element 3.

EXAMPLES:

Editing a field.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6
UNI	EDT	040	040		F	\	9	9	9	9	9	9	9	\	F	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	U	N	I	T	S		X	]	1	5	8					1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	U	N	I	T	S		X	]					1	5	8	1

Editing a signed field.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5
HOU	EDT	050	050		F	\	9	9	.	9	9	-	\	F	

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	H	O	U	R	S		X	]	2	8	0	0			1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	H	O	U	R	S		X	]	2	8	.	0	0		1

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	H	O	U	R	S		X	]	8	0	0	-			1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	H	O	U	R	S		X	]		8	.	0	0	-	1

Zero suppression.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8
	ADD	1B5	1B5		0	\	F	\	F	■								
	EDT	1B7	1B7		F	\	9	9	,	9	9	9	.	9	9	\	F	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	A	M	T		X	]	0	0	0	0	9	4	2			1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	A	M	T		X	]						9	.	4	2	1

Editing a field with a floating dollar sign.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9
PAY	EDT	010	010		F	\	$	9	9	,	9	9	9	.	9	9	\	F	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	P	A	Y	M	T		X	]	1	2	3	4	5						1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	P	A	Y	M	T		X	]				$	1	2	3	.	4	5	1

Editing a field with asterisk protest.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0
DOL	EDT	020	020		F	\	*	9	9	,	9	9	9	.	9	9	\	T	A	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	D	O	L	L	A	R	S		X	]	1	2	3	4	5		1

TA											

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	D	O	L	L	A	R	S		X	]	1	2	3	4	5		1

TA	*	*	*	*	1	2	3	.	4	5	

Editing a field with a floating dollar sign and asterisk protect.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7
PRO	EDT	155	155		F	\	*	9	9	9	.	9	9	\	F	■	
	EDT	156	156		F	\	$	9	9	9	9	9	9	9	\	F	■
	GTO	157	157		1	■											

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	N	E	T		X	]	2	5	0	0	0					1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	N	E	T		X	]	$	*	2	5	0	.	0	0		1

MINUS OVER PUNCH

MOP

MOP LOC\ LOC\ CON ■

The data at the location specified by Element 1 is edited for sign. The sign character is dropped and the units character is replaced with an overpunch character and placed in Element 2 right justified, and left filled with the character in Element 3.

NOTES:

- Arithmetic computation may not be performed on a field after the minus overpunch instruction has been executed.
- The overpunch characters for negative numbers are:

  0 - Non Displayable (ASCII OCTAL 175)
  1 - J
  2 - K
  3 - L
  4 - M
  5 - N
  6 - O
  7 - P
  8 - Q
  9 - R

EXAMPLE:

Minus overpunch the field and left fill with zeros.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
BAL	MOP	010	010		F	\	F	\	0	■
	GTO	020	020		1	■				

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	B	A	L		X	]	1	2	3	4	-				1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[	B	A	L		X	]	0	0	0	0	1	2	3	M	1

NOTE:

- M is the overpunch character that replaces "4-".

RIGHT JUSTIFY

RTJ LOC \ CON ■

The data in Element 1 is right justified and left filled with the character in Element 2.

NOTES:

- The characters "⌉" and "\" are invalid entries for Element 2.

EXAMPLES:

Right justify the subtotal accumulator and fill with zeros.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9
	RTJ	010	010		S	A	\	0	■

Before

SA	6	2	5	1							

After

SA	0	0	0	0	0	0	0	6	2	5	1

Right justify the field and fill with asterisks.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
	RTJ	020	020		F	\	*	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	F	I	E	L	D		X	]	1	2	3					1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	F	I	E	L	D		X	]	*	*	*	*	1	2	3	1

Right justify a field that contains a negative value and left fill with spaces.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
NO.	RTJ	030	030		F	\		

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	N	O	.		X	]	4	3	2	1	-					1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	N	O	.		X	]					4	3	2	1	-	1

SHIFT

SHF LOC\ CON\ CON ■

Data in Element 1 is shifted in the direction of Element 2 the number of places specified in Element 3.

NOTES:

- Element 2 is an R or L for a shift right or left, respectively.
- Element 3 is a constant from 1-9.
- Any characters shifted beyond the capacity of Element 1 are not retained; therefore, these characters should be stored prior to the execution of the SHF instruction if they are required further on in the program.

EXAMPLES:

Shift the contents of the field six positions to the right.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10
	SHF	110	110		F	\	R	\	6	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	F	I	E	L	D		X	]	1	2	3							1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[	F	I	E	L	D		X	]							1	2	3	1

Save the original contents of Register 1; then shift Register 1 left seven positions.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	10	11
	MVE	120	120		R	1	\	R	2	■	
	SHF	121	121		R	1	\	L	\	7	■

Before

R1							4	6	2	5	3
R2											

After

R1	6	2	5	3							
R2							4	6	2	5	3

TAB COMPRESSION

T C P

TCP LOC ■

A " \ " is inserted in the position following the last non blank character of Element 1.

NOTES:

- No action occurs, if there are no trailing blanks.
- Only the data and the horizontal tab character are released to the output device.

EXAMPLES:

Tab compression of the name field.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
NAM	TCP	010	010		F	■
	GTO	020	020		1	■

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	N	A	M	E		W	]	J	O	N	E	S							|

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	N	A	M	E		W	]	J	O	N	E	S	\						|

Tab compress the "NAME", "ADDRESS", and "CITY-STATE" fields.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
NAM	TCP	030	030		F	■
	GTO	040	040		1	■
ADD	TCP	030				
	GTO	040				
CIT	TCP	030				
	GTO	040				

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
[	N	A	M	E								W	]	J	O	E		D	O	E											1
[	A	D	D	R	E	S	S					W	]	1	2	3		M	A	I	N		S	T	R	E	E	T			1
[	C	I	T	Y	-	S	T	A	T	E		W	]	R	O	M	E	,		N	.	Y	.								1
[	Z	I	P									N	]	1	2	4	1	5	■												

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
[	N	A	M	E								W	]	J	O	E		D	O	E	\										1
[	A	D	D	R	E	S	S					W	]	1	2	3		M	A	I	N		S	T	R	E	E	T	\		1
[	C	I	T	Y	-	S	T	A	T	E		W	]	R	O	M	E	,		N	.	Y	.	\							1
[	Z	I	P									N	]	1	2	4	1	5	■												

NOTE:

- The following is outputted to the I/O Buffer.

JOE DOE $\overset{H}{T}$ 123 MAIN STREET $\overset{H}{T}$ ROME, N.Y. $\overset{H}{T}$ 12415 ■

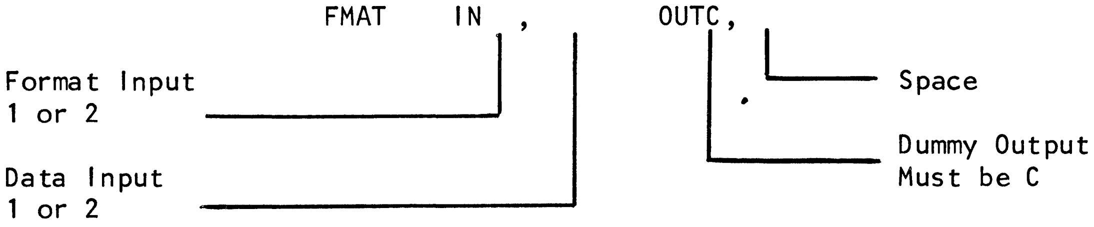
VERIFY

VFY LOC\ CON\ LOC ■

If the data keyed in the current field, Element 1, differs from the original data read in from cassette, program control jumps the number of instructions specified by Element 2. Unverified data is stored in the location specified by Element 3. Element 3 is optional.

NOTES:

- Data record lengths are limited to 256 characters or less.
- Printer control codes (\ and ¬) and printer control sequences (FF, LF and ESC followed by HT, VT, A-F) which are constant data fields must be included in both the data collection and verify formats.
- The job selection sequence must specify a "Dummy" primary output as follows:



- So long as the data typed in a field being verified matches the data input from cassettes the verify operation does not differ from normal data entry. However, if a miss-match is found a VFY error will be displayed when the operator completes the field. When verify errors occur, the cursor is left under the first incorrect character in the field. The operator must then check the source document carefully and either change the letter on the screen or type the same letter again to correct the previously keyed data. Each error in the field will be flagged individually. In cases where the entire field is wrong, the operator can use the clear field key to clear the data and permit re-entry of the complete field. The field will then automatically clear again so that the new entry can be immediately verified. When errors are found, the data is corrected by an automatic insertion on the data tape.

EXAMPLES:

Verify the name, customer number and ship via fields of the following record.

Data Collection Format

```
[NAME             A]                          ¬|
[CUSTOMER NO.     R]      ¬¬|

[C]FRAGILE\|

                    E
[SHIP VIA M]     [*]SA ■
                    C
```

Data Verification Format

```
[NAME             W]                           ¬|
[CUSTOMER NO.     X]      ¬¬|

[1M]         \|

                    E
[SHIP VIA  W]    [*]SA ■
                    C
```

Field programs to be used with the verification format

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
NAM	VFY	001	001		F	\	1	■
	GTO	002	002		1	■		
CUS	VFY	001						
	GTO	002						
SHI	VFY	001						
	GTO	002						

NOTE:

- "FRAGILE" will be brought into a secondary input field (1M) without operator action and will not have to be verified.

In a case where the original data collection format was used with an application program, some reprocessing may have to be done during data verification. The following example illustrates this point.

Data Collection Format with Processing

```
[SITE NO.      C]3791

[ITEM NO.      R]        |
[QUANTITY      X]        |
[PRICE         X]           |
[AMOUNT        N]              ■

[TOTAL     X]              [*] ■
```

Data Verification Format with Processing

```
[6M] [SITE NO.    1M]      I

[ITEM NO.         X]       I
[QUANTITY         X]       I
[PRICE            X]          I
[AMOUNT           N]             [7M]           ■

[TOTAL       X]             [*] ■
```

Field Programs to be used with Verification Format

FIELD PROGRAMS										1		
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	0	1	2
ITE	RTJ	009	009		F	\		■				
	VFY	008	008		F	\	1	■				
	GTO	002	002		1	■						
QUA	VFY	008										
	MVE	001	001		F	\	R	0	■			
	GTO	002										
PRI	VFY	008										
	MPY	003	003		F	\	R	0	\	R	0	■
	EXF	002										
	MVE	004	004		R	0	\	F	■			
	ADD	005	005		F	\	R	1	\	R	1	■
	GTO	002										
TOT	MVE	006	006		R	1	\	F	■			
	MVE	007	007		0	\	R	1	■			
	VFY	008										
	GTO	002										

NOTES:

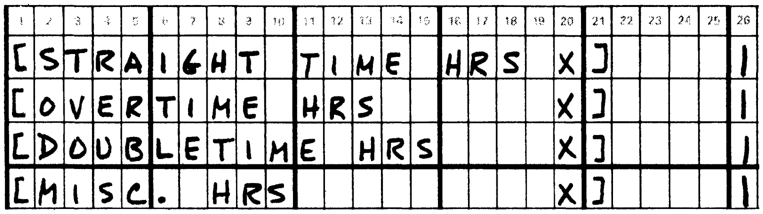
- Since the data being verified contains an item record (a record that may be repeated an indefinite number of times) the verification format must begin with a [6M] I field and, like the format, must not contain a [*]. The [6M] I field provides a pause for the operator to use the next format key after the last item has been verified.

BEL Any defined table can be used.

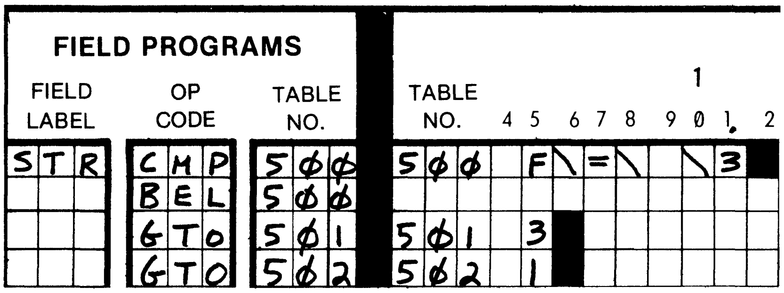
The alarm is sounded once. Program execution continues uninterrupted.

EXAMPLE:

If the Straight Time Hrs. field is blank move three fields, into the Misc. Hrs. column.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
[	S	T	R	A	I	G	H	T		T	I	M	E		H	R	S		X	]					1
[	O	V	E	R	T	I	M	E		H	R	S							X	]					1
[	D	O	U	B	L	E	T	I	M	E		H	R	S					X	]					1
[	M	I	S	C	.		H	R	S										X	]					1



FIELD PROGRAMS																			1		
FIELD LABEL			OP CODE			TABLE NO.				TABLE NO.			4	5	6	7	8	9	0	1	2
S	T	R	C	M	P	5	0	0	■	5	0	0		F	\	=	\		\	3	■
			B	E	L	5	0	0	■												
			G	T	O	5	0	1	■	5	0	1		3	■						
			G	T	O	5	0	2	■	5	0	2		1	■						

Additional Notes

ERROR

ERR CON ■

Element 1 contains a three character error message which appears on the status line when the ERR instruction is executed. The alarm sounds and the keyboard locks except for the ERROR RESET key. After the operator depresses the ERROR RESET key, the error message disappears, the cursor returns to the beginning of the current data field, and the field program may be reexecuted.

NOTES:

- The depression of ERROR RESET after the execution of the ERR command does not clear the field that caused the error condition.

EXAMPLE:

Insuring that the operator enters at least eight hours in a field.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.									1					
												4	5	6	7	8	9	0	1	2	3	4	5
H	O	U	C	M	P	0	1	0	0	1	0		F	\	>	=	\	8	0	0	\	2	■
			G	T	O	0	2	0	0	2	0		1	■									
			E	R	R	0	3	0	0	3	0		H	R	S	■							

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	H	O	U	R	S		X	]	7	0	0			1

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[	H	O	U	R	S		X	]	7	0	0			1

NOTE:

- After depression of the ERROR RESET key the cursor returns to the beginning of the hours field. The seven will remain on the screen until new data is keyed over it.

Additional Notes

EXECUTE FIELD

EXF CON ■

The field pointer moves forward or backward for the number of fields specified by Element 1. Additional instructions in the field program may refer to a different field, after execution of EXF.

NOTES:

- To move the field pointer backwards use the minus symbol after the number, i.e. 2-.
- The programmer has the responsibility to stay within the fields of the current format.
- The EXF instruction may move the field pointer, but execution of the current field program continues.
- The EXF instruction considers constant fields the same as any other fields.

EXAMPLES:

After the freight charge is entered, the sales tax and invoice total is displayed in the two subsequent fields. Register 1 contains the invoice total.

FIELD PROGRAMS																		1			
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3
F	R	E	M	V	E	7	0	1	7	0	1		F	\	R	0	■				
			E	X	F	7	0	2	7	0	2		1	■							
			M	P	Y	7	0	3	7	0	3		4	\	R	1	\	F	\	2	■
			A	D	D	7	0	4	7	0	4		F	\	R	1	\	R	1	■	
			E	X	F	7	0	2													
			A	D	D	7	0	6	7	0	6		R	0	\	R	1	\	F	■	
			G	T	O	7	0	2													

Before

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	F	R	E	I	G	H	T				X	]					1		
[	S	A	L	E	S		T	A	X		N	]					1		
[	I	N	V		T	O	T				N	]							■

After

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[	F	R	E	I	G	H	T				X	]	2	5	0	0	1		
[	S	A	L	E	S		T	A	X		N	]	4	0	0	0	1		
[	I	N	V		T	O	T				N	]	1	0	6	5	0	0	■

R1	1	0	0	0	0	0					

R1	1	0	6	5	0	0					

To move program control from the TAX CODE field back to the TAX field to display the amount of tax, the EXF instruction would be coded as follows:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
[	D	A	T	E		X	]	0	9	3	0	7	4	1			
[	T	A	X			N	]							1			
[	I	N	V		N	O		W	]	A	1	2	5	4	3	2	1
[	T	E	R	R		N	O		N	]	4	5	1				
[	T	A	X		C	O	D	E		X	]	4		1			

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7
	EXF	980	980		3	—	

Additional Notes

GO TO

GTO   CON ■

Element 1 specifies the number of fields, forward or backward, to move the cursor. The execution of the current field program stops and control returns to the operator.

NOTES:

- To move the cursor backward, use the minus symbol <u>after</u> the number i.e. 2-.
- The programmer has the responsibility to stay within the fields of the current format.
- GTO 1 in the last field of a format causes output to the I/O Buffer.
- The GTO instruction counts constant fields the same as any other field.

EXAMPLES:

Display the dollar amount in its own field and advance the cursor to the tax rate field.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
[	D	O	L	L	A	R	S		X	]										[	T	A	X		R	A	T	E		X	]			1

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9
D	O	L	M	V	E	1	1	0	1	1	0		R	I	\	F	■
			G	T	O	1	1	1	1	1	1		1	■			

If the tax rate entered is zero skip the tax amount field.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
[	G	R	O	S	S							X	]									1
[	T	A	X		R	A	T	E				X	]				1					
[	T	A	X		A	M	O	U	N	T		N	]					1				
[	N	E	T									X	]									1

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4
G	R	O	M	V	E	1	2	0	1	2	0		R	1	\	F	■					
			G	T	O	1	2	2	1	2	2		1	■								
T	A	X	C	M	P	1	2	4	1	2	4		F	\	/	=	\	0	\	6	■	
			M	P	Y	1	2	6	1	2	6		F	\	R	1	\	R	2	\	2	■
			E	X	F	1	2	2														
			M	V	E	1	2	8	1	2	8		R	2	\	F	■					
			A	D	D	1	3	0	1	3	0		F	\	R	1	\	R	1	■		
			G	T	O	1	2	2														
			G	T	O	1	3	2	1	3	2		2	■								
N	E	T	M	V	E	1	3	4	1	3	4		R	1	\	F	■					

NEXT FORMAT/UNCONDITIONAL

NFT Any defined table can be used.

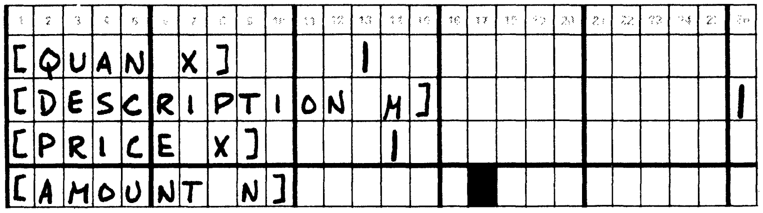
Replace the current format on the CRT with the next format from the format input device. The field program is terminated.

NOTES:

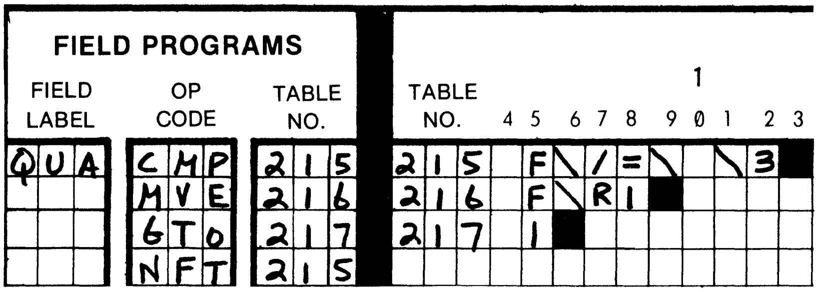
- The NFT advances the format input device 1 logical record.
- Since the NFT instruction terminates the field program and replaces the format on the CRT process and output any required data before executing the NFT instruction.

EXAMPLE:

Enter a blank in the quantity column after all the line items for the invoice have been entered. This will cause the trailer format to be automatically advanced onto the screen.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
[	Q	U	A	N		X	]					1													
[	D	E	S	C	R	I	P	T	I	O	N		M	]											1
[	P	R	I	C	E		X	]					1												
[	A	M	O	U	N	T		N	]							■									



FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3
Q	U	A	C	M	P	2	1	5	2	1	5		F	\	/	=	\		\	3	■
			M	V	E	2	1	6	2	1	6		F	\	R	1	■				
			G	T	O	2	1	7	2	1	7		1	■							
			N	F	T	2	1	5													

Additional Notes

NEXT FORMAT/CRT

N X F

NXF CON ■

Advances the format device, forward or backward, by the number of physical records specified in Element 1. The next format is then read onto the CRT, replacing the previous format. The field program is terminated.

NOTES:

- The NXF instruction advances physical records from the device specified by the format input in the status line.
- The minus sign is placed after the number to indicate backward movement (i.e. 2-).
- If the format input is a C, representing magnetic tape or card reader, Element 1 can not be negative.
- If Element 1 is zero, the format input is neither advanced nor backspaced, before the next format is read onto the CRT.
- Since the NXF instruction terminates the field program and replaces the format on the CRT, process and output any required data, before executing the NXF instruction.

EXAMPLES:

If the operator enters "YES", read a new format onto the CRT.

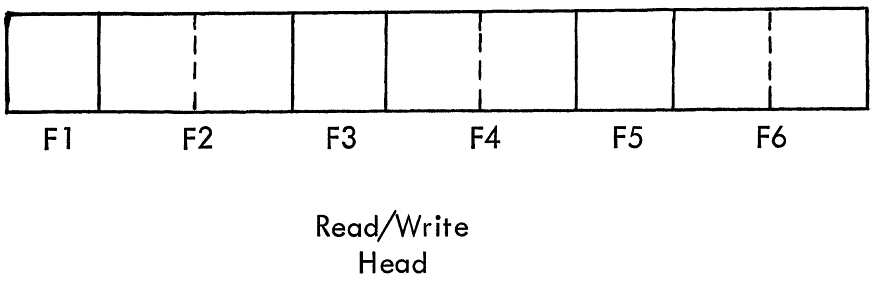
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	M	O	R	E		I	N	V	O	I	C	E	S		6	W	]				1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2
MOR	ETC	010	010		F	\	2	\	Y	E	S	■
	NXF	020	020		2	■						
	GTO	030	030		1	■						

Referring to the diagram below and assuming that the third format, F3, is on the CRT, the following is true.

NXF	Format that will be read onto the CRT
0	F4
2	F5
3	F6
1-	F3 (reread the record)
4-	F1



NOTE:

- A dotted line means that two physical records comprise one format.

NEXT FORMAT/BUFFER

NXT CON ■

Move the format device, forward or backward, the number of physical records specified in Element 1. The next format is then read onto the CRT, replacing the previous format. The field program is terminated.

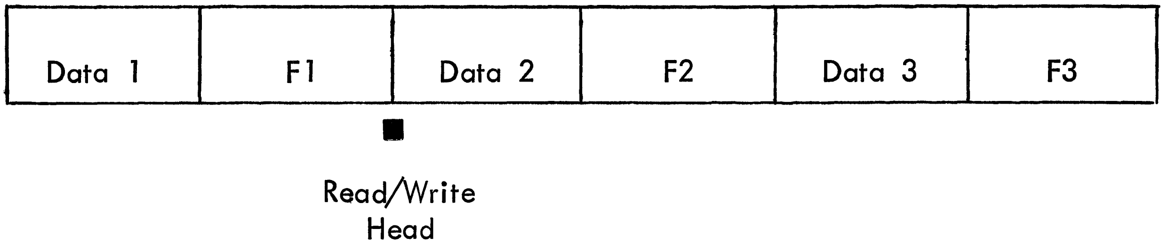
NOTES:

- NXT differs from NXF in that the skipped records are read into the I/O Buffer; therefore, the I/O Buffer contains the last record skipped when the NXT instruction terminates.
- The NXT moves physical records, 256 or fewer characters, not logical records.
- The format device is specified by the format input in the status line.
- The minus sign is placed after the number to indicate backward movement. i.e. 2-.
- If the format input is a C, representing magnetic tape or card reader, Element 1 can not be negative.
- If Element 1 is zero, the format device is neither advanced nor backspaced, before the next format is read onto the CRT.
- Since the NXT instruction terminates the field program and replaces the format on the CRT, process and output any required data before excuting the NXT instruction.

EXAMPLE:

Referring to the cassette tape diagram below and assuming that the records shown are under 256 characters in length the following would be true if the first two records have been read.

NXT	Record Read onto CRT	Data Contained in the I/O Buffer after Execution of the NXF Instruction
1	F2	Data 2
3	F3	Data 3
1-	F1 (Reread the Record)	Remains Unchanged



OVERLAY

O V L

Loads the overlay module from the primary input device into memory and brings the logical record following it onto the CRT as the next format. The cursor appears in the first data field of this format. The field program is terminated.

OVL Any defined table can be used.

NOTES:

- In multiple page applications a programmer can divide the program into overlay modules to make more memory available for each page of format.
- Since the OVL instruction terminates the field program and replaces the format on the CRT, process and output any required data before executing the OVL instruction.

EXAMPLES:

The extended memory field control character, W, and the I/O device control character, 7, are combined to allow the program to execute without operator interaction. No data is output since the I/O device control character prevents it.

1	2	3	4	5	6
[	7	W	]		

FIELD PROGRAMS						
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
7W]	OVL	100	100		1	

It is not necessary to have a separate field program to read in the overlay. After the data on the CRT has been output OVL can be specified in place of a GTO and it will read in the overlay and end the current field program.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1
	PUT	249	249		F	\	1	\	8	5	■
	WRT	250	250		2	■					
	OVL	100									

The format below is to be used only once before the next overlay is read in. To eliminate the procedure of putting the record to the I/O Buffer and then writing it out; an alternate procedure may be used. Switch 1 is set to zero when the format shown below is advanced onto the screen. It is changed to one after failing the test and processing continues on thru the rest of the format. The record is written out and control returns to the [7W] field when [7W] is activated this time, an overlay will be read in because Switch 1 is set to 1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
[	7	W	]		1																
[	A	C	C	O	U	N	T		N	O		X	]							1	
[	P	R	O	D	U	C	T		C	O	D	E		M	]						1
[	A	M	O	U	N	T							X	]						■	

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0
7W]	TSW	100	100		1	\	1	\	3	■
	SSW	101	101		1	\	0	■		
	OVL	100								
	SSW	100								
	GTO	100								

REGISTERS AND SWITCHES

REG NONE

The REG instruction allocates memory for Registers 0 thru 9 and the switch byte.

NOTES:

- Never include REG within Field Program.
- REG should be circled at the bottom of the TAL Program form along with the other OP CODES used and enter with them during program generation.
- REG requires no table because it is never entered within a field program.

Additional Notes

READ

RDX CON ■

Element 1 specifies the input device from which a physical record will be read into the I/O Buffer.

NOTES:

- Element 1 can be a 1, 2 or C for Cassette 1, Cassette 2 or Magnetic Tape, respectively.
- A physical record from cassette 1 that contains 300 characters and store it in memory.
- When an EOF is encountered, the field program is terminated and an EOF error message is displayed.

EXAMPLE:

Read the next logical record from cassette one and store the entire record in memory. The record is 300 characters in length.

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4	5	6	7
			R	D	X	1	1	1	1	1	1		1	■											
			M	V	M	1	1	2	1	1	2		2	5	6	\	3	\	1	\	2	0	\	1	■
			R	D	X	1	1	1																	
			M	V	M	1	1	3	1	1	3		2	5	6	\	3	\	1	\	2	1	\	1	■

NOTE:

- RDX 111 will read the first 256 characters and RDX 113 will read the last 44.

Additional Notes

WRITE

WRT CON ■

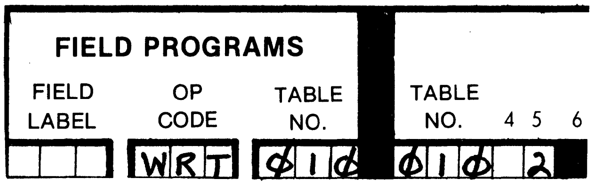
Write the contents of the I/O Buffer onto the output device specified in Element 1.

NOTES:

- Element 1 can be a 1, 2, C or P for Cassette 1, Cassette 2, Magnetic Tape or Printer, respectively.
- The WRT instruction has no affect upon any data displayed on the CRT or the execution of any subsequent steps of the program.
- Element 1 may differ from the output device selected in the FORMAT output device in the status line.
- Caution should be used when Element 1 matches the output device selected in the FORMAT output device. The data could be outputted twice; once via the WRT instruction and once when the format is completed.
- Data is transferred from Buffer to I/O device until a RS or the end of the Buffer is encountered. If encountered anywhere in the Buffer an error will occur.

EXAMPLES:

Write the contents of the I/O Buffer onto cassette 2.



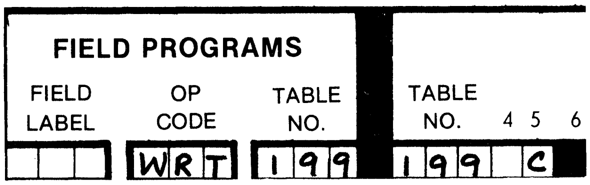
FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
	WRT	010	010		2	

NOTE:

- Cassette 2 need not be specified as either primary or secondary output in the status line to execute the above instruction.

Write the contents of the I/O Buffer onto magnetic tape.



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
	WRT	199	199		C	

INSERT

INS CON ■

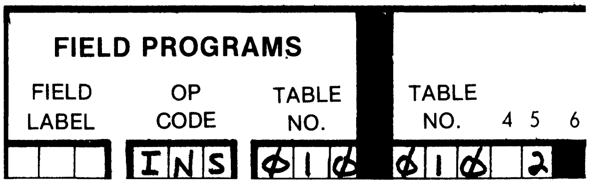
The cassette specified by Element 1 is backspaced one physical record; and the contents of the I/O Buffer is written onto it.

NOTES:

- Element 1 can be a 1 or 2.
- The record in the I/O Buffer must be the same length as the original record on the tape.

EXAMPLE:

Insert the contents of the I/O Buffer onto cassette 2.



FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
	INS	010	010		2	

Additional Notes

SEARCH

S R H

SRH CON \ CON ■

Search the device specified in Element 1 for the identifier in the current data field. Each physical record is read into the I/O Buffer, until a match is found. If no match is found, a jump forward or backward is taken according to Element 2.

NOTES:

- Element 1 can be a 1, 2, or C.
- The SRH instruction is not position oriented. The complete identifier can be present in any part of the physical record.
- Leading and trailing blanks in the identifier are significant.
- The search instruction can only operate on physical records.
- During the search a record separator temporarily appears on the screen after the last character in the identifier.
- The search instruction can search on compound identifiers, as in Search Mode.
- The jump in Element 2 cannot jump over other field programs; but must jump thru them.
- To jump two instructions back use 2-.

EXAMPLES:

Search the magnetic tape for the record containing "712".

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
[	I	T	E	M		#		L	]	7	1	2	■			1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
ITE	SRH	010	010		C	\	2	■
	GTO	020	020		1	■		
	REW	010						
	ERR	030	030		I	T	E	■

NOTE:

- The operator must depress the TAB/SKIP key to initiate the search.
- The RS shown in the format above is not a permanent part of the search identifier; but is temporarily displayed in the field as a function of SRH.

Search cassette 1 for the compound identifier "JONES:ELM:FORD".

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
[	N	A	M	E				W	]	J	O	N	E	S	:	E	L	M	:	F	O	R	D	■					1

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
NAM	SRH	010	010		1	\	2	■
	GTO	010						
	REW	010						
	ERR	020	020		N	A	M	■

NOTE:

- The compound identifier "JONES:ELM:FORD" must appear in the same order on the physical record.

REWIND

R E W

REW CON ■

Element 1 specifies the I/O device to be rewound.

NOTES:

- Element 1 can be a 1, 2, or C for Cassette 1, Cassette 2, or Magnetic Tape, respectively.
- If the tape device specified in Element 1 is already rewound, the execution of a REW instruction will cause an End of Tape, "EOT", to occur.

EXAMPLES:

Rewind cassette 2.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
	REW	701	701		2	

Rewind the magnetic tape.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6
	REW	966	966		C	

Additional Notes

BACKSPACE

BSP

BSP CON \ CON,IR ■

Element 1 is backspaced the number of physical records specified by Element 2.

NOTES:

- Element 1 can be a 1, 2 or C for Cassette 1, Cassette 2 and Magnetic Tape, respectively.
- The maximum number of physical records you may backspace with one instruction is 256.

EXAMPLES:

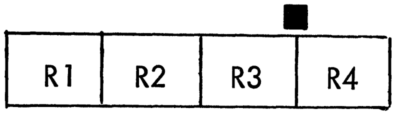
Backspace cassette 1 two physical records.

FIELD PROGRAMS

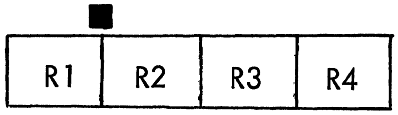
FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
	BSP	500	500		1	\	2	

NOTE:

- Position of Read/Write head before.



- Position of Read/Write head after.



Backspace cassette 2 the number of records contained in the index register.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9
	BSP	616	616		2	\	I	R	

IR 00000100

NOTE:

- Since the index register contains the value four, the cassette will be backspaced four physical records.

Backspace the magnetic tape four physical records.

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
	BSP	747	747		C	\	4	

TAL PROGRAMMING TECHNIQUES

Various programming techniques have been developed through experience which enables the programmer to conserve memory, and/or to accomplish certain required results not readily apparent to the new programmer. Each instruction added to a program requires the loading of subroutines which provide for the execution of that particular instruction. As a general rule, when the memory requirements of the program are known to be approaching the memory limits of the terminal in which the program is to be executed, the number of OP codes used should be limited as much as possible. OP codes already used in a program can sometimes be used to accomplish the same results without having to introduce new ones into the program. Following are samples of some of these techniques:

Eliminating Compare Instruction

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
NUM	CMP	0A0	0A0		F	\	<	\	R	0	\	2	■
	GTO	0B0	0B0		1	■							
	ERR	0C0	0C0		S	E	Q	■					

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
NUM	SUB	0A0	0A0		F	\	R	0	\	R	1	■	
	TSC	0B0	0B0		R	1	\	2	\	F	\	—	■
	ERR	0C0	0C0		S	E	Q	■					
	GTO	0D0	0D0		1	■							

Eliminating Duplicate Instruction

FIELD PROGRAMS

																		1					
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3	4	5
F	I	E	D	U	P	0	0	A	0	0	A												■
			G	T	O	0	0	B	0	0	B		1	■									

FIELD PROGRAMS

																		1			
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3
F	I	E	T	S	C	1	0	0	1	0	0		F	\	3	\	T	\		■	
			G	E	T	1	0	2	1	0	2		F	\	2	0	0	\	1	0	■
			G	T	O	1	0	3	1	0	3		1	■							
			P	U	T	1	0	4	1	0	4		F	\	2	0	0	\	1	0	■
			G	T	O	1	0	3													

Eliminating Move Instruction for Mixed Data

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9
			M	V	E	0	1	0	0	1	0		F	\	R	0	■

FIELD PROGRAMS

																		1				
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	0	1	2	3	4
			P	U	T	0	1	0	0	1	0		F	\	2	0	0	\	1	0	■	
			G	E	T	0	2	0	0	2	0		R	0	\	2	0	0	\	1	0	■

Eliminating the Move Instruction for Numeric Data

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9
	MVE	030	030		F	\	R	0	■

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1
	ADD	030	030		F	\	0	\	R	0	■

Eliminating Shift Instruction

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0
	SHF	001	001		F	\	L	\	1	■

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3
	PUT	001	001		F	\	2	0	0	\	1	0	■
	PUT	002	002			\	2	1	1	\	1	■	
	GET	003	003		F	\	2	0	1	\	1	0	■

Replacing Right Justify with Edit

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8
	RTJ	00A	00A		F	\		■

FIELD PROGRAMS

FIELD LABEL	OP CODE	TABLE NO.	TABLE NO.	4	5	6	7	8	9	1 0	1	2	3	4
	EDT	00A	00A		F	\	9	9	9	9	9	\	F	■

Replacing Right Justify and Move with Minus over Punch

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9
			M	V	E	0	0	B	0	0	B		R	0	\	F	■
			R	T	J	0	0	C	0	0	C		F	\	0	■	

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1
			M	O	P	0	0	B	0	0	B		R	0	\	F	\	0	■

Editing a Field with a Floating Dollar Sign and Left Fill with Asterisks

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
[	P	R	I	C	E		X	]	1	7	2	4	1	2	1							
[	P	R	I	C	E		X	]	*	*	*	$	1	7	,	2	4	1	.	2	1	■

FIELD PROGRAMS

FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2
P	R	I	E	D	T	0	V	0	0	V	0		F	\	$	9	,	9	9	9	,	9	9	9	.	9	9	\	F	■
			E	D	T	0	W	0	0	W	0		F	\	*	9	9	9	9	9	9	9	9	9	9	9	9	\	F	■
			G	T	O	0	X	0	0	X	0		1	■																

Placing a File Separator in the I/O Buffer

FIELD PROGRAMS																			
FIELD LABEL			OP CODE			TABLE NO.			TABLE NO.			4	5	6	7	8	9	1 0	1
			M	V	E	1	0	0	1	0	0		<	\	S	W	■		
			S	S	W	1	0	1	1	0	1		6	\	0	■			
			P	U	T	1	0	2	1	0	2		S	W	\	1	\	1	■
			P	U	T	1	0	3	1	0	3		R	S	\	2	\	1	■

NOTE:

- A File Separator consists of the transparent octal ASCII character 034 and a Record Separator. Since it is not possible to generate a table which contains a File Separator, we can simulate the operation by setting the switch byte to the ASCII byte configuration for a File Separator (bit sequence 00011100). The closest valid character configuration is the character "<" (bit sequence 00111100). By changing switch 6 to 0, we create the File Separator. This is put in the I/O Buffer in position 1, followed by a Record Separator in position 2.

Additional Notes

MEMORY REQUIREMENTS

The number of bytes that a table requires equals the number of characters in the table, including the record separator, less the sum of the number of elements in the table plus three.

Table Bytes required = R - (n + 3)

R = number of characters in the table.

n = number of table elements.

The number of bytes that a field program requires is the number of instructions plus one multiplied by four.

Program Bytes required = 4 (n + 1)

n = number of instructions.

The number of bytes required by each OP CODE varies. The table on the next page contains the number of bytes required for each OP CODE and the subroutines the OP CODE calls. The OP CODE bytes are counted only one time, though the OP CODE itself may be used several times in one field program or several field programs. Subroutine bytes are counted only once also. Subroutine one is called by almost every OP CODE; but the 36 bytes need to be counted only once.

The number of bytes required by each OP CODE varies. The table on the next page contains the number of bytes required for each OP CODE and the subroutines the OP CODE calls. The OP CODE bytes are counted only one time, though the OP CODE itself may be used several times in one field program or several field programs. Subroutine bytes are counted only once also. Subroutine one is called by almost every OP CODE; but the 36 bytes need to be counted only once.

OP CODE BYTE REQUIREMENTS

OP CODE	BYTES	SUBROUTINES CALLED
ACP	126	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 19, 24, 35, 36, 37
ADD	8	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 19, 20, 21, 22, 24, 27,
BEL	4	
BSP	72	1, 2, 3, 5, 8, 9, 15, 17, 24, 29, 30, 37
CK0	105	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK1	102	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK2	105	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK3	96	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK4	105	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK5	90	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK6	99	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 19, 20, 24
CK7	105	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK8	105	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CK9	119	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CKA	133	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 24
CMP	151	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 18, 19, 20, 24, 25, 26, 33, 34
CVB	72	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 18, 19, 22
CVD	99	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 19, 24, 27
DIV	472	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 19, 20, 22, 24, 27
DUP	92	1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 19, 23, 24
EDT	188	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 19, 20, 24
ERR	25	1, 2, 3, 5, 8, 9, 10, 19, 24
ETC	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 18, 19, 20, 24, 25
EXF	10	1, 2, 3, 5, 8, 9, 10, 15, 17, 19, 24
GET	12	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 33, 34
GTI	36	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 34
GTO	40	0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 15, 17, 19, 24
INS	12	1, 2, 3, 8, 9, 17, 24, 29, 30, 37
IXR	34	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 34
JMP	6	1, 2, 3, 5, 8, 9, 10, 18, 19, 24
LMA	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 34
LOD	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 32, 34
MOP	87	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 19, 20, 24
MPY	247	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 19, 20, 22, 24, 27
MVE	61	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 19, 24

OP CODE BYTE REQUIREMENTS

OP CODE	BYTES	SUBROUTINES CALLED
MVM	80	1, 2, 3, 5, 8, 9, 10, 12, 15, 17, 19, 24, 34
NFT	22	1, 37
NXF	106	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 17, 19, 24, 28, 29, 30, 37
NXT	106	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 17, 19, 24, 28, 29, 30, 37
OVL	153	0, 1, 6, 7, 8, 9
PTI	127	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 34
PUT	41	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 33, 34
RDX	10	1, 2, 3, 8, 9, 17, 24, 28, 29, 37
REG*	111	
REW	40	1, 2, 3, 8, 9, 17, 24, 29, 30, 37
RNG	50	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 17, 18, 19, 20, 24, 25, 26
RTJ	61	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 17, 18, 24
SCN	80	1, 2, 3, 5, 8, 9, 10, 18, 19, 24
SHF	90	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 19, 20
SIR	13	1, 2, 3, 5, 8, 9, 15, 19, 24
SMA	57	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 35
SRH	122	1, 2, 3, 5, 6, 7, 8, 9, 10, 17, 18, 19, 24, 28, 29, 37
SSW	39	2, 8, 9, 16, 17
STR	14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 19, 24, 33, 35
SUB	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 19, 20, 21, 22, 24, 26
TCP	33	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 19, 20, 24
TLU	52	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 18, 19, 20, 23, 24, 25
TSC	87	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 18, 19, 20, 24, 25
TSW	43	1, 2, 3, 5, 8, 9, 10, 16, 18, 19
VFY	614	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 19, 24
WRT	10	1, 2, 3, 8, 9, 17, 24, 29, 31, 37

* REG must be specified when using switches or registers.

SUBROUTINE BYTE REQUIREMENTS

SUBROUTINE NUMBER	BYTES
00	84
01	36
02	26
03	26
04	176
05	39
06	10
07	10
08	13
09	13
10	13
11	28
12	26
13	10
14	50
15	69
16	8
17	74
18	105
19	17
20	28
21	154
22	80
23	40
24	55
25	10
26	141
27	47
28	47
29	16
30	59
31	43
32	24
33	26
34	93
35	34
36	35
37	17

INTRODUCTION

The Software Library contains several peripheral drivers for magnetic tapes, punch cards and printers. These drivers are included in both library cassettes (TAL and COMM); and they may be combined with either TAL programs or package programs. The Generator must be used to link the drivers to other programs. Detailed operating instructions are provided in the program generation section of this manual. A description of each driver and a utility program follows.

PUNCH CARDS

Card Reader #CR

This hardware reads 80 column cards at a rate of 250 cards per minute. To use the card reader a C is selected as input in the job selection sequence. Each card constitutes an 80 character record with a record separator appended. Upon sensing an empty hopper in program mode a file separator is constructed in place of the last card read. Therefore, a blank card should be placed at the end of the card file.

In format and batch modes a "CRD" message is displayed when the hopper empty condition is sensed, the last card read is not output, and no file separator record is constructed.

In batch mode the settings of the PROG CTL and MEM CTL switches produce the following results:

	Program Control	Memory Control
ON	File separator constructed on "hopper empty" condition.	Three cards are blocked into one 240 character record.
OFF	"CRD" message is displayed "hopper empty" condition.	One card is read into one 80 character record.

In all modes the AUTO OPRT switch controls the termination of the card read operation.

ON - Two file separators are required to terminate operation (2 or 6 blank cards).

OFF - One file separator is required to terminate operation (1 or 3 blank cards).

Card Punch #CP

This hardware punches 80 column cards at a rate of 25 character per second. A card that contains a record with less than 80 characters is released anytime a record separator is output to the card punch. The card will not be released if the record separator is in column one such as an 80 character record. This is to avoid releasing blank cards with no punch in them. A card is released after 80 characters are output even if not followed by a record separator as in the case of larger than 80 character records.

PRINTER DRIVERS

The following drivers provide control for Models 3480, 3481, 3482, 3484, 3485 and 3486 printers. To use a software driver, Option Zero must be activated by the Sycor customer engineer. This is a permanent change and the hardware driver is no longer accessible when Option Zero is activated. These drivers can be used with a package program under program control; with TAL instructions under format mode; or alone under batch mode.

Use a P when selecting the printer as output during the job selection sequence.

The software drivers use the same vertical control characters recognized by the ROM driver plus six additional ones for #CT, #CO and #PT. The table on the following page lists these characters and their function.

PRINTER VERTICAL CONTROL CHARACTERS

Vertical Control Characters	Function
ESC A	Skips to Tab Channel A
ESC B	Skips to Tab Channel B
ESC C	Skips to Tab Channel C
ESC D	Skips to Tab Channel D
ESC E	Skips to Tab Channel E
ESC F	Skips to Tab Channel F
ESC G	Skips to Tab Channel G
ESC H	Skips to Tab Channel H
ESC I	Skips to Tab Channel I
ESC J	Skips to Tab Channel J
ESC K	Skips to Tab Channel K
ESC L	Skips to Tab Channel L
FF	Skips to TOP OF FORM. Line count is reset to value assigned to Channel A in the vertical format record.
VT	Skips to Channel B

All the drivers provide automatic page restore. This takes place when Channel 6 overflows. After the line defined as Channel L is printed, the paper is slewed to the top of form as defined in the vertical format tape.

A vertical format tape (VFT) must be installed on the printer that corresponds to the length of the form. Any standard punch tape will work. The length of the VFT is proportional to the printed form used. Each sprocket hole represents one line on the form (10 sprocket holes to the inch).

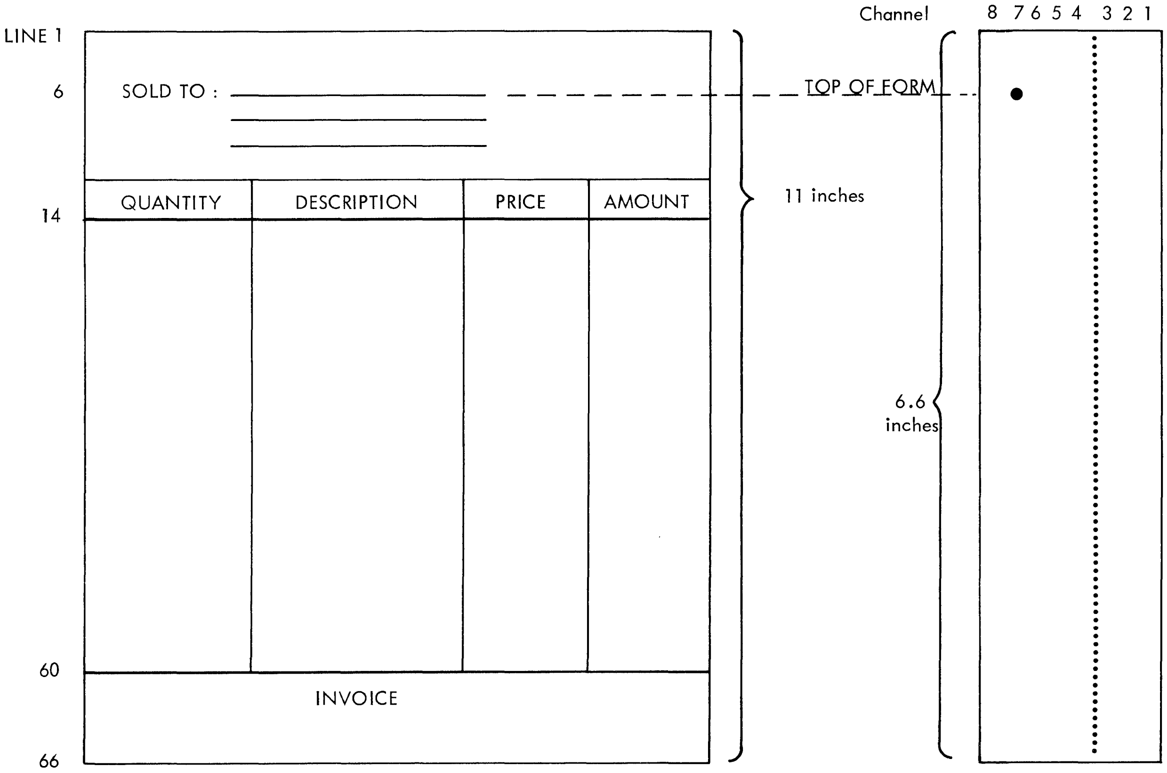
All printers use an eight channel (one inch) VFT except for Model 3480 which requires none. A five channel tape may be used instead of eight on Models 3484 and 3485. Following is a chart of the printer drivers and their VFT requirements.

Driver Name	Printer Model No.	Channels to be defined in Vertical Tab Record	Vertical Format Tape	
			Minimum Length	Channel of First Line Punch
#CT	3481, 82 3484, 85	12	6.6 4.8	7 1 thru 5
#CO	Same as #CT with the exception that the zero is not slashed when printed.			
#ST	3480	6	N/A	N/A
#PT	3486	12	5.6	1

* The 3484 and 3485 printers also require a channel 3 punch in every frame.

The diagram below shows a standard 11 inch continuous invoice form and the VFT required for a Model 3481 or 3482 printer. After the VFT has been installed depression of the TOP OF FORM key causes the VFT to be positioned at the channel A punch. The form is then manually lined up to the first print line, which is the "SOLD TO" in this case.

After "SOLD TO" has been keyed in, an ESC B should be specified in the format. This will cause the paper to move to line 14 in the body of the invoice. Any time an FF is specified in the format on the CRT the paper will move to the first print line of the next form. If, however, printing does occur on the 60th line, the Auto Page Restore will be activated and cause an FF to occur. The paper will move to the first print line on the next form, providing Channel L in the vertical tab record is set to 60.



Models 3481, 3482, 3484 and 3485 Serial Printers #CT

A carriage return character will be automatically placed at the end of the 132 character print record unless a carriage return is already in position 132 or 133.

Model 3480 Serial Printer #ST

A carriage return character will be automatically placed at the end of a 132 character print record unless the data contains carriage return codes within the 133 characters.

Model 3486 Line Printer #PT

A carriage return character will be automatically placed at the end of a 132 character print record unless a carriage return is already in position 132 or 133.

MAGNETIC TAPE

The Magnetic Tape drivers available on the 340 use half-inch magnetic tape encoded at either 556 or 800 bits per inch. The 340 input/output operations are overlapped; therefore, a magnetic tape may not be used as both input and output in the same job selection. A "C" is specified for either input or output in the job selection sequence. To manually control the magnetic tape (writing a tape mark, rewinding, etc.) refer to the 340 Operators Manual. The table below lists the available drivers.

Driver Name	Computer Manufacturer	No. of Tracks	Code	Parity	Notes
#M7	IBM	9	BCD	Even	
#M9	IBM	9	EBCDIC	Odd	
#MA	NCR	9	ASCII	Odd	
#MG	GE	7	BCD	Odd	
#MH	Honeywell	7	BCD	Odd	
#MN	NCR	7	BCD	Odd	
#MS	IBM	9	EBCDIC	Odd	1600 BPI
#MT	Honeywell	7	BCD	Odd	
#MX	IBM	9	EBCDIC	Odd	No FS written

MERGE AND PRINT MAP

The merge and print program takes data from two separate input sources, merges the data from both sources according to a fill character and outputs the final result to one device. The terminal brings the fixed data on the CRT. The variable data is read into the buffer. The presence of a fill character brings the data from the buffer to overlay the fill characters. The results are released to the output device.

The fixed data input source contains the constant data, such as headings and specifies by a fill character an area to be filled with variable data. When a file separator is read from the fixed input device it rewinds this device. The fixed data tape also determines the printed format by the presence of printer control characters (HT, CR, ESC and A-F). The fixed data tape can contain a special fill character, form feed (FF), to insure the merging remains synchronous.

The variable data input source must contain as many data characters as the number of fill characters on the fixed data source. However, to avoid using blanks to specify no data for a particular field or to fill the remainder of a filed, a single HT character may be used. The presence of a HT character in the variable data will fill one entire field or the remainder of one field with blanks. If the special fill character FF is detected on the fixed data then the next character from the variable data source must be a FF.

The following is an example of fixed and variable sources and the resulting print out.

FIXED DATA:

```
FC
FR  BALL POINT PENS        $.@@        QTY @@@

 C
 R  RULED PADS             $.@@        QTY @@@

 C
 R  STAPLES A-84           $.@@        QTY @@@■
```

VARIABLE DATA:

$\genfrac{}{}{0pt}{}{F}{F}2504475123\genfrac{}{}{0pt}{}{H}{T}\genfrac{}{}{0pt}{}{H}{T}$ ■

MERGED RESULTS:

```
BALL POINT PENS          $.25         QTY 044

RULED PADS               $.75         QTY 123

STAPLES A-84             $            QTY
```

The presence of two HT characters cause the last price and quantity fields to be blank. If the FF did not correspond the program will issue an error (SEQ).

The job selection for the merge and print program has the following options:

PROG IN A, OUT $\underline{B}$,$\underline{C}$ $\underline{D}$

A: fixed data input device	1 or 2
B: variable data input device	1,2, or C
C: output device	1,2,C, or P
D: fill character	Any keyboard character (Use a visible keyboard character not present elsewhere in the fixed data).

The following package programs and peripheral drivers are available in the Sycor Software Library:

Program Name	Available on Terminal*	Locations (Octal)	Memory Required Decimal Bytes	Transfer Location**
#AP	340B	Relocatable	217	RO
#AR	340B	Relocatable	653	RI
#CO	340	4;040-5;374	477	P
#CP	340	Relocatable	346	RO
#CR	340	Relocatable	164	RI
#CT	340	Relocatable	382	P
#M7	340A1/B	10;000-17;150	1896	RI, RO
#M9	340A1/B	10;000-17;233	1947	RI, RO
#MA	340A1/B	10;000-17;220	1936	RI, RO
#MG	340A1/B	10;000-17;150	1896	RI, RO
#MH	340A1/B	10;000-17;150	1896	RI, RO
#MN	340A1/B	10;000-17;236	1950	RI, RO
#MS	340A1/B	10;000-17;300	2092	RI, RO
#MT	340A1/B	10;000-17;150	1896	RI, RO
#MX	340A1/B	10;000-17;300	2092	RI, RO
#PT	340	4;376-5;332	335	P, RI
#ST	340	Relocatable	362	P
ANS	340A1/B	Relocatable	230	MC
B5K	340B	11;140-17;377	1696	None
B7K	340B	31;140-37;377	1696	None
CCS	340A1/B	Relocatable	507	MC, MS
CR1	340A	21;000-23;777	994	None
CRC	340	6;000-6;341	226	None
DTP	340B	Relocatable	1588	PM
EBC	340	7;000-7;377	256	None
ITS	340B	13;060-17;377	1232	RO, PM
MAP	340	Relocatable	279	PM
MRT	340B	Relocatable	4	None
MSA	340B	Relocatable	1403	PM, MS, MC
RJE	340A1/B	Relocatable	2515	MS, PM
SEC	340B	Relocatable	4	None
TCM	340	Relocatable	219	MC, PM
TCS	340	Relocatable	20	MC, PM
TID	340B	Relocatable	4	None
UBP	340	Relocatable	902	PM

* 340 Indicates that the program is available on all 340 models.

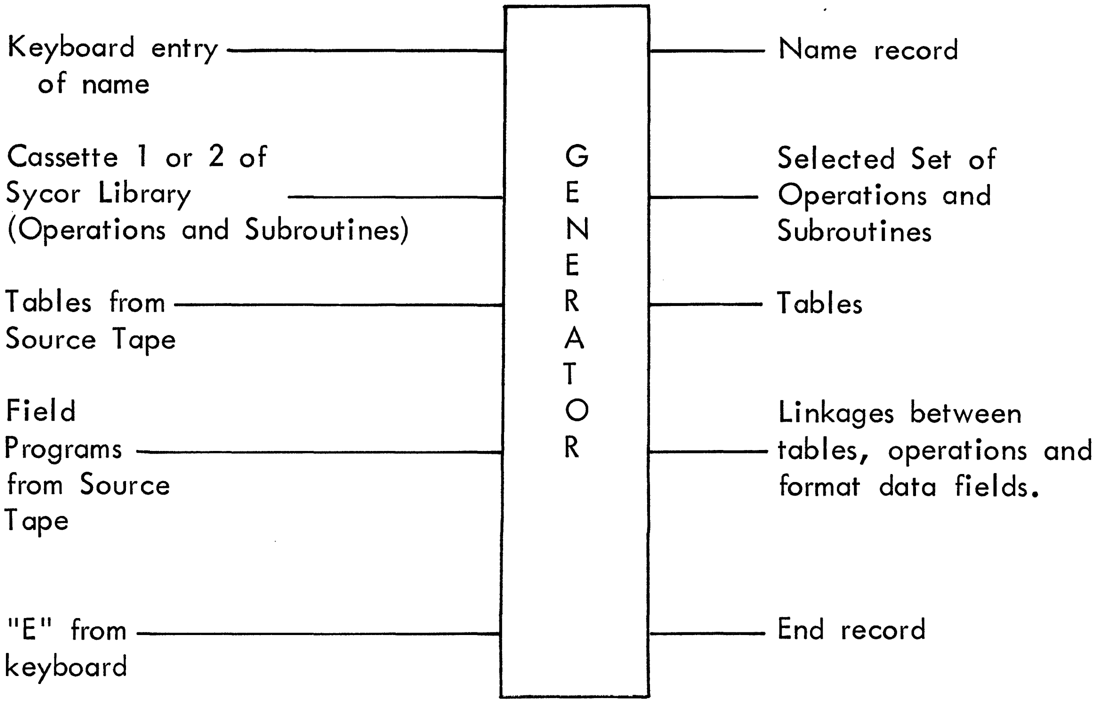
** Transfer Location: MC-Memory Control, MS-Memory Select, P-Printer, PM-Program Mode, RI-RAM Input, RO-RAM Output.

Additional Notes

PROGRAM GENERATION

TAL PROGRAM GENERATION

The Sycor Customer uses the Program Generator in a Model 340 terminal equipped with a minimum of 3K of extended memory to generate special purpose application programs. The program routines and Program Generator constitute the Software Library. The library is stored on two cassettes (TAL Library and COMM Library). Both cassettes are furnished when the Library is ordered; but either cassette can stand alone. Figure contains a list of the programs routines on each cassette. During generation application programs are recorded on a cassette tape to be loaded by an operator and used during formatted data collection. Below is shown how the generator acts on a keyboard entered name, the Sycor library, tables and field programs to generate an application program on cassette tape.



Program generation involves a series of well defined steps:

1. Preparation
2. Set Up
3. Required Operations
4. Tables
5. Field Programs
6. End
7. Load and Test

The steps are described in the subsections below.

Preparation

Figure 2 is a sample TAL Program which can be used as a guide during program preparation and generation. The program name has been filled out and the OP CODES have been circled. Since registers one thru four are used in the program REG has also been circled in the OP CODE section at the bottom of the TAL form.

The tables and field programs should be recorded ahead of time on a source tape and advanced onto the CRT during program generation. The source tape should be organized as shown in Figure 1 . A record separator is required at the end of each table and field program. A record separator is also used, by itself, to separate tables and field program groups. The source tape is prepared by typing the tables and field tables on the CRT and manually writing them out one by one in the order shown in Figure .

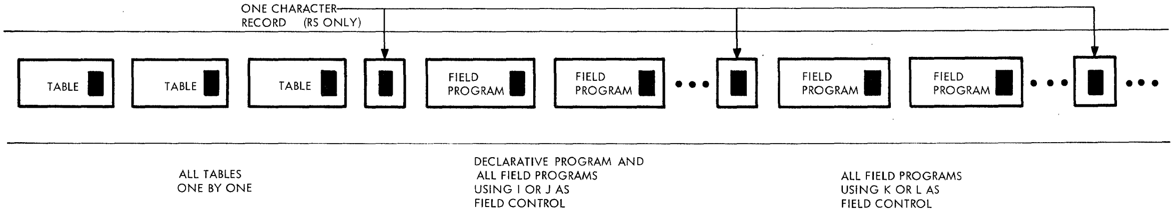
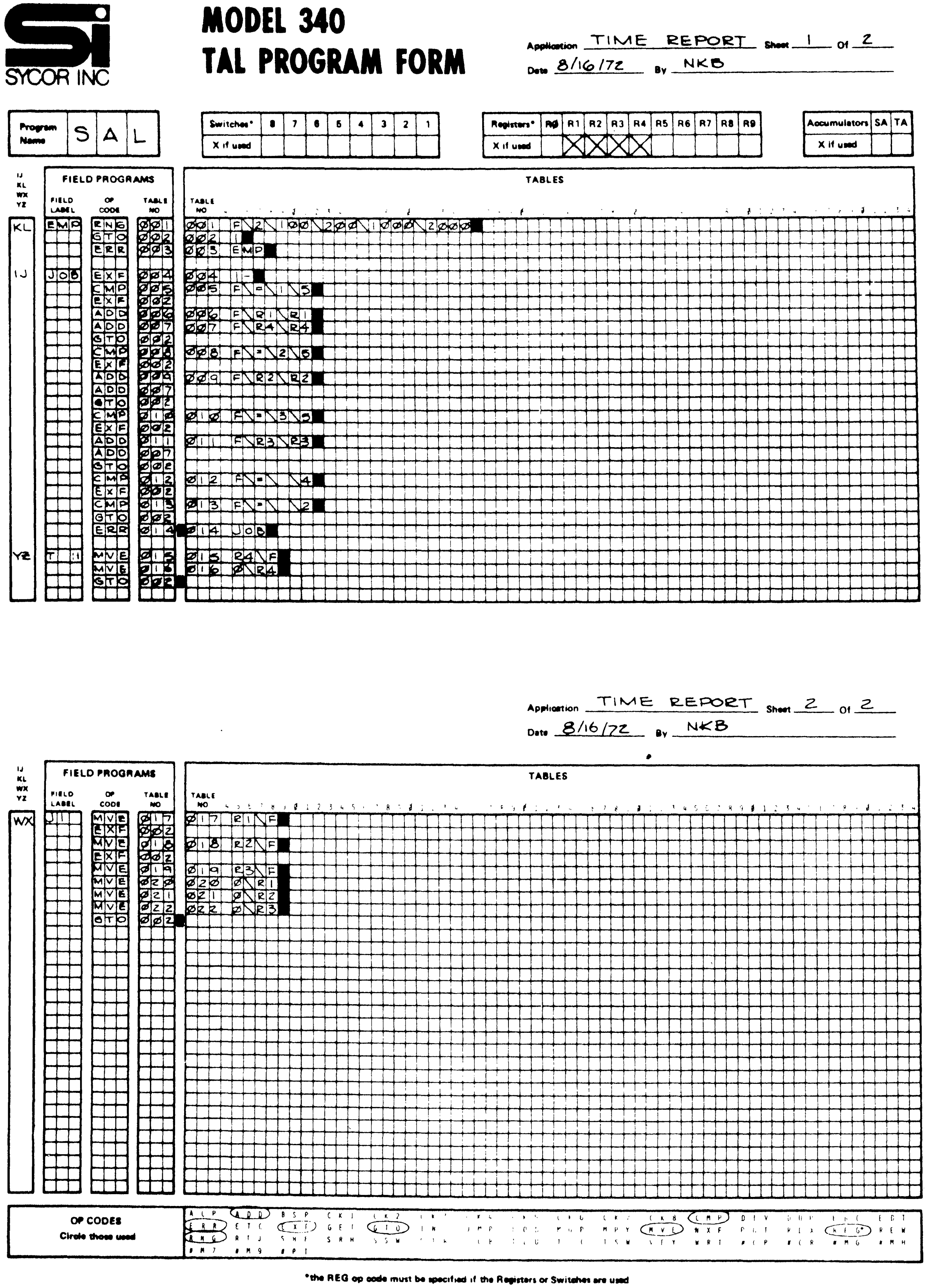


Figure 1 Source Tape Organization



SYCOR INC

MODEL 340
TAL PROGRAM FORM

Application TIME REPORT Sheet 1 of 2
Date 8/16/72 By NKB

Program Name: S A L

Switches*	8	7	6	5	4	3	2	1
X if used								

Registers*	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
X if used		X	X	X	X					

Accumulators	SA	TA
X if used		

FIELD PROGRAMS / TABLES

IJ KL WX YZ	FIELD LABEL	OP CODE	TABLE NO	TABLE NO	TABLES
KL	EMP	RNG	001	001	F\2\100\200\1000\2000
		GTO	002	002	1
		ERR	003	003	EMP
IJ	JOB	EXF	004	004	1-
		CMP	005	005	F\=\1\5
		EXF	002		
		ADD	006	006	F\R1\R1
		ADD	007	007	F\R4\R4
		GTO	002		
		CMP	008	008	F\=\2\5
		EXF	002		
		ADD	009	009	F\R2\R2
		ADD	007		
		GTO	002		
		CMP	010	010	F\=\3\5
		EXF	002		
		ADD	011	011	F\R3\R3
		ADD	007		
		GTO	002		
		CMP	012	012	F\=\\4
		EXF	002		
		CMP	013	013	F\=\\2
		GTO	002		
		ERR	014	014	JOB
YZ	T 1	MVE	015	015	R4\F
		MVE	016	016	0\R4
		GTO	002		

Application TIME REPORT Sheet 2 of 2
Date 8/16/72 By NKB

FIELD PROGRAMS / TABLES

IJ KL WX YZ	FIELD LABEL	OP CODE	TABLE NO	TABLE NO	TABLES
WX	J1	MVE	017	017	R1\F
		EXF	002		
		MVE	018	018	R2\F
		EXF	002		
		MVE	019	019	R3\F
		MVE	020	020	0\R1
		MVE	021	021	0\R2
		MVE	022	022	0\R3
		GTO	002		

OP CODES
Circle those used

ALP	[✓ ADD]	BSP	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	[✓ CMP]	DIV	DUP	ENC	EDT
[✓ ERR]	ETC	[✓ EXF]	GET	[✓ GTO]	IN	JMP	[illegible]	[illegible]	MPY	[✓ MVE]	NXF	PUT	RLX	[✓ REG*]	REW
[✓ RNG]	RTJ	SHF	SRH	SSW	[illegible]	[illegible]	[illegible]	[illegible]	TSW	VFY	WRT	#CP	#CR	#MG	#MH
#M7	#M9	#PT													

*the REG op code must be specified if the Registers or Switches are used

Figure 2 TAL Program Form

SOFTWARE LIBRARY
PROGRAM ROUTINES

Cassette 1 TAL				Cassette 2 Communications	Common to Both cassettes
ACP	CRC	LMA	RNG	B5K	#AP
ADD	CVB	LOD	RTJ	B7K	#AR
BEL	CVD	MAP	SCN	CCS	#CO
BSP	DIV	MOP	SHF	CR1	#CP
CK0	DUP	MPY	SIR	CRC	#CR
CKA	EBC	MVE	SMA	DTP	#CT
CK1	EDT	MVM	SRH	EBC	#M7
CK2	ERR	NFT	SSW	ITS	#M9
CK3	ETC	NXF	STR	MRT	#MA
CK4	EXF	NXT	SUB	MSA	#MG
CK5	GET	OVL	TCP	RJE	#MH
CK6	GTI	PTI	TLU	SEC	#MN
CK7	GTO	PUT	TSC	TCM	#MS
CK8	INS	RDX	TSW	TCS	#MT
CK9	IXR	REG	VFY	TID	#MX
CMP	JMP	REW	WRT	UBP	#PT
					#ST
					ANS

The steps in preparing a source tape are:

1. Load a blank tape on cassette one.
2. Turn program control off.
3. Depress job select, F, space four times and enter.
4. Type a table and a record separator (RS) on the CRT.

   Example:

   015_R4\ F ■

5. Write the table on tape 1 (tape 1, write).
6. Repeat step 4 and 5 until all of the tables have been recorded on the source tape (record each table only once).
7. Type a record separator (RS) in position one on the first line of the CRT.

   ■

8. Write the record separator on tape 1 (tape 1, write). This single record separator marks the end of the tables.
9. Organize the field programs into up to four groups according to the field control characters used in the format. I and J field programs will be one group, K and L another group, W and X another group and Y and Z another group.
10. Do steps 11 through 15 below for each of the letter pair groups you have used.
11. Type a field program and a record separator (RS) on the CRT. The field program should be typed from left to right, top to bottom, leaving at least one space after the three letter label and between OP CODES and table numbers. The record separator is always the last character in a field program.

Example:

EMP_RNG_001_GTO_002_ERR_003 ■

12. Write the field program on tape 1 (tape 1, write).

13. Repeat steps 11 and 12 until all of the field programs in the group have been recorded on tape 1.

14. Type a record separator (RS) in position one on the first line of the CRT.

    ■

15. Write the record separator on tape 1 (tape 1, write). This single record separator marks the end of the field programs for a letter pair.

16. Repeat steps 11 through 15 for each of the letter pairs you have used.

17. Optional: Write a file separator on tape 1 (tape 1, shift FS).

18. Rewind and remove tape 1 (tape 1, rewind) and mark it "Source Tape". This tape will be used as input during program generation.

When a source tape has been prepared, the systems analyst is ready to set up the generator and generate the actual application program. If errors occur during program generation, refer to the error recovery procedures at the end of this section.

Set Up

The set up for program generation consists of copying the Loader on the output cassette tape (so that it will be convenient for the operator to use) and using the Loader to load the Generator. The systems analyst then selects program mode and types a three character application program name. (This is the name the operator will use later to load the program). The systems analyst then depresses enter and three "pointers" are displayed in the status line for reference during program generation.

```
BUF  XX  XXX     SYM  YY  YYY     MEM  ZZ  ZZZ
```

Each pointer consists of two octal numbers that identify a location in memory by page (04 through 17) and byte (000 through 377). BUF and SYM refer to locations in the terminal being used to generate the program tape. Figure 3 shows how extended memory is utilized during program generation. The generator is stored in pages 03 through 10 (shaded area). The buffer (BUF) starts at the "top" of the remaining memory in page 10 and the symbol table (SYM) starts at the "bottom" of memory in page 17. The systems analyst must check to be sure that BUF and SYM do not overlap during program generation. MEM (Figure 3) refers to the amount of memory needed in the terminal that will use the application program.

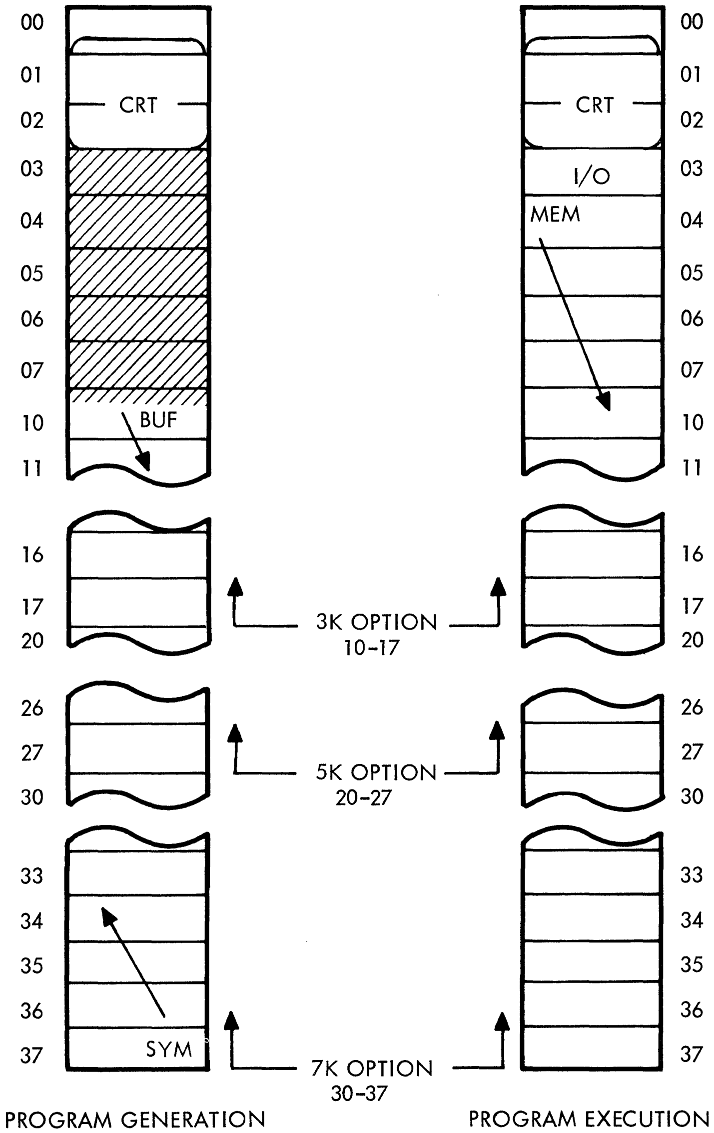


Figure 3

The steps for setting up program generation are:

1. Put the appropriate Sycor library cassette (TAL or COMM) on tape 1 and a blank cassette on tape 2.
2. Turn AUTO OPRT and MEM CTL off.
3. Depress JOB SELECT, F, SPACE, SPACE, SPACE, and ENTER.
4. Advance tape 1 (TAPE 1, ADV RCD).
5. Write the loader on tape 2 (TAPE 2, WRITE).
6. Load the loader (Depress SHIFT and LOAD simultaneously).
7. Depress JOB SELECT, P, 1, SPACE, SPACE, and ENTER.
8. Type GEN (the three character name for the generator) and depress ENTER.
9. The loader will search for the program name GEN.
10. When GEN is found it will be loaded by the loader.
11. Loading is complete.
12. Depress JOB SELECT, P, 1, 2, SPACE, and ENTER.
13. The status line will display:

```
PROG IN1  OUT2,  BUF  XX  XXX  SYM  YY  YYY  MEM  ZZ  ZZZ
```

14. Type the three character TAL program name (SAL for instance).
15. Depress ENTER, the name will be written out to the program tape as a name record.
16. The CRT now displays a list of the OP CODES available in the Sycor Software Library.

When set up has been completed the systems analyst is ready to select the required operations.

Required Operations

The status line will display "0" and a list of the names of all the operations in the library will be displayed on the CRT. Type in the names of the operations required, leaving one space between each. When all of the required operations are on the CRT put a record separator and depress the enter key. At this point the library tape is searched and as the operations are found they are relocated (addresses changed on CRT) and copied on cassette 2. A list of required subroutines is also being created by the program generator. When all the operation names have been erased from the CRT by the generator, the required subroutines will be loaded. When an EOF appears in the status line, the library tape may be rewound, removed and replaced by the source tape containing the tables and field programs.

The actual steps in selecting required operations are:

1. Type in the required operations, with one space (refer to the TAL Program Form). Don't forget that REG is required if the registers or switches are being used!

2. When only the names of the operations you need are left on the CRT, put a record separator following the instructions and depress enter.

3. As the operations are copied on cassette 2, their names will be erased from the CRT.

4. When all the names have been erased and an EOF appears in the status line, depress error reset.

5. Rewind and remove tape 1 (tape 1, rewind).

6. Leave tape 2 in place to collect further program output.

7. Depress enter to end the operation phase of program generation.

8. The cursor will be in the status line.

9. Make a note of the MEM address.

Continue with the next phase of program generation.

Tables

The systems analyst types T (for tables) in the status line. The tables can be entered automatically or manually, depending if PROG CTL is on or off.

To Enter The Tables.

1. Load the program source tape containing the tables and field programs on tape 1. (Refer to Figure for source tape format).
2. Type T for tables.
3. If PROG CTL is on, go to step 4. If PROG CTL is off, go to step 5. *
4. Depress ENTER. The tables will be converted to object form and written out on tape 2. Go to step 8.
5. Advance a table from tape 1 (TAPE 1, ADV RCD).
6. Depress ENTER. The table will be converted to object form and written out to tape 2.
7. Repeat steps 5 and 6 until all of the tables and a single record separator have been entered.
8. The cursor will be in the status line.
9. Make a note of the MEM address.

Continue with the next phase of program generation.

* If Any editing must be done to the tables turn PROG CTL off.

Field Programs

The eight extended memory field control letters are matched to form four pairs.

I and J

K and L

W and X

Y and Z

The field programs are similarly arranged into four groups and recorded on the source tape.

Each group of field programs is started by typing one of the field control letters for the froup in the status line (W, for instance, for W and X). The field programs for the group may be entered automatically or manually depending if PROG CTL is on or off. The systems analyst may then start another field program group, by typing another field control letter in the status line.

The steps for entering the field programs are:

1. Type I or J, K or L, W or X, Y or Z in the status line.
2. If PROG CTL is on, go to step 3. If PROG CTL is off go to step 4.
3. Depress ENTER. The field programs will be converted to object form and written on tape 2. Go to step 7.
4. Advance the first field program from the source tape.
5. Depress ENTER. The field program will be cleared from the CRT.

6. Repeat steps 4 and 5 until all of the field programs for the letter pair and a single record separator have been entered.

7. Start over with step 1 for another letter pair, until all field program groups have been entered.

8. The cursor will be in the status line.

9. Make a note of the MEM address.

Continue with the next page of program generation.

End

After the groups of field programs have been entered, note should be made of the MEM (memory) address. This address gives an indication of the amount of extended memory required for the terminal that will use the application program. If MEM is less than or equal to 10,000, the program can be used in a terminal equipped with 1K of extended memory. MEM must be less than or equal to 20,000 for use in a terminal equipped with 3K of extended memory. If the MEM address exceeds the available memory, consult Appendix D for hints on how to reduce the size of the program.

The steps for ending the program are:

1. Make a note of the address in the status line after MEM.

2. Type E for end.

3. Rewind and remove tape 1, the source tape.

4. Rewind tape 2, the program tape and leave it in place for program loading and testing.

Load and Test

The systems analyst must load the application program and test it. The first record on the program tape is the loader which is loaded manually with the shift/load key. The systems analyst then goes into program mode, types the three character program name (SAL for instance) and depress enter. The program tape is then automatically searched and the program is loaded. When the cassette stops advancing the system analyst rewinds and removes the application program cassette and loads the format and a blank cassette. The systems analyst should load and test each program and format to be sure they are working properly.

The steps for loading and testing an application program are:

1. The program tape should be on tape 2.
2. Advance the loader onto the CRT from the program tape (tape 2, adv rcd).
3. Depress shift/load to load the loader.
4. Depress job select, P, 2, space twice and enter.
5. Type the three character name you gave the program (SAL for instance) and depress enter.
6. The loader will search for the program name.
7. When the program is found it will be loaded by the loader.
8. When loading is complete, rewind and remove tape 2, the application program tape.
9. Load the format tape on tape 1.
10. Load a blank tape on tape 2.
11. Depress job select, turn program control and auto operation on, Depress F, 1, space, 2, space (or another suitable job selection) and enter.
12. Test each data field.

PROGRAM GENERATOR ERROR RECOVERY

The error recovery instructions in this section refer to errors that may occur during program generation or program testing. For general error recovery instructions consult section three of the Model 340 Operator's Manual.

FMT Format error, field program phase of program generation.

SITUATION	ACTION
The OP CODE is misspelled or is missing.	Depress ERROR RESET, correct the field program and depress enter.
The OP CODE is correct but the operation was not included during the operation phase of program generation.	Depress JOB SELECT. Program generation will have to be started over to include the missing operation.
Typing I, J, K, L, W, X, Y or Z in the status line	Depress ERROR RESET. The field letter pair you are using has already been used once. Enter the field programs under a different letter pair (and change the format accordingly) or depress JOB SELECT and start program generation over again.

LBL The field program label has already been used in the current field program group.

SITUATION	ACTION
The label is incorrect.	Depress ERROR RESET, correct the label and depress enter.

SITUATIONS	ACTION
The label is correct.	Change the label or use this field program in a different field program group.
The label error was intentionally created to correct a previously entered field program.	Depress ERROR RESET and ENTER (no intervening key depression). The field program on the CRT will replace the field program previously entered with the same label. When substitution occurs, all field programs following the replaced field program are destroyed and must be re-entered.

MOD The character you have typed is inappropriate at this point in program generation. Depress ERROR RESET and read the generator instructions in carefully.

PGM Program error.

SITUATION	ACTION
During program testing.	The current field program contains a table that does not meet the requirements of the operation. The program will have to be re-generated.
Operation phase just started.	The library tape is out of sequence. Program generation must be restarted. Depress JOB SELECT or ERROR RESET. Rewind both cassettes and reload the generator.

SITUATION	ACTION
Two operations have been specified that require the same memory locations.	Program generation must be restarted and one of the "overlapping" operators must be omitted. Depress JOB SELECT or ERROR RESET. Rewind both cassettes and reload the generator.
C as format input and a negative constant in a next format instruction.	Revise the application program or change the job selection to use a cassette tape as the format input.
The same operation has been specified twice.	Program generation must be restarted. Depress JOB SELECT or ERROR RESET. Rewind both cassettes and reload the generator.
During field program entry (I, J, K, L, W, X, Y or Z just typed in the status	Program generation must be started over to include GTO. Depress ERROR RESET or JOB SELECT. Rewind both cassettes and reload the generator.
During field program entry (a field program just entered and BUF $\geq$ 20 000)	Break the field programs into smaller groups and regenerate the program.
During table or field program entry (SYM $\leq$ 20 040)	The terminal does not have enough memory to generate the program.

RD Read error during the operation phase of program generation. Restart program generation using a fresh library tape.

TAB Table error during program generation.

SITUATION	PROBABLE CAUSE	ACTION
During the table phase.	Two tables with the same number or a table number starting with an alpha character.	Depress ERROR RESET Correct the table number. Depress ENTER.
During field program entry. The cursor indicates the table number that is causing the error.	The field program calls for a table that has not been entered.	
	Table number incorrect:	Depress ERROR RESET, correct the table number and depress enter.
	Table omitted:	Depress JOB SELECT. The program will have to be regenerated to include the missing table.

TP1 or There may be a hardware error on one of the cassette drives. If this happens at the beginning of a tape remove the cassette and check the protect tabs. You may be trying to write on a protected cassette. Replace the cassette, making sure it is firmly seated and the door is properly closed.

SITUATION	ACTION
Operation phase of program generation.	Depress ERROR RESET twice, correct the tape problem and depress ENTER.
Table phase or field program phase of program generation.	Input tape: Depress ERROR RESET twice, correct the tape problem and (if necessary) reposition the input tape before continuing. Output tape: Depress ERROR RESET twice, correct the tape problem and depress ENTER.

PROGRAM GENERATION WITH OVERLAYS

The program using overlays is divided into two modules, the resident and the overlays. The resident module contains all of the OP CODES (including OVL) that are used in the entire program, and any tables and field programs that are used in more than one overlay module. The resident module resides in memory for the duration of the program. After the resident module is loaded into memory, overlay module #1 is automatically loaded into memory. The memory used by the resident module is the base from which each of the overlay modules will start.

The overlay instruction can use any table previously defined in the field program. Each overlay module contains only the tables and field programs necessary for the subsequent format page(s). The overlay module is loaded into memory and stays there, until the next overlay is called for by the program. As a result, the total memory for an application is defined by the resident module plus the largest overlay module, since only one overlay is present in memory at any one time.

Because of these features, the source tape has a different structure. The preparation is the same as described under Preparation. Refer to Figure 2 for the source tape structure.

The steps for setting up program generation are:

1. Load the Sycor TAL Library cassette on tape 1 and a blank cassette on tape 2, making sure both are rewound.
2. Turn AUTO OPRT and MEM CTL off.
3. Depress JOB SELECT, F, SPACE, SPACE, SPACE, SPACE and ENTER.
4. Advance tape 1 (TAPE 1, ADV RCD).
5. Write the loader on tape 2 (TAPE 2, WRITE).
6. Load the loader (Depress SHIFT and LOAD simultaneously).
7. Depress JOB SELECT, P, 1, SPACE, SPACE, and ENTER.
8. Type GEN (the three character name for the generator) and depress ENTER.

9. The loader will search for the program name GEN.

10. When GEN is found it will be loaded by the loader.

11. Loading is complete.

12. Depress JOB SELECT, P, 1, 2, SPACE and ENTER.

13. The status line will display:

```
PROG  IN1  OUT2,  BUF  XX  XXX  SYM  YY  YYY  MEM  ZZ  ZZZ
```

14. Type the three character TAL program name (INV for instance).

15. Depress ENTER, the name will be written out to the program tape as a name record.

16. The CRT now displays a list of the program routines available in the Sycor Software Library.

17. Type in all of the required operations on the CRT, followed by a record separator (refer to the TAL Program FORM). Don't forget REG (if the registers or switches are being used).

18. Depress ENTER to start the generation of the program routines on to the object cassette.

19. As the operations are copied on cassette 2, their names will be erased from the CRT.

20. When all the names have been erased and an EOF appears in the status line, depress ERROR RESET.

21. Rewind and remove tape 1 (TAPE 1, REW).

22. Leave tape 2 in place to collect further program output.

23. Depress ENTER to end the operation phase of program generation.

24. The cursor will be in the status line.

25. Make a note of the MEM address.

NOTE: The following steps are included, if there are tables and field programs in the resident module; otherwise, skip to step 45.

26. Load the program source tape containing the tables and field programs on tape 1. (Refer to figure 4 for source tape format).

27. Type T for tables.

28. To automatically enter the tables, go to step 29. To manually enter the tables, go to step 30.

29. Turn PROG CTL on and depress ENTER. The tables will be converted to object form and written on Tape 2. Go to step 34.

30. Turn PROG CTL off.

31. Advance a table from tape 1 (TAPE 1, ADV RCD).

32. Depress ENTER. The table will not be converted to object form and written out to tape 2, until a block is filled. The block size varies depending upon memory allocation.

33. Repeat steps 31 and 32, until all of the tables and a single record separator have been entered.

34. The cursor will be in the status line.

35. Make a note of the MEM address.

36. Type I, or J, K or L, W or X or Y or Z in the status line.

37. To automatically enter the group, go to step 38. To manually enter the group, go to step 39.

38. Turn PROG CTL on and depress ENTER. Go to step 43.

39. Turn PROG CTL off.

40. Advance the first field program from the source tape.

41. Depress ENTER. The field program will be cleared from the CRT.

42. Repeat steps 40 and 41, until all of the field programs for the letter pair and a single record separator have been entered.

43. Start over with step 36 for another letter pair, until all field programs have been entered.

44. The cursor will be in the status line.

45. Make a note of the MEM address.

NOTE: Any letter group or field program name in the resident module cannot be used in any of the overlay modules.

46. Type R. (The R signifies that overlay modules will follow, this R goes on the status line but not in the same location as the T, E, etc.).

47. Type T (for tables).

48. Write a file separator on tape 2 to separate the resident module from the overlay module #1. (TAPE 2, SHIFT, FS).

49. Repeat the table phase for overlay #1. Steps 28-35.

50. Continue the field program phase for overlay #1. Steps 36-45.

51. Type E (End of overlay module).

    The terminal is in free form. The first format can be advanced from the source tape onto the CRT, or it may be created from the keyboard at this time.

52. Depress ENTER. (The page of format is written on the program tape). Continue entering format(s), until a single record separator has been entered. The cursor is now in the status line, and the generation of overlay module #2 can begin.

53. Type T. Steps 28-45.

    This sequence repeats until the last page of formats of the last overlay module has been entered on the object tape.

54. Depress TAPE 2, FS (file separator). The resulting object tape should be similar to the structure of the tape in Figure 5 .

PROGRAM LOADING WITH OVERLAYS

Because the file separator automatically rewinds the format input device, and the resident module and overlay module #1 are loaded together, it is necessary to change the structure of the object program tape. Refer to Figure 6 . This structure will insure that the loader, name record and resident module will be encountered only once.

The final version of the program tape is ready for testing.

1. Position the object tape immediately after the first file separator. A convenient way of advancing the tape to this position is a batch job selection with no output and AUTO OPRT off.
2. Depress JOB SELECT, F, SPACE, SPACE, SPACE, SPACE, and ENTER.
3. Advance the loader on the CRT.
4. Depress SHIFT and LOAD simultaneously.
5. Depress JOB SELECT.
6. Depress P.
7. Depress 1.
8. Depress SPACE, SPACE, and ENTER.
9. Type in the 3 character name (INV for example).
10. Depress ENTER, the resident module and overlay #1 is read into memory.
11. Depress JOB SELECT.
12. Switch PROG CTL and AUTO OPRT on.
13. Enter the required job selection as specified on the Format Layout Form.
14. The file separator will rewind the tape and the first page of format will be read onto the screen.

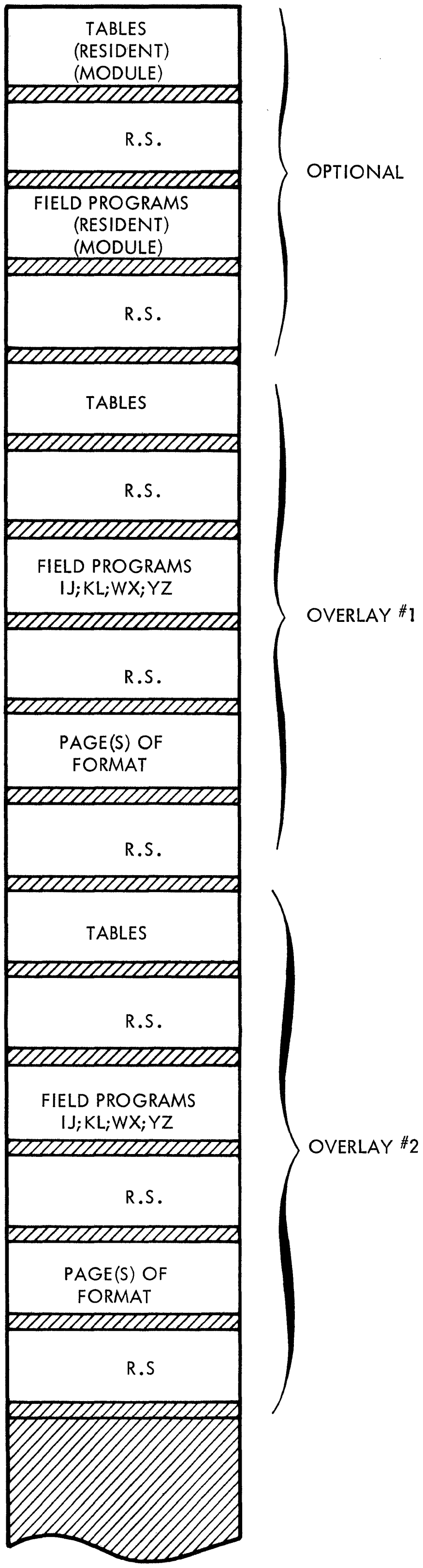


Figure 4 Source Tape Structure

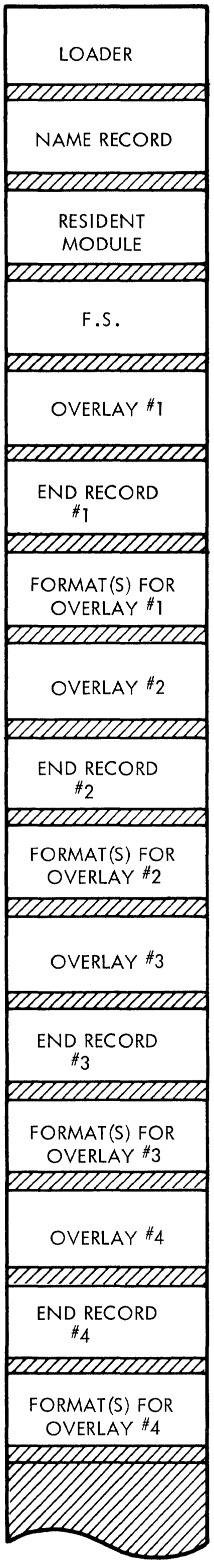


Figure 5 Object Tape

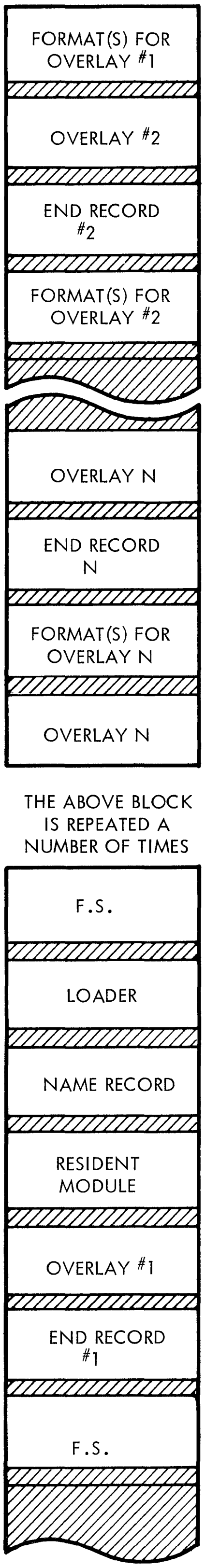


Figure 6 Final Object Tape

Additional Notes

APPENDIX

PRINTER VERTICAL TAB CODES

	0	1	2	3	4	5	6	7	8	9
0		PA	PB	PC	PD	PE	PF	PG	PH	PI
10	PJ	PK	PL	PM	PN	PO	AP	AA	AB	AC
20	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
30	AN	AO	BP	BA	BB	BC	BD	BE	BF	BG
40	BH	BI	BJ	BK	BL	BM	BN	BO	CP	CA
50	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK
60	CL	CM	CN	CO	DP	DA	DB	DC	DD	DE
70	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO
80	EP	EA	EB	EC	ED	EE	EF	EG	EH	EI
90	EJ	EK	EL	EM	EN	EO	FP	FA	FB	FC
100	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM
110	FN	FO	GP	GA	GB	GC	GD	GE	GF	GG
120	GH	GI	GJ	GK	GL	GM	GN	GO	HP	HA
130	HB	HC	HD	HE	HF	HG	HH	HI	HJ	HK
140	HL	HM	HN	HO	IP	IA	IB	IC	ID	IE
150	IF	IG	IH	II	IJ	IK	IL	IM	IN	IO
160	JP	JA	JB	JC	JD	JE	JF	JG	JH	JI
170	JJ	JK	JL	JM	JN	JO	KP	KA	KB	KC
180	KD	KE	KF	KG	KH	KI	KJ	KK	KL	KM
190	KN	KO	LP	LA	LB	LC	LD	LE	LF	LG
200	LH	LI	LJ	LK	LL	LM	LN	LO	MP	MA
210	MB	MC	MD	ME	MF	MG	MH	MI	MJ	MK
220	ML	MM	MN	MO	NP	NA	NB	NC	ND	NE
230	NF	NG	NH	NI	NJ	NK	NL	NM	NN	NO
240	OP	OA	OB	OC	OD	OE	OF	OG	OH	OI
250	OJ	OK	OL	OM	ON	OO				

Octal/Decimal Conversion Table

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63
10	64	65	66	67	68	69	70	71
11	72	73	74	75	76	77	78	79
12	80	81	82	83	84	85	86	87
13	88	89	90	91	92	93	94	95
14	96	97	98	99	100	101	102	103
15	104	105	106	107	108	109	110	111
16	112	113	114	115	116	117	118	119
17	120	121	122	123	124	125	126	127
20	128	129	130	131	132	133	134	135
21	136	137	138	139	140	141	142	143
22	144	145	146	147	148	149	150	151
23	152	153	154	155	156	157	158	159
24	160	161	162	163	164	165	166	167
25	168	169	170	171	172	173	174	175
26	176	177	178	179	180	181	182	183
27	184	185	186	187	188	189	190	191
30	192	193	194	195	196	197	198	199
31	200	201	202	203	204	205	206	207
32	208	209	210	211	212	213	214	215
33	216	217	218	219	220	221	222	223
34	224	225	226	227	228	229	230	231
35	232	233	234	235	236	237	238	239
36	240	241	242	243	244	245	246	247
37	248	249	250	251	252	253	254	255

PERIPHERAL DRIVER CHARACTER CONVERSION TABLE

USASCII			Peripheral Driver Program Name							
Character	Octal	Hex	#M9*	#M7	#MH	#MG	#MA	#MN	#MT	#CR
Space	040	20	40	20	15	20	040	14	15	
!	041	21	4F	52	40	77	041	20	57	12-8-7
"	042	22	7F	57	52	76	042	33	55	8-7
#	043	23	7B	13	13	13	043	61	52	8-3
$	044	24	5B	53	53	53	044	54	53	11-8-3
%	045	25	6C	34	74	74	045	52	35	0-8-4
&	046	26	50	60	37	32	046	15	17	12
'	047	27	7D	77	32	57	047	74	12	8-5
(	050	28	4D	35	75	35	050	55	74	12-8-5
)	051	29	5D	74	34	55	051	56	34	11-8-5
*	052	2A	5C	54	54	54	052	60	54	11-8-4
+	053	2B	4E	37	77	60	053	40	20	12-8-6
,	054	2C	6B	33	73	73	054	13	73	0-8-3
-	055	2D	60	40	57	52	055	17	40	11
.	056	2E	4B	73	33	33	056	16	33	12-8-3
/	057	2F	61	21	61	61	057	57	61	0-1
0	060	30	F0	12	00	00	060	00	00	0
1	061	31	F1	01	01	01	061	01	01	1
2	062	32	F2	02	02	02	062	02	02	2
3	063	33	F3	03	03	03	063	03	03	3
4	064	34	F4	04	04	04	064	04	04	4
5	065	35	F5	05	05	05	065	05	05	5
6	066	36	F6	06	06	06	066	06	06	6
7	067	37	F7	07	07	07	067	07	07	7
8	070	38	F8	10	10	10	070	10	10	8
9	071	39	F9	11	11	11	071	11	11	9
:	072	3A	7A	15	60	15	072	35	14	8-2
;	073	3B	5E	56	56	56	073	32	32	11-8-6
	074	3C	4C	76	36	36	074	72	60	12-8-4
=	075	3D	7E	17	17	75	075	53	13	8-6
	076	3E	6E	16	16	16	076	73	16	0-8-6
?	077	3F	6F	72	20	17	077	34	37	0-8-7

PERIPHERAL DRIVER CHARACTER CONVERSION TABLE

USASCII			Peripheral Driver Program Name							
Character	Octal	Hex	#M9*	#M7	#MH	#MG	#MA	#MN	#MT	#CR
	140	60	40	20	15	20	100	14	15	8-1
a	141	61	40	20	15	20	101	14	15	12-0-1
b	142	62	40	20	15	20	102	14	15	12-0-2
c	143	63	40	20	15	20	103	14	15	12-0-3
d	144	64	40	20	15	20	104	14	15	12-0-4
e	145	65	40	20	15	20	105	14	15	12-0-5
f	146	66	40	20	15	20	106	14	15	12-0-6
g	147	67	40	20	15	20	107	14	15	12-0-7
h	150	68	40	20	15	20	110	14	15	12-0-8
i	151	69	40	20	15	20	111	14	15	12-0-9
j	152	6A	40	20	15	20	112	14	15	12-11-1
k	153	6B	40	20	15	20	113	14	15	12-11-2
l	154	6C	40	20	15	20	114	14	15	12-11-3
m	155	6D	40	20	15	20	115	14	15	12-11-4
n	156	6E	40	20	15	20	116	14	15	12-11-5
o	157	6F	40	20	15	20	117	14	15	12-11-6
p	160	70	40	20	15	20	120	14	15	12-11-7
q	161	71	40	20	15	20	121	14	15	12-11-8
r	162	72	40	20	15	20	122	14	15	12-11-9
s	163	73	40	20	15	20	123	14	15	11-0-2
t	164	74	40	20	15	20	124	14	15	11-0-3
u	165	75	40	20	15	20	125	14	15	11-0-4
v	166	176	40	20	15	20	126	14	15	11-0-5
w	167	77	40	20	15	20	127	14	15	11-0-6
x	170	78	40	20	15	20	130	14	15	11-0-7
y	171	79	40	20	15	20	131	14	15	11-0-8
z	172	7A	40	20	15	20	132	14	15	11-0-9
	173	7B	40	20	15	20	133	14	15	12-0
|	174	7C	40	20	15	20	134	14	15	12-11
-0	175	7D	40	20	15	20	135	14	40	11-0
	176	7E	40	20	15	20	136	14	15	11-0-1
DEL	177	7E	40	20	15	20	137	14	15	12-9-7

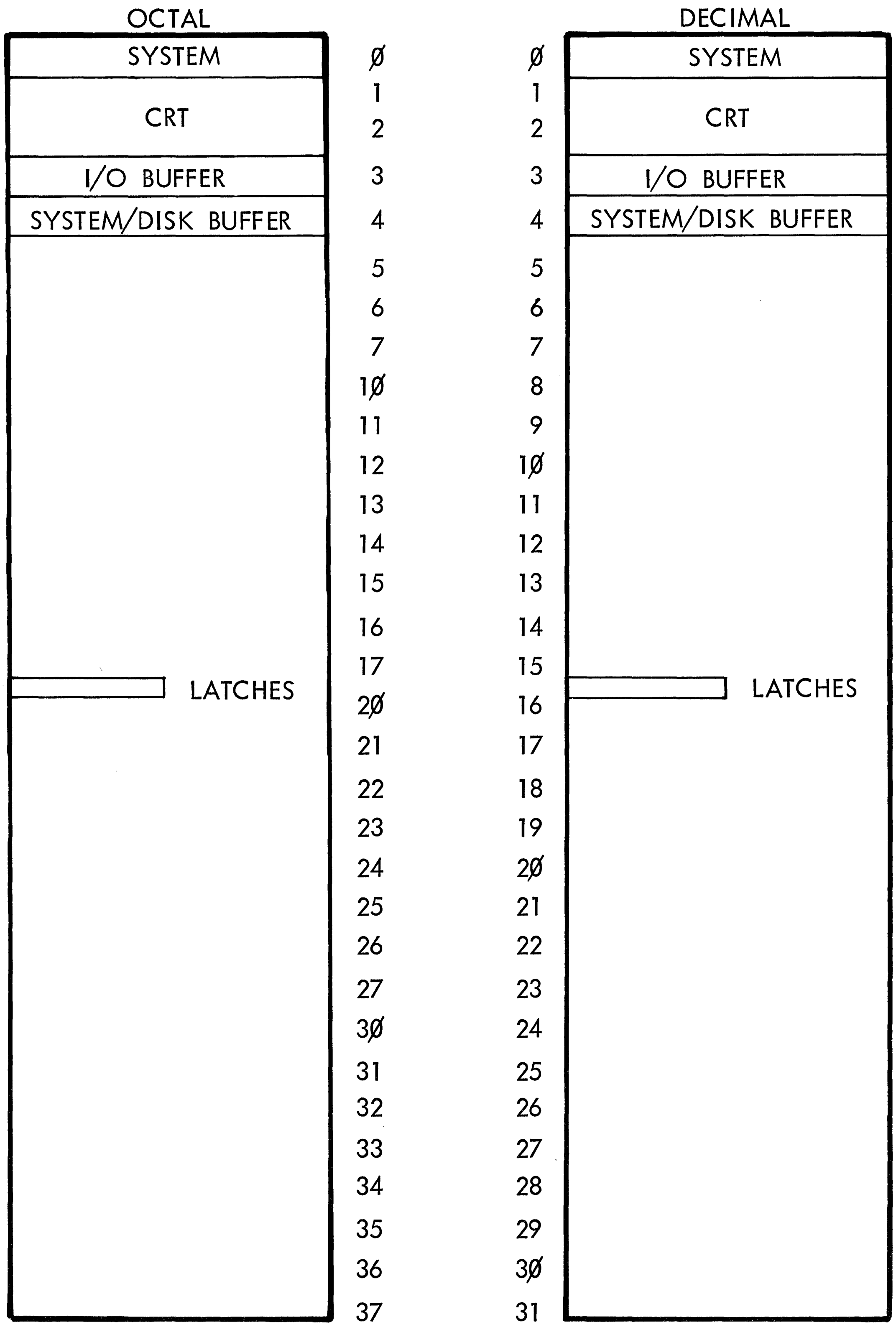
PERIPHERAL DRIVER CHARACTER CONVERSION TABLE

USASCII			Peripheral Driver Program Name							
Character	Octal	Hex	#M9*	#M7	#MH	#MG	#MA	#MN	#MT	#CR
@	100	40	7C	14	14	14	100	12	72	8-4
A	101	41	C1	61	21	21	101	21	21	12-1
B	102	42	C2	62	22	22	102	22	22	12-2
C	103	43	C3	63	23	23	103	23	23	12-3
D	104	44	C4	64	24	24	104	24	24	12-4
E	105	45	C5	65	25	25	105	25	25	12-5
F	106	46	C6	66	26	26	106	26	26	12-6
G	107	47	C7	67	27	27	107	27	27	12-7
H	110	48	C8	70	30	30	110	30	30	12-8
I	111	49	C9	71	31	31	111	31	31	12-9
J	112	4A	D1	41	41	41	112	41	41	11-1
K	113	4B	D2	42	42	42	113	42	42	11-2
L	114	4C	D3	43	43	43	114	43	43	11-3
M	115	4D	D4	44	44	44	115	44	44	11-4
N	116	4E	D5	45	45	45	116	45	45	11-5
O	117	4F	D6	46	46	46	117	46	46	11-6
P	120	50	D7	47	47	47	120	47	47	11-7
Q	121	51	D8	50	50	50	121	50	50	11-8
R	122	52	D9	51	51	51	122	51	51	11-9
S	123	53	E2	22	62	62	123	62	62	0-2
T	124	54	E3	23	63	63	124	63	63	0-3
U	125	55	E4	24	64	64	125	64	64	0-4
V	126	56	E5	25	65	65	126	65	65	0-5
W	127	57	E6	26	66	66	127	66	66	0-6
X	130	58	E7	27	67	67	130	67	67	0-7
Y	131	59	E8	30	70	70	131	70	70	0-8
Z	132	5A	E9	31	71	71	132	71	71	0-9
0	133	5B	4A	75	35	12	133	75	56	12-8-2
0	134	5C	6A	72-34	72-34	37	134	77	36-34	0-8-2
0	135	5D	5A	55	55	34	135	76	75	11-8-2
0	136	5E	5F	36	76	40	136	37	76	11-8-7
0	137	5F	40	20	15	20	137	14	77	0-8-5

PERIPHERAL DRIVER CHARACTER CONVERSION TABLE

USASCII			Peripheral Driver Program Name							
Character	Octal	Hex	#M9*	#M7	#MH	#MG	#MA	#MN	#MT	#CR
NUL	000	00	00	32-40	72-40	72-40	000	36-40	36-40	12-0-9-8-1
SOH	001	01	01	32-01	72-01	72-01	001	36-01	36-01	12-9-1
STX	002	02	02	32-02	72-02	72-02	002	36-02	36-02	12-9-2
ETX	003	03	03	32-03	72-03	72-03	003	36-03	36-03	12-9-3
EOT	004	04	37	32-04	72-04	72-04	004	36-04	36-04	9-7
ENQ	005	05	2D	32-05	72-05	72-05	005	36-05	36-05	0-9-8-5
ACK	006	06	2E	32-06	72-06	76-06	006	36-06	36-06	0-9-8-6
BEL	007	07	2F	32-07	72-07	72-07	007	36-07	36-07	0-9-8-7
BS	010	08	16	32-10	72-10	72-10	010	36-10	36-10	11-9-6
HT	011	09	05	32-11	72-11	72-11	011	36-11	36-11	12-9-5
LF	012	0A	25	32-12	72-12	72-12	012	36-12	36-12	0-9-5
VT	013	0B	0B	32-13	72-13	72-13	013	36-13	36-13	12-9-8-3
FF	014	0C	0C	32-14	72-14	72-14	014	36-14	36-14	12-9-8-4
CR	015	0D	0D	32-15	72-15	72-15	015	36-15	36-15	12-9-8-5
SO	016	0E	0E	32-16	72-16	72-16	016	36-16	36-16	12-9-8-6
SI	017	0F	0F	32-17	72-17	72-17	017	36-17	36-17	12-9-8-7
DLE	020	10	10	32-20	72-20	72-20	020	36-20	36-20	12-11-9-8-1
DC1	021	11	11	32-21	72-21	72-21	021	36-21	36-21	11-9-1
DC2	022	12	12	32-22	72-22	72-22	022	36-22	36-22	11-9-2
DC3	023	13	13	32-23	72-23	72-23	023	36-23	36-23	11-9-3
DC4	024	14	3C	32-24	72-24	72-24	024	36-24	36-24	9-8-4
NAK	025	15	3D	32-25	72-25	72-25	025	36-25	36-25	9-8-5
SNY	026	16	32	32-26	72-26	72-26	026	36-26	36-26	9-2
ETB	027	17	26	32-27	72-27	72-27	027	36-27	36-27	0-9-6
CAN	030	18	18	32-30	72-30	72-30	030	36-30	36-30	11-9-8
EM	031	19	19	32-31	72-31	72-31	031	36-31	36-31	11-9-8-1
SUB	032	1A	3F	32-32	72-32	72-32	032	36-32	36-32	9-8-7
ESC	033	1B	27	32-33	72-33	72-33	033	36-33	36-33	0-9-7
FS	034	1C	22	32-35	72-35	72-35	034	36-35	36-35	11-9-8-4
GS	035	1D	1D	32-41	72-41	72-41	035	36-41	36-41	11-9-8-5
RS	036	1E	1E	32-36	72-36	72-36	036	36-36	36-36	11-9-8-6
US	037	1F	1F	32-37	72-37	72-37	037	36-37	36-37	11-9-8-7

MEMORY MAP


OCTAL
SYSTEM
CRT
I/O BUFFER
SYSTEM/DISK BUFFER
LATCHES
0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
22
23
24
25
26
27
30
31
32
33
34
35
36
37
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
DECIMAL
SYSTEM
CRT
I/O BUFFER
SYSTEM/DISK BUFFER
LATCHES

MEMORY MAP

OCTAL			DECIMAL
BOOT LOADER (ROM)	40	32	BOOT LOADER (ROM)
MIXER	41	33	MIXER
	42	34	
	43	35	
SYSTEM	44	36	SYSTEM
KEYBOARD	45	37	KEYBOARD
	46	38	
RESIDENT LOADER	47	39	RESIDENT LOADER
JOB SELECT/SYSTEM	50	40	JOB SELECT/SYSTEM
CASSETTE I/O	51	41	CASSETTE I/O
	52	42	
PRINTER I/O	53	43	PRINTER I/O
	54	44	
SYSTEM I/O & ERROR MESSAGES	55	45	SYSTEM I/O & ERROR MESSAGES
	56	46	
DISK DRIVER	57	47	DISK DRIVER
DISK HANDLER	60	48	DISK HANDLER
	61	49	
	62	50	
	63	51	
	64	52	
FREE-FORM	65	53	FREE-FORM
	66	54	
	67	55	
RESIDENT CONTROL PROGRAM	70	56	RESIDENT CONTROL PROGRAM
	71	57	
	72	58	
	73	59	
	74	60	
	75	61	
	76	62	
	77	63	

cut here

REVISION REQUEST AND COMMENT FORM

Please send me copies of all revisions and additions to the

MODEL 340 PROGRAMMER'S MANUAL

ISSUED SEPTEMBER, 1974

---

fold here

Suggested improvements: ....
....
....
....

Errors noted: ....
....
....
....

General comment: ....
....
....
....

---

fold here

Please print or type:

Name: ....
Title: .... Phone: ....
Company: ....
Address: ....
City: .... State: .... Zip Code: ....

cut here

When complete, please fold and staple or tape closed with the return address facing outward. Thank you for your selection of Sycor Data Communications Systems.

FIRST CLASS
PERMIT NO. 1531
ANN ARBOR
MICHIGAN

Business Reply Card *No postage stamp necessary if mailed in the United States*

Postage will be paid by

SYCOR INC.
TRAINING AND DOCUMENTATION DEPARTMENT
100 PHOENIX DRIVE
ANN ARBOR, MICHIGAN 48104

SYCOR INC
100 Phoenix Drive
Ann Arbor, Michigan 48104