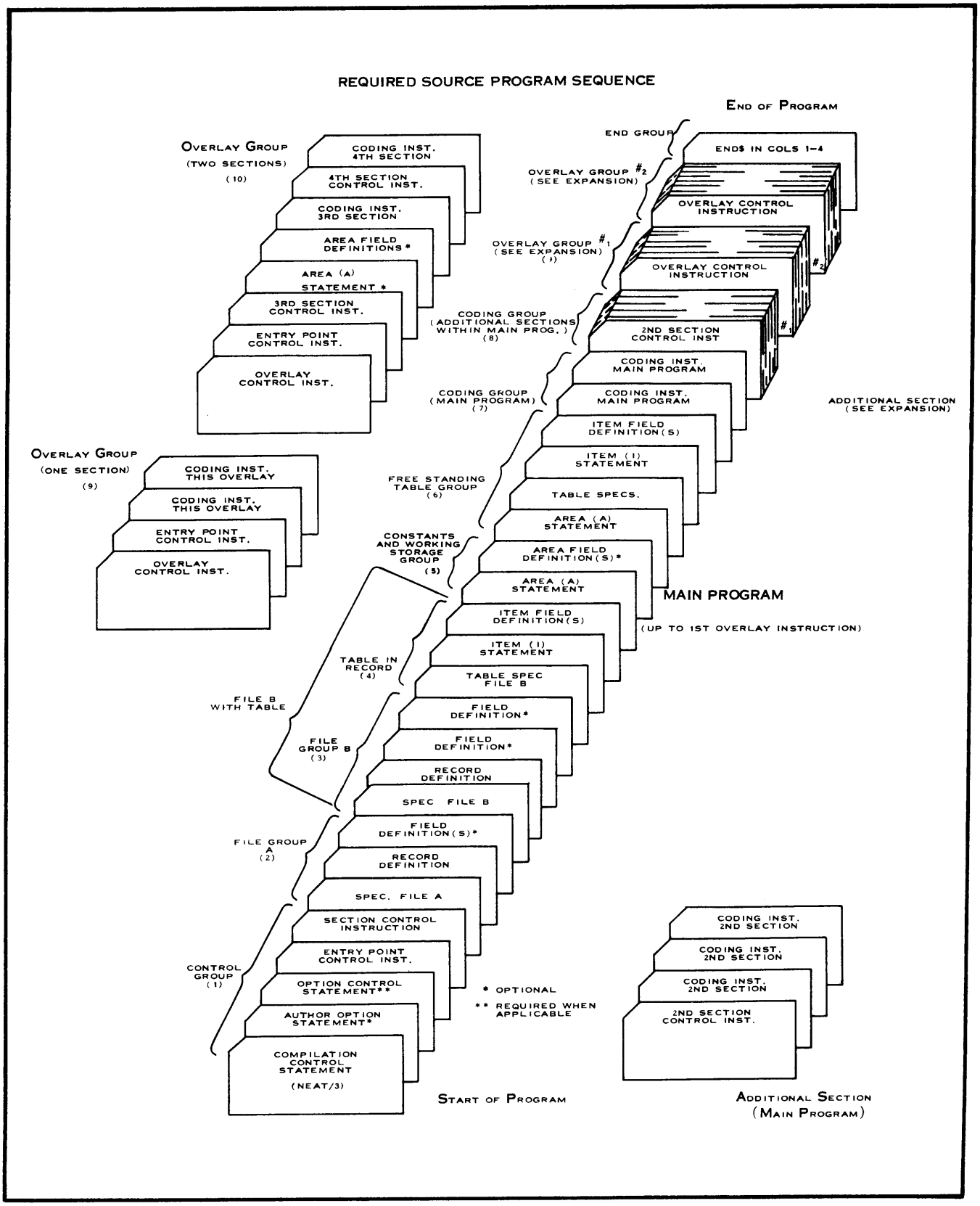
ORGANIZATION OF A SOURCE PROGRAM

INTRODUCTION

Previous chapters of this manual have included explanations of the various source statements (control instructions, specifications, data, and coding) which distinguish the general categories of program information. As a word of explanation, the terms source statement and source line, which are used throughout this manual, are synonymous. Each represents one line of program information as handwritten on any of the various worksheets. Once the source line or statement is punched into a data recording media, a source record is created. Each source line, statement, or record then represents an integral part of the total input known as the source program.

To create an object program from the source program, the source records must be presented to the compiler in a predetermined sequence. The programmer assigns page and line numbers which follow this required sequence.

The illustration used for this discussion appears on the following page. It shows the correct sequence of source statements for an initial compilation. Although the illustration uses punched-cards, the sequence does not vary when using paper tape. For the purpose of showing various possible arrangements of source statements, the illustration includes a main program and two overlays.


REQUIRED SOURCE PROGRAM SEQUENCE
END OF PROGRAM
END GROUP
END$ IN COLS 1-4
OVERLAY GROUP #2
(SEE EXPANSION)
OVERLAY CONTROL INSTRUCTION
OVERLAY GROUP #1
(SEE EXPANSION)
(1)
OVERLAY CONTROL INSTRUCTION
#2
#1
2ND SECTION CONTROL INST
CODING GROUP
(ADDITIONAL SECTIONS WITHIN MAIN PROG.)
(8)
CODING INST. MAIN PROGRAM
CODING GROUP
(MAIN PROGRAM)
(7)
CODING INST. MAIN PROGRAM
ADDITIONAL SECTION
(SEE EXPANSION)
ITEM FIELD DEFINITION(S)
ITEM (I) STATEMENT
FREE STANDING TABLE GROUP
(6)
TABLE SPECS.
AREA (A) STATEMENT
CONSTANTS AND WORKING STORAGE GROUP
(5)
AREA FIELD DEFINITION(S)*
AREA (A) STATEMENT
MAIN PROGRAM
(UP TO 1ST OVERLAY INSTRUCTION)
ITEM FIELD DEFINITION(S)
ITEM (I) STATEMENT
TABLE IN RECORD
(4)
TABLE SPEC FILE B
FILE B WITH TABLE
FIELD DEFINITION*
FIELD DEFINITION*
FILE GROUP B
(3)
RECORD DEFINITION
SPEC FILE B
FIELD DEFINITION(S)*
FILE GROUP A
(2)
RECORD DEFINITION
SPEC. FILE A
SECTION CONTROL INSTRUCTION
ENTRY POINT CONTROL INST.
CONTROL GROUP
(1)
OPTION CONTROL STATEMENT**
AUTHOR OPTION STATEMENT*
COMPILATION CONTROL STATEMENT
(NEAT/3)
* OPTIONAL
** REQUIRED WHEN APPLICABLE
START OF PROGRAM
OVERLAY GROUP
(TWO SECTIONS)
(10)
CODING INST. 4TH SECTION
4TH SECTION CONTROL INST.
CODING INST. 3RD SECTION
AREA FIELD DEFINITIONS *
AREA (A) STATEMENT *
3RD SECTION CONTROL INST.
ENTRY POINT CONTROL INST.
OVERLAY CONTROL INST.
OVERLAY GROUP
(ONE SECTION)
(9)
CODING INST. THIS OVERLAY
CODING INST. THIS OVERLAY
ENTRY POINT CONTROL INST.
OVERLAY CONTROL INST.
CODING INST. 2ND SECTION
CODING INST. 2ND SECTION
CODING INST. 2ND SECTION
2ND SECTION CONTROL INST.
ADDITIONAL SECTION
(MAIN PROGRAM)

As an aid in understanding the card by card illustration, the following outline is presented to show the various groups (control, file, etc.) of source statements that may be found in a source program. The sequence of the source statements in the outline is established by reading across and down the page, beginning with the control group (1) and terminating with the END group (11). For example, in the control group, the required initial source statement is the compilation control statement. This is followed by the author option statement (if desired), option control statement (if applicable), entry control instructions, and section control instruction (if applicable) in that order.

OUTLINE FOR SOURCE PROGRAM ORGANIZATION

NO.	GROUP	REQUIRED INITIAL SOURCE STATEMENT	OTHERS REQUIRED	OPTIONAL
1	Control	Compilation Control Statement	Option Control Statement (If Applicable) Entry Control Instruction Section Control Instruction (If Applicable)	Author Option Statement
2 3	File A File B	File Specifications	Record Definition	Field Definitions
4	Table in File Record	Table Specifications	Item (I) Statement	Item Field Definitions
5	Constants and Working Storage	Area (A) Statement		Area Field Definitions
6	Free-Standing Table	Area (A) Statement	Table Specifications Item (I) Statement Item Field Definitions	

NO.	GROUP	REQUIRED INITIAL SOURCE STATEMENT	OTHERS REQUIRED	OPTIONAL
7	Main Program Coding	Coding Instructions		
8	Main Program (Additional Sections) Coding	Section Control Instruction	Coding Instructions	
9	First Over-Lay - with One Section	Overlay Control Instruction	Entry Control Instruction Coding Instructions	
10	Second Overlay - with Two Sections	Overlay Control Instruction	Entry Control Instruction Section Control Instruction Coding Instructions Section Control Instruction Coding Instructions	Area (A) Statement Area Field Definitions
11	End	End$		

The minimum source program is made up of the following groups:

- Control (1)
- File (2)
- Main Program Coding (7)
- END (11)

A discussion of the source statements within the various groups of the illustrated program follows:

CONTROL GROUP (1)

Compilation Control Statement

The NEAT/3 compilation control statement, which is coded on compiler specification sheet 1, must always be the first source statement of the main program presented to the compiler. As the initial statement, it must always be assigned page/line number 000000. If the initial source statement is not the compilation control statement, the compiler halts with an error display.

Author Option Statement (Optional)

If the programmer chooses to include his name in the source program, the author option statement is provided for this purpose and must always follow the compilation control statement. If used, this source statement is coded at the bottom of compiler specification sheet 1 and is assigned page/line number 000001. Columns 18 through 23 are filled with the word AUTHOR followed by the author's name in columns 24 through 43.

Option Control Statement

The option control statement, which is an extension of the compilation specification statement, is coded on compiler specification sheet 2 (reverse side of compiler control sheet 1). This statement must follow the sheet 1 if the source program contains any of the special set of instructions available as optional hardware features in more advanced members of the NCR Century Series.

Entry Control Instruction

An entry control instruction must be presented to the compiler immediately following both the compilation specification statement (and related statements) for the main program and each overlay control statement.

Normally, when a NEXTDO is used to call a program (e.g. NEXTDO PROGNAME), the execution of the program begins with the instruction referenced in the first entry control instruction following the compilation specification statement. If the programmer chooses to start at some other entry point in the main program, he may specify the desired entry point as qualified by the program name, e.g. NEXTDO PROGNAME.ENTPNTNAME. (The term PROGNAME represents the desired program name and the qualifier, ENTPNTNAME, represents the desired entry point.) This paragraph applies to an entry point in a main program only. The application of an entry point in an overlay is explained later in this chapter.

Section Control Instruction

The program may contain one or more sections; if the entire source program consists of only one section, a section control instruction is not necessary. It may be used purely for documentation. However, if multiple sections are used (as in this chapter), the section control instruction must be presented, both here and where necessary, to indicate to the compiler that subsequent statements are to be included in a section. When assigning sections, discretion should be used to ensure that they begin with a different source-statement category. For example, do not begin a section within a group of file specifications or between the file specifications and their related record definitions.

Next in the required sequence are the various source statements that can be used to describe each file in the source program. These statements are separated into two groups - file A without tables and file B with tables.

FILE (A and first half of FILE B) GROUP (2 & 3)

File Specifications

All file specifications must appear in the main program and each must precede the definition of the records within that file. The required initial source statement(s) of each file group is the file specification. These file specifications may require only one source statement as shown in file group A or more than one as shown in file group B.

Record Definition

A related record definition must immediately follow each set of file specifications. This statement, coded on a data layout sheet, defines the length of the records in the file.

Field Definitions (Optional)

Following each record definition may be a series of field definitions if desired by the programmer. These statements, coded on a data layout sheet, define the fields within the record.

TABLE IN FILE RECORD (second half of FILE B) GROUP (4)

Table Specifications

Next in sequence and similar in form to the file specification statements are the table specifications which define the table within the record. The table specifications must immediately follow the record and field definitions of the file to which the table is related.

Item (I) Statement

Following next is the item (I) statement. This statement is coded on a data layout sheet and must follow the table specifications to which it refers.

Item Field Definition(s) (Optional)

After the I statement and relative to it are the item field definition(s), coded on data layout sheets, which define the fields within the items of the table.

CONSTANTS AND WORKING STORAGE GROUP (5)

Area (A) Statement

A programmer may choose to have an area in memory reserved for constants and/or working storage. Since constants and working storage areas should be described independently of the files and tables, this group is presented next - prior to the precedural instructions. The initial entry in this group, the area (A) statement, must be coded on a data layout sheet. This statement defines the length required for the area and reserves the necessary memory space.

Area Field Definition(s) (Optional)

The programmer may use the area field definition(s) to divide an area of memory into fields. These fields are described on data layout sheets and, if used, must follow their repsective area descriptions.

MEMORY RESIDENT TABLE GROUP (6)

Area (A) Statement

If the programmer chooses to use a memory resident table, a definition of the length required for this table area must precede the table specifications. The area (A) statement, coded on the data layout sheet, is the initial source statement of this group.

Table Specifications

Next in sequence must be the table specifications which are coded on the table specification sheet.

Item (I) Statement

Following next must be the item (I) statement. This source statement is coded on the data layout sheet and must follow the specifications for the table.

Item Field Definition(s)

After the item (I) statement, and relative to it, must follow the item field definition(s). These source statements, coded on data layout sheets, define the fields within the items of the table.

MAIN PROGRAM CODING GROUP (7)

Coding Instructions

Up to this point, the format of the data (files, records, fields, tables, items, and constants) has been defined to the compiler. The remaining source statements are procedural instructions.

NOTE: If no additional sections or overlays were needed, the END$ statement would be presented next to signify the end of the source program.

MAIN PROGRAM (Additional Sections) CODING GROUP (8)

For the purpose of this chapter, a second section is inserted into the source program sequence to illustrate a method in which coding for the main program may be accomplished by more than one programmer.

Section Control Instruction

As discussed earlier in the control group, sections may be used to permit more than one programmer to use duplicate reference tags freely within the same source program. The initial source statement of each additional section must be a section control (SECT) instruction. This instruction indicates to the compiler both the end of the previous section and the beginning of a new section.

Coding Instructions

The procedural instructions for this section of the main program must be presented next.

NOTE: If no additional sections or overlays were needed, the END$ statement would be presented next to signify the end of the source program. Except for the END$ statement, this concludes the explanation of source statements belonging to the main program.

Following the main program there may be, at the discretion of the programmer, groups of optional source statements which concern the use of the overlay. As shown in the illustration, the first overlay contains one section (of coding only) and the second overlay contains two sections.

FIRST OVERLAY GROUP - WITH ONE SECTION (9)

Overlay Control Instruction

The initial source statement in all overlays must be the overlay control instruction. This instruction, coded on either a coding sheet or a data layout sheet, is used to inform the compiler that the statements which follow are to be incorporated in an overlay. In addition, this instruction automatically indicates to the compiler that the previous section is ended and a new section is begun; therefore, it also serves as a section control instruction.

Entry Control Instruction

Immediately following the overlay control instruction must be an entry control instruction to inform the compiler of each entry point into this overlay.

NOTE: Although not shown in this overlay, the area (A) statement and the area field definition(s), in that order, could also be included next in the sequence, if desired.

Coding Instructions

Since there are no data definitions within this overlay, the procedural instructions for this overlay must be presented next.

NOTE: If no additional sections or overlays were needed, the END$ statement would be presented next to signify the end of the source program.

SECOND OVERLAY GROUP - WITH TWO SECTIONS (10)

Overlay Control Instruction

The initial source statement in all overlays must be the overlay control instruction. This instruction is used to inform the compiler that the previous section is ended and a new section is begun; therefore, it also serves as a section control instruction.

Entry Control Instruction

Another entry control instruction must be presented next to inform the compiler of each entry point into this overlay.

Section Control Instruction

At his discretion (at any point in the source program), the programmer indicates the beginning of a new section by presenting a section control instruction; this is discussed previously for the main program.

Area (A) Statement (Optional)

The programmer may desire to reserve a work or storage area in this overlay. These areas are not associated with the previously mentioned records or files in the main program and are associated only with this overlay. This area (A) statement must appear prior to the coding instructions in this overlay.

Area Field Definition(s) (Optional)

The area in memory (mentioned above) may be divided into fields. These fields are described by field definition statements and must follow their respective area descriptions.

Coding Instructions

The procedural instructions for this section of the overlay must now be presented to the compiler. In addition to the data defined in the main program, the data defined in this overlay can also be manipulated by these instructions. However, it is important to note that data defined in this overlay cannot be manipulated by the coding in the main program.

Section Control Instruction

The programmer may include more than one section within an overlay. For the purpose of this discussion, this overlay includes a second section - possibly to divide the coding work within the overlay. To do this the programmer must present a section control instruction at this point in the source program.

Coding Instructions

The procedural instructions for this section of the overlay are now presented to the compiler.

END GROUP (11)

END$

The last source statement in the source program must be the END$. Coded on a standard coding sheet (in columns 1-4), this statement signifies to the compiler the end of the source program.

PUNCHING PUNCH CARD SOURCE PROGRAM RECORDS

Before the various source program records can be entered into the NCR Century computer in the required sequence, they must be properly punched in a format acceptable to SPUR (Source Program Utility Routines).

This document describes the means and methods required for converting source lines, as prepared by the programmer, into precise program input records.

It is assumed that the programmer has properly filled out the necessary NEAT/3 source documents and that each source line contains a page and line number to guarantee that SPUR's sort routine sequences each line properly.

CARD PUNCHING EQUIPMENT

There are various machines and combinations of machines capable of recording information on punched cards. Although a keypunching device can be actuated by parent equipment (such as an accounting machine), this is not practical when punching source lines. Instead, it is more practical to punch cards directly by using one of the available keypunches.

SITE STANDARD CODE

Punch all control items and source statements in the site standard code (Hollerith Extended "A" Set or "H" Set). This code, set up during the initial installation, must be a site specified code. The procedures for establishing a site standard code are described in the Operators Manual. Normally, punch card source lines will be punched in the NCR Century Series standard Extended "A" Set. This code is illustrated in the document "Punch Card Files" under FILES, tab 1.

PROGRAM CARD

Without Alternate Program

When the alternate program feature is not available on the keypunch, you must punch the source lines contained on the coding and data layout sheets under primary program control. Punch all other source lines without program control.

The following is a suggested way to layout the program card. Your individual needs will dictate the most efficient layout for you.

PRIMARY PROGRAM (CODING AND DATA WORKSHEET)

AUTO. FUNC.	COLUMNS	MODE	FIELD
Dup	1-3	N	Page
	4-6	N	Line
Dup	7	A	C or D
	8-17	A	Reference
	18-23	A	Code and Location or Operation
	24-29	A	Start of Operands or Length and DP
	30	A	Operands or Type
	31-50	A	Operands or Value
	51-74	A	Comments
	75-80	A	Identification

With Alternate Program

When an alternate program feature is available on the keypunch, use the primary program (upper portion of program card) when punching source lines from the coding sheet and the alternate program (lower portion of program card) when punching source lines from the data layout sheet. Punch all other source lines without program control.

The following is a suggested way to layout the program card. Your individual needs will dictate the most efficient layout for you.

PRIMARY PROGRAM (CODING SHEET)

AUTO. FUNC.	COLUMNS	FIELD	MODE
Dup	1-3	Page	N
	4-6	Line	N
Dup	7	C	A
	8-17	Reference	A
	18-23	Operation	A
	24-50	Operands	A
	51-74	Comments	A
	75-80	Ident.	A

NOTE:

A = Alpha, N = Numeric

ALTERNATE PROGRAM (DATA SHEET)

AUTO. FUNC.	COLUMNS	FIELD	MODE
Dup	1-3	Page	N
	4-6	Line	N
Dup	7	D	A
	8-17	Reference	A
	18	Code	A
	19-23	Location	N
	24-29	Length, DP	N
	30	Type	A
	31-50	Value	N
	51-74	Comments	A
	75-80	Ident.	A

PUNCHING PROCEDURES

General Procedures

Providing the page number in columns 1 through 3 and the worksheet code in column 7 do not change from one source line to the next, duplicate them from the preceding card.

To enter a delete digit in column 74, skip to column 75 and backspace to column 74.

Special Characters

When using the NCR Century Series standard Hollerith Extended "A" Set, NEAT/3 source records may require (depending on equipment) the use of special characters not included on the keyboard. These characters and their respective codes are illustrated in the following table.

SPECIAL CHARACTER		DIGITS	SPECIAL CHARACTER		DIGITS
[	....	12, 2, 8	\	....	0, 2, 8
<	....	12, 4, 8	-	....	0, 5, 8
(	....	12, 5, 8	>	....	0, 6, 8
+	....	12, 6, 8	?	....	0, 7, 8
!	....	12, 7, 8	:	....	2, 8
]	....	11, 2, 8	# ≠	....	3, 8
)	....	11, 5, 8	'	....	5, 8
;	....	11, 6, 8	=	....	6, 8
^	....	11, 7, 8	"	....	7, 8

CORRECTION PROCEDURES FOR PUNCHING ERRORS

Correct errors on punched card entries found prior to compilation by physically replacing the card in error with a correctly punched substitute.

PUNCHING PAPER TAPE SOURCE PROGRAM RECORDS

Before the various source program records can be entered into the NCR Century computer in the required sequence, they must be properly punched in a format acceptable to SPUR (Source Program Utility Routine). In addition, control lines to be input to Monitor must also be punched in the proper format.

This publication describes the means and methods required for converting source lines, as prepared by the programmer, into precise program input records using an NCR carriage-type accounting machine wired to a punched paper tape recorder.

It is assumed that the programmer has properly filled out the necessary NEAT/3 source documents and that each source line contains a page and line number to guarantee that SPUR's sort routine sequences each line properly.

It is also assumed that the wiring of the accounting machine is standard. For a discussion of the types of wiring classified as standard, see the TECHNIQUES AND PROCEDURES REFERENCE MANUAL, GENERAL, "Standard Accounting Machine Wiring".

CODE SETS

Site Standard Code

All control items (NEXTDO, etc.) and source statements must be punched in the site standard code. This code, set up during the initial installation and used primarily by SPUR and Monitor, must be a site-specified code. The procedure for establishing a site standard code is described in the UTILITY ROUTINES REFERENCE MANUAL, DATA AND MEDIA ORIENTED, pub. no. 6.

Definition of Terms

The following list provides definitions of terms with which the punched paper tape user should be familiar:

- Data Character - A character which is regarded as data; part of a record.
- Control Characters - A non-data character used to format data characters into records.
- Invalid Character - A character which is not defined within the site-specified code set; any character which is not a data or control character.
- Record Termination Character - A control character which is recognized by NCR Century paper tape software as an end-of-record indicator. The record termination character of one record is also used as the originating control character for the next record on the tape.

CONTROL CHARACTERS

Control characters are used to separate data records on punched paper tape. The basic control characters for the standard NCR Century USASI code and the NCR 315 General Purpose (GP) code sets, and their meanings to NCR Century Punched Paper Tape software, are shown in the following table.

Basic Control Characters	NCR Century USASI Code	NCR 315 GP Code*	Internal Hexadecimal Representation
Tape Feed (Null)	NUL	TAPE FEED	80
Delete	DEL	DELE	FF
Escape	ESC	---	9B
End-of-Media	EOM	STOP	99
End-of-File	FS	COMP	9C
Lower-Shift	--	LOWER-SHIFT	8F
Upper-Shift	--	UPPER-SHIFT	8E
Record Termination Characters			
Line Feed (New Line)	EOR	---	8A
Carriage Return	CR	---	8D
Horizontal Tabulation (Tab)	EOF	---	89
Void Data	CAN	PUT	98
Synchronize	SYN	CLEAR	96
End-of-Record (Rec. separate)	RS	CARRIAGE RET	9E
End-of-Field (Unit separate)	US	TAB	9F

*NCR 315 General Purpose code is available through the Site Code Change Routine. See the UTILITY ROUTINES REFERENCE MANUAL, DATA AND MEDIA ORIENTED, pub. no. 6.

Not all of the control characters described in the table above apply to all code sets. The user must know the limitations of the particular set he is using. For example, only certain control characters can be used with NEAT/3 source records. The characters and their meanings to NCR Century Punched Paper Tape software are shown in the following table.

Name of Control Character	NCR Century USASI Code	Internal Hexadecimal Representation
End-of-File (File Separator)	FS	9C
End-of-Record (Record Separator)	RS	9E
End-of-Media	EOM	99
Void Data	CAN	98
Delete	DEL	FF
Tape Feed (Null)	NUL	80
Escape	ESC	9B
Carriage Return	CR	8D
End-of-Field (Unit Separator) and all other control characters	US	9F

The NCR Century standard code set is an 8-channel, even-parity, one-shift type and uses all of the aforementioned control characters except the Lower-Shift and Upper-Shift characters. Any other code set must be completely defined by the user. (See the UTILITY ROUTINES REFERENCE MANUAL, DATA AND MEDIA ORIENTED, pub. no. 6.)

NCR CENTURY [USASI] SITE STANDARD PAPER TAPE CODE

CHARACTER I/D	8	7	6	5	4	·	3	2	1	Internal value hex
SP (Φ)	●		●			·				20
!			●			·			●	21
"			●			·		●		22
# (£)	●		●			·		●	●	23
$			●			·	●			24
%	●		●			·	●		●	25
&	●		●			·	●	●		26
'			●			·	●	●	●	27
(			●		●	·				28
)	●		●		●	·			●	29
*	●		●		●	·		●		2A
+			●		●	·		●	●	2B
,	●		●		●	·	●			2C
-			●		●	·	●		●	2D
.			●		●	·	●	●		2E
/	●		●		●	·	●	●	●	2F
0			●	●		·				30
1	●		●	●		·			●	31
2	●		●	●		·		●		32
3			●	●		·		●	●	33
4	●		●	●		·	●			34
5			●	●		·	●		●	35
6			●	●		·	●	●		36
7	●		●	●		·	●	●	●	37
8	●		●	●	●	·				38
9			●	●	●	·			●	39
:			●	●	●	·		●		3A
;	●		●	●	●	·		●	●	3B
<			●	●	●	·	●			3C
=	●		●	●	●	·	●		●	3D
>	●		●	●	●	·	●	●		3E
?			●	●	●	·	●	●	●	3F

CHARACTER I/D	8	7	6	5	4	·	3	2	1	Internal value hex
@	●	●				·				40
A		●				·			●	41
B		●				·		●		42
C	●	●				·		●	●	43
D		●				·	●			44
E	●	●				·	●		●	45
F	●	●				·	●	●		46
G		●				·	●	●	●	47
H		●			●	·				48
I	●	●			●	·			●	49
J	●	●			●	·		●		4A
K		●			●	·		●	●	4B
L	●	●			●	·	●			4C
M		●			●	·	●		●	4D
N		●			●	·	●	●		4E
O	●	●			●	·	●	●	●	4F
P		●		●		·				50
Q	●	●		●		·			●	51
R	●	●		●		·		●		52
S		●		●		·		●	●	53
T	●	●		●		·	●			54
U		●		●		·	●		●	55
V		●		●		·	●	●		56
W	●	●		●		·	●	●	●	57
X	●	●		●	●	·				58
Y		●		●	●	·			●	59
Z		●		●	●	·		●		5A
[	●	●		●	●	·		●	●	5B
\		●		●	●	·	●			5C
]	●	●		●	●	·	●		●	5D
^	●	●		●	●	·	●	●		5E
_		●		●	●	·	●	●	●	5F

NUL						·				80	*1
SOH	●					·			●	81	
STX	●					·		●		82	
ETX						·		●	●	83	
EOT	●					·	●			84	
ENQ						·	●		●	85	
ACK						·	●	●		86	
BEL	●					·	●	●	●	87	
BS	●				●	·				88	
HT					●	·			●	89	*2
LF					●	·		●		8A	*3
VT	●				●	·		●	●	8B	
FF					●	·	●			8C	
CR	●				●	·	●		●	8D	*4
SO	●				●	·	●	●		FE	*5
SI					●	·	●	●	●	FE	*5
DLE	●			●		·				90	
DC1				●		·			●	91	
DC2				●		·		●		92	
DC3	●			●		·		●	●	93	
DC4				●		·	●			94	
NAK	●			●		·	●		●	95	
SYN	●			●		·	●	●		96	*6
ETB				●		·	●	●	●	97	
CAN				●	●	·				98	*7
EM	●			●	●	·			●	99	*8
SS	●			●	●	·		●		9A	
ESC				●	●	·		●	●	9B	*9
FS	●			●	●	·	●			9C	*10
GS				●	●	·	●		●	9D	
RS				●	●	·	●	●		9E	*11
US	●			●	●	·	●	●	●	9F	*12

`			●	●		·				60	
a	●		●	●		·			●	61	
b	●		●	●		·		●		62	
c			●	●		·		●	●	63	
d	●		●	●		·	●			64	
e			●	●		·	●		●	65	
f			●	●		·	●	●		66	
g	●		●	●		·	●	●	●	67	
h	●		●	●	●	·				68	
i			●	●	●	·			●	69	
j			●	●	●	·		●		6A	
k	●		●	●	●	·		●	●	6B	
l			●	●	●	·	●			6C	
m	●		●	●	●	·	●		●	6D	
n	●		●	●	●	·	●	●		6E	
o			●	●	●	·	●	●	●	6F	
p	●		●	●	●	·				70	
q			●	●	●	·			●	71	
r			●	●	●	·		●		72	
s	●		●	●	●	·		●	●	73	
t			●	●	●	·	●			74	
u	●		●	●	●	·	●		●	75	
v	●		●	●	●	·	●	●		76	
w			●	●	●	·	●	●	●	77	
x			●	●	●	●	·			78	
y	●		●	●	●	●	·		●	79	
z	●		●	●	●	●	·			7A	
{			●	●	●	●	·		●	7B	
|	●		●	●	●	●	·	●		7C	
}			●	●	●	●	·	●		7D	
¬			●	●	●	●	·	●	●	7E	
DEL	●		●	●	●	●	·	●	●	FF	*13

*1 - Null
*2 - Horizontal Tabulation (Tab)
*3 - Line Feed (New Line)
*4 - Carriage Return
*5 - Invalid (Undefined)
*6 - Synchronize
*7 - Void Data
*8 - End-of-Media
*9 - Escape
*10 - End-of-File
*11 - End-of-Record
*12 - End-of-Field
*13 - Delete

EQUIPMENT SETUP

The accounting machine used in punching paper tape must have at least a 16 inch carriage, a typewriter with alpha character capabilities, and must be wired to a punched paper tape recorder. NCR class 31, 32, 33, 35 and 36 accounting machines can be used to punch paper tape.

Punched paper tape codes are entered through the typewriter keyboard exclusively when punching NEAT/3 source records in paper tape. A journal containing a visual record of entries is created as a by-product of punching source records.

Standard accounting machine wiring (as discussed in the TECHNIQUES AND PROCEDURES REFERENCE MANUAL, GENERAL, "Standard Accounting Machine Wiring") is assumed with the following exceptions. In the operating instructions included in this publication, the following keys are assumed to be wired to perform a specific function in addition to their normal function. If the key/function relationship is different on the customer's accounting machine, the operating instructions should be modified to reflect his system.

- The typewriter TAB key and the R2 key are wired to emit the hexadecimal code for the unit separator control character (9F) in addition to their normal functions.
- The R1 key is wired to turn OFF the Alpha Typing switch, yet emit no code, in addition to its normal function.
- The R3 key is wired to emit the hexadecimal code for the record separator control character (9E) in addition to its normal function.

NOTE

The Alpha Typing light must be ON before the keys will cause the correct character to be punched.

Form Bars Used to Punch NEAT/3 Source Records

- Front Form Bar

STOP	STOP CONSTRUCTION	LOCATION
1 (Tape Feed)	3 block and 4 insert	5.0 - 5.3 left of 0
2	Type	4.3 - 4.4 left of 0
3	Type	3.5 - 3.6 left of 0
4 (Home Position)	3 block and 4 insert	3.0 - 3.3 left of 0
5	Type	1.9 - 2.0 left of 0
6	Type	0.8 - 0.9 left of 0
7	3 block and 2 insert	1.6 - 1.9 right of 0
8	Type	2.3 - 2.4 right of 0
9	Type	5.1 - 5.2 right of 0
10	Type	5.7 - 5.8 right of 0

NOTE

Depending on the type of Alpha Coupler and wiring present on the accounting machine, refer to the appropriate section of the Accounting Machine Product Information manual for instructions for turning Alpha Typing ON.

- If the accounting machine is equipped with the T feature, refer to the programming instructions on form F-3303.
- If the accounting machine is equipped with WA wiring and a 434-2 Alpha Coupler, refer to the wiring principles discussed in section 8 of the 395 Product Information manual.
- If the accounting machine is equipped with WLA wiring, refer to section 35-200 of the Accounting Machine Product Information manual.
- If the type of coupler and/or wiring is not known, refer to the Common Carriage section of the Accounting Machine Product Information manual.

Column Typing Guide

The accuracy of the source records and the ease of reference for punching them depends, in a large measure, on the construction and use of the column typing guide. This paper guide fits under the clear plastic shield on the face of the front form bar cover. When properly positioned, this guide shows the column into which the next data character will be entered.

- Types of Scales

  There are three scales on the column typing guide to indicate the three different types of source line entries.

  - D Scale

    Use this scale exclusively when entering columns 1 through 29 of the data layout worksheet.

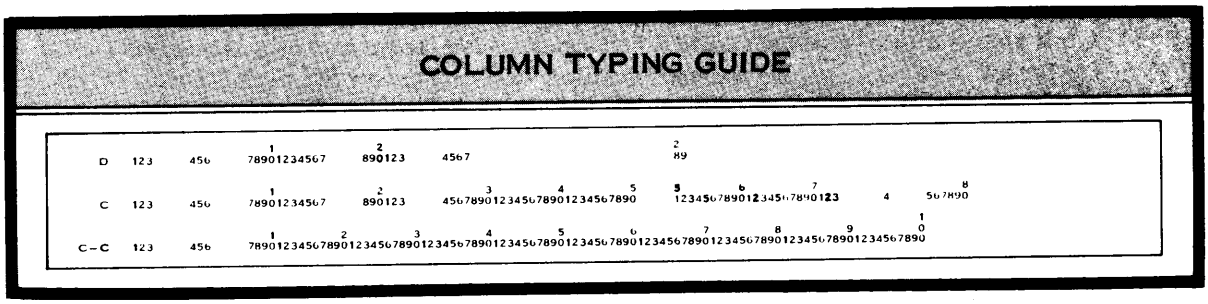
  - C Scale

    Use this scale when entering all columns of the coding worksheet, as well as columns 30 through 80 of the data layout worksheet.

  - C-C Scale

    Use this scale when entering source lines from all other worksheets.

- Constructing the Column Typing Guide



1. Place the paper on which the guide is to be printed into the carriage. This paper must be at least 15 inches wide.
2. Install the front form bar, and position the carriage on stop 4 (the home position).
3. Follow the illustration to create the guide. Press the type TAB key whenever there is a separation between groups of characters.
4. Allow enough room between scales to permit the entry of the appropriate tens digit above each zero (see the column typing guide illustration). Space all three scales so that they can easily be seen within the area provided on the front form bar cover.
5. Place the completed guide in the front form bar cover. Position it so that column 1 on the scale is aligned with the type indicator when the carriage is in the home position (stop 4).

Final Preparation

It is important that the final stage of equipment setup be handled in a systematic manner. These steps should be followed:

1. Mount the appropriate front and (if needed) rear form bars.
2. Check the paper tape supply in the recorder.
3. Empty the chad drawer in the paper tape recorder, and empty it at intervals during the punching process.
4. Set the spacing levers on R1, R2 and R3 to feed the journal.
5. Load journal paper for a hard copy record.
6. Turn on all equipment, including the paper tape recorder.
7. Move the carriage to the tape feed position (stop 1). To do this, press the CARR RET key all the way down (causing the carriage to return to the extreme right); then press the TAB key on the upper keyboard to position the carriage at stop 1.
8. Press the paper tape feed button until approximately five feet of leader has been punched. Thread the leader onto the takeup reel.
9. Press the middle motor bar to tabulate the carriage to the home position (stop 4). The type location indicator is now pointing to column 1 on the column typing guide. The Alpha Punch indicator must be ON when the carriage stops in the home position.
10. Punch the paper tape source lines.

PUNCHING PROCEDURES

Splicing

Because paper tape is a continuous media, splicing is occasionally required. Splicing can be expedited by punching a series of ignore codes (such as tape feed) on a reel of source tape at each of the following locations:

1. After the entries on the last sheet of the set of compiler specification worksheets.
2. Before the END$ entry.
3. Before the End-of-Media (EM) control code.
4. Before the EXITTO entry.

Zero Suppression

When a field is terminated with an end-of-field character (US), punch only the significant characters (if any) in that field. SPUR right-justifies all numeric fields and zero-fills to the left. SPUR also left-justifies all alpha fields and space-fills to the right. If there are no significant characters within the field, SPUR sets the entire field to spaces.

Zero suppression cannot be used when entering records in the column-for-column format.

Paper Tape Format Code

The paper tape format code must be the first entry made each time the source document type changes. This code is preprinted on each source document.

For example, when using the column-for-column format, enter /99 as the format code.

First and Last Source Lines

The following conventions apply to all fields in which no data is to be entered.

- Records Using the Column-for-Column Format (End-of-Field not used)

  If there are no succeeding fields of significant data within the record, terminate the record with a record separator character (RS).

  If there are succeeding fields of significant data within the record, punch each column of the current field with a space character to maintain the column-for-column format.

- Records Using End-of-Field (Unit Separator - US) Termination of Data Fields

  If there are no succeeding fields of significant data within the record, terminate the record with a record separator character.

  If there are succeeding fields of significant data within the record, enter a unit separator character in the current field and each succeeding field that has no data.

Early Termination of Records

Whenever the last significant data character has been entered in any record, terminate the record with a record separator (RS) character. Any succeeding field not entered is assumed to be spaces.

Double-Control Code at the End of a Record

Some paper tape readers are controlled by special control codes punched in the paper tape. To accommodate this situation, use a double-control code (record separator and carriage return symbols) to terminate a source record. SPUR ignores the carriage return code and performs the usual record termination when the record separator code is detected.

If a double code is necessary, the user must punch the record separator first and follow it be a carriage return. SPUR ignores the carriage return only if it immediately follows a record separator and if no data characters are transmitted with it.

End-of-Media Procedure

If the tape supply is running low and punching has not been completed, use the following procedure:

1. Punch record separator (RS) code after the last source record on the current reel.
2. Punch a series of ignore characters (NUL) to facilitate possible splicing.
3. Punch the EM character.
4. Punch about five feet of tape with a series of ignore codes.
5. Load a new reel of paper tape.
6. Punch about five feet of tape with a series of ignore codes.
7. Punch the label (if being used) and follow it with a series of ignore codes.
8. Punch the paper tape format code. The first source record of a new reel of paper tape must be preceded by the appropriate paper tape format code.
9. Continue punching the source lines.

Control Code Patterns for Source Documents

Each source document has a unique control code pattern which will normally be used when data is punched from that specific document. The pattern is used by the compiler to establish the location and limits of all fields designated for a particular document.

Punch a unit separator code (except when using the column-for-column format) whenever the symbol ↛ appears on the source document. If the symbol is above a column, the unit separator code is punched prior to punching the column.

Punch a record separator code whenever the symbol ≡ appears on the source document and at the end of each source record.

Punching Source Records

The punching of paper tape requires a systematic approach. The following descriptions assume the use of the column typing guide.

- Optional Label

  To facilitate the referencing of various reels of paper tape, the user has the option of punching an information label ahead of the source program. If the source program extends to more than one reel of paper tape, label each reel with the same program name and number the reels sequentially. The label has no paper tape format code. Following the punching of a leader (approximately five feet), prepare the label as follows:

  1. Make sure the carriage is in the home position, with the type location indicator pointing to column 1 on the column typing guide.
  2. Type the language name (e.g., NEAT/3).
  3. Press the R2 key. This causes a unit separator code to be punched, returns the carriage to the home position, and spaces the journal.
  4. Type the user's program name (10 characters maximum), if one is used, and press R2. If no program name is used, press the typewriter TAB key.
  5. Type a 3-character numeric reel count (001-099).
  6. Press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.
  7. Continue by entering the source records. Make sure the format code precedes the first record of each reel.

- Coding Worksheets

  Coding worksheet entries are made using the C scale on the column typing guide, unless the operands and/or comment entries extend to an additional line. In this event, it is easier to use the column-for-column format on the C-C scale.

  1. Make sure the carriage is in the home position with the type location indicator pointing to column 1 on the column typing guide.
  2. If this is the first of a series of coding sheets, type the 3-character paper tape format code.
  3. Press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.

  NOTE

  Ignore steps 2 and 3 if the paper tape format code has already been entered for a previous coding sheet within the same series of like entries.

4. With type location indicator pointing to column 1, type a 3-digit page number; press the type TAB key. If the page number of this source line is the same as that of the previous source line, press the type TAB key without entering a page number.
5. With the type location indicator pointing to column 4, type a 3-digit line number.
6. Press the type TAB key.
7. Type the balance of the source line using the C scale. After entering the last character of each field (with the exception of the last field of the source line), press the type TAB key. This causes a record separator code to be punched and tabs the carriage to the next field. After entering the last field of the source line, press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.
8. Continue entering coding source lines in the same manner.

- Data Layout Worksheets

When punching source lines from data layout worksheets, use both the D and the C scales. As shown in the column typing guide illustration, the D scale extends only to column 29. Press R2 and punch the remaining columns using the C scale. By using both scales, one source line appears as two lines on the journal.

In some cases, the value or picture field may extend beyond the column limits (column 50) on the C scale. Therefore, after punching in column 50, the column location can be maintained only by referencing the entry on the data layout worksheet.

1. Make sure the carriage is in the home position, with the type location indicator pointing to column 1 on the column typing guide.
2. If this is the first of a series of data layout worksheet entries, type the 3-character paper tape format code.
3. Press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.

NOTE

Ignore steps 2 and 3 if the paper tape format code has already been entered for a previous data layout worksheet within the same series of like entries.

4. With the type location indicator pointing to column 1, type a 3-digit page number; press the type TAB key. If the page number of this source line is the same as that of the previous source line, press the type TAB key without entering a page number.
5. With the type location indicator pointing to column 4, type a 3-digit line number.
6. Press the type TAB key.

7. Type the succeeding characters through column 29 using the D scale. After entering the last character of each field (with the exception of column 29), press the type TAB key. This causes a unit separator code to be punched and tabs the carriage to the next field. After typing the character in column 29, press the R2 key. This causes a unit separator code to be punched, returns the carriage to column 30 on the C scale, and spaces the journal.
8. Type the balance of the source line using the C scale. After entering the last character of each field (with the exception of the last field of the source line), press the type TAB key. After entering the last field of the source line, press the R3 key. This causes an RS code to be punched, returns the carriage to the home position, and spaces the journal.
9. Continue entering data source lines in the same manner.

LISTING FORMAT FOR VERTICALLY FORMATTED WORKSHEETS (Compiler Control, File Spec., Etc.)

When using the listing format, use the column typing guide for home position referencing.

NEAT/3 source lines may be punched in this format as well as vertically formatted worksheets as long as there are not more than 39 characters in any field. If the source document exceeds this 39 character limit, use the column-for-column format.

1. Turn the recorder OFF. Lock the R2 key to the type tab function by moving the Carriage Position Control lever toward the front of the machine.
2. Make sure the carriage is in the home position, with the type location indicator pointing to column 1 on the column typing guide.
3. Turn the recorder ON.
4. Type the 3-character paper tape format code.
5. Press the R3 key. This causes an RS code to be punched, returns the carriage to the home position, and spaces the journal.
6. Type the balance of the worksheet.

   Whenever the symbol ↘ appears, press the R2 key. The carriage returns to the home position and immediately tabs to a typing stop; a US code is punched, and the journal spaces.

   Whenever the symbol ≡ appears, press the R3 key. The carriage returns to the home position, but does not tab to a typing stop; an RS code is punched, and the journal spaces.

7. When you are finished using the listing format, turn off the recorder and disengage the Carriage Position Control Slide by moving it toward the back of the machine.
8. Turn the recorder ON before punching any other format.

Column-for-Column Format

When using the column-for-column format, punch columns 7 through 100 (or less) using the C-C scale. Punch one column at a time, including spaces, without any intervening US codes. Any or all NEAT/3 source lines may be punched using this format. However, the records being punched in this format will usually be from source documents other than data layout and coding worksheets.

1. Make sure the carriage is in the home position, with the type location indicator pointing to column 1 on the column typing guide.
2. If this is the first of a series of column-for-column entries, type the 3-digit paper tape format code /99.
3. Press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.

NOTE

Ignore the steps 2 and 3 if the paper tape format code has already been entered for a previous entry within the same series of like entries.

4. With the type location indicator pointing to column 1, type a 3-digit page number, and then press the type TAB key. If the page number of this source line is the same as that of the previous source line, press the type TAB key without entering a page number.
5. With the type location indicator pointing to column 4, type a 3-digit line number.
6. Press the type TAB key.
7. With the type location indicator pointing to column 7, type columns 7 through 100 (or less), one column at a time including spaces, without any intervening unit separator codes.
8. After entering the last significant character, press the R3 key. This causes a record separator code to be punched, returns the carriage to the home position, and spaces the journal.
9. Continue entering the source lines in the same manner.

ERROR CORRECTION PROCEDURES

NEAT/3 source statements must be processed by SPUR, which permits the user to take advantage of SPUR's error correction procedures.

- If the error is detected before the record separator character is punched, enter the void record code (CAN) followed by a record separator code; then repunch the control statement.

- If the error is detected after the record separator character has been punched, repunch the record correctly, using the same page and line number as the incorrect record. When two or more records have the same page and line number, SPUR retains only the last record received.

- Errors made when punching Compiler Control statements are not omitted by SPUR; therefore, a correct Compiler Control statement must be spliced in place of the erroneous record.

MONITOR INPUT CONVENTIONS

The procedures for punching control instructions to be input directly to Monitor are the same as those described under "PUNCHING PROCEDURES", with the following exceptions:

- Monitor will not accept label entries; do not punch them.
- Because Monitor has no sort capabilities, punch the control instructions in the desired input sequence.
- To correct an error before punching the record separator code, enter the void record code followed by the record separator code. Repunch the corrected record.
- To correct an error after punching the record separator code, repunch the entire control instruction and splice the new instruction in place of the original.

Punch approximately twelve null or delete codes after each record separator code, to enable visual location of the end of each record and to allow for splicing.

The format code should be punched each time it changes, provided that all Monitor master control items are punched from coding sheets (format 00) as specified. If the Monitor master control items are punched with other than format 00, the format code should be punched ahead of each line.

Each series of control items punched on one piece of paper tape should end with a FINISH or an INTYPE control item, to bring Monitor to an orderly display at the end of each piece of paper tape. Monitor does not recognize the end-of-media control code (EM).

OVERVIEW OF THE COMPILATION PROCESS

INTRODUCTION

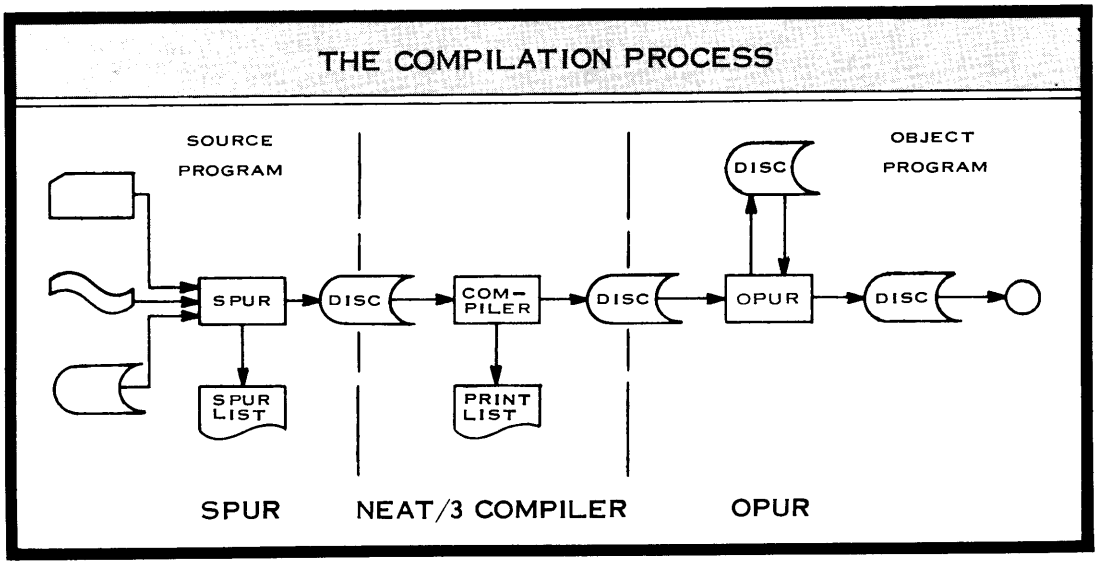
The compilation process is the procedure by which a source program is translated and assembled into an object program. More specifically, this process involves the precompilation of a source statement file from either punched cards or punched paper tape and generating a source program file on disc. The precompilation phase is required because the compiler only accepts input from disc. During the compilation phase, the compiler translates the source program file and generates an object program file on the same disc pack. The object program file is then copied to another disc pack to create the final object program. Both the precompilation and compilation phases are capable of producing complete program listings as required.

To accomplish this processing for the NEAT/3 language requires the use of three separate but related software systems:

- Source Program Utility Routines (SPUR)
- NEAT/3 Compiler
- Object Program Utility Routines (OPUR)

Each of these systems is discussed briefly in this publication. For a complete description of SPUR and OPUR, see UTILITY ROUTINES REFERENCE MANUAL, PROGRAM ASSOCIATED, "Source Program Utility Routines - SPUR" and "NCR Century OPUR."

The relationship between these three systems during the compilation process is shown in the following illustration.



Types of Compilation

The programmer may specify any of four types of compilation for the NEAT/3 Compiler.

- A full compilation (type F) is always required for the initial compilation, and whenever source program changes occur in the main program (with the exception explained for partial compilations).
- An overlay compilation (type O) is always a recompilation and is specified whenever source program changes occur in one or more of the program overlays. Only those overlays receiving changes are recompiled. In addition, overlay compilation is used in conjunction with full and partial compilations to compile large programs. The use of overlay compilation is discussed in the TECHNIQUES AND PROCEDURE REFERENCE MANUAL, COMPILER RELATED, "Independent Overlay Compilation" and "Compiling Programs Exceeding 12,000 Source Statements."
- A partial compilation (type P) is always a recompilation and is applicable only to large programs exceeding approximately 12,000 source statements.

NOTE

Large programs must be divided into two sections to be compiled. The initial compilation requires a full compilation followed by an overlay compilation.

A partial compilation is specified whenever changes occur in the main program and overlays contained in the first section of large programs and is followed by an overlay compilation for changes occurring in the second section. Overlays of large programs may be compiled independently when main program changes are not present. The use of partial compilation is discussed in the TECHNIQUES AND PROCEDURES REFERENCE MANUAL, COMPILER RELATED, "Compiling Programs Exceeding 12,000 Source Statements."

- A module compilation (type M) is specified for source programs that have been organized into modules. This type of compilation is to be discussed in a future publication.

The table on the following page illustrates the relationship between the various types of compilation available for the NEAT/3 Compiler.

RELATIONSHIP BETWEEN NEAT/3 COMPILATIONS		
TYPE OF COMPILATION	PROGRAMS NOT EXCEEDING 12,000 SOURCE STATEMENTS	LARGER PROGRAMS WITH OVERLAYS AND/OR MODULES EXCEEDING 12,000 SOURCE STATEMENTS
Initial Compilation	Full Compilation	Full Compilation Overlay Compilation
Recompilation Entire Program	Full Compilation	Partial Compilation Overlay Compilation
Recompilation Overlays Only	Overlay Compilation	Overlay Compilation
Recompilation Modules Only	Module Compilation	Module Compilation

THE COMPILATION PROCESS

The compilation process, which involves the translation of a source program into an object program, is essentially the same for all types of compilation. The general functions performed by the three basic software systems during the compilation are discussed in the following sections.

Source Program Utility Routines (SPUR)

The source program is first processed by SPUR, which performs certain functions required for the NEAT/3, COBOL, and FORTRAN Compilers. These compilers accept input from disc only. SPUR converts the source program input from punched cards, punched paper tape or disc and arranges this input on another disc for use by one of the compilers. Other input media, such as magnetic tape from the NCR 736 Magnetic Tape Encoder, is not directly acceptable to SPUR. This media must be copied by a utility routine to a disc prior to input to SPUR.

SPUR performs the following functions for the NEAT/3 Compiler:

- Reads source program from punched cards, punched paper tape, or disc.
- Sorts the source program into page and line number sequence if requested on the Compiler Control Statement.
- Performs COPY operations.
- Performs OMIT operations.
- Retains only the last source record read when two or more source records have duplicate page and line numbers.
- Merges corrected source records with the Recompilation Master (previously compiled source program).
- Reassigns page and line numbers to the source program as specified on the Compiler Control Statement.

- Prints a listing of all records upon which some action was taken or an error noted. Opposite the source record is printed a notation which describes such actions/errors as:

  - Copied source records.
  - Deleted and retained source records.
  - Added source records (for a recompilation).
  - Untranslatable characters.
  - No COPY - invalid statement.

Sort

On the initial compilation run the SPUR program reads each source record of the input data. If a sort is requested on the Compiler Control Statement, the page and line numbers are sequence-checked and any out-of-sequence records are sorted and merged later with the in-sequence records. If no sort is requested, the source records will be renumbered automatically. Source records must be sorted when doing a recompilation.

COPY

The COPY operation may be used in the source program to indicate to SPUR that certain source records in another source program, residing on the source-object disc, are to be copied into the source program that is being processed. For example, file specifications and related data definitions may be copied into the source program from another program.

OMIT

The OMIT operation is used in the source program to remove incorrect source records during the recompilation run.

The SPUR Listing

The listing printed by SPUR indicates all the actions taken and all the format errors detected during the SPUR run. (To make any corrections in the source program, the programmer must rerun SPUR before presenting the source program to the NEAT/3 Compiler.)

NEAT/3 Compiler

The next step in the compilation process belongs to the NEAT/3 Compiler program. It processes the source program, which has been placed on the disc by SPUR, and creates a complete object program along with a printed listing of the program.

The source program to be compiled may be any of the following types:

- A complete program including overlays.
- A program to which overlays will be added later.
- An overlay to add to an existing program.
- An overlay to replace an overlay in an existing object program.

In addition to compiling the object program, the NEAT/3 Compiler generates a complete program listing. This listing shows all the source lines in their proper order and identifies incorrect source lines with error comments.

At the programmer's option, the NEAT/3 Compiler also produces a cross-reference listing of all the reference names used in a program.

Object Program Utility Routines (OPUR)

The Object Program Utility Routines (OPUR) perform the following functions:

- Copy the object program from the source-object disc pack to another disc pack.
- Insert an overlay in an existing program.
- Replace an overlay in an existing program with a later version overlay.

RELATED PUBLICATIONS

The broad scope of the compilation process is applicable to both user programs and system control strings, and is dependent upon several utility routine functions and procedural techniques. The many related publications are therefore contained in several different manuals. The following is a general listing by manual of specific publications concerning the compilation process.

- NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 3,
  Compiler Control Statements
  COPY Control Instructions
  OMIT Control Instructions
  Overlay Control Instructions
  ENTRY Control Instruction
  SECTION Control Instructions
  RENAME Control Instruction
  END Control Instruction
  SETPL Control Instruction

- NEAT/3 REFERENCE MANUAL, COMPILATION PROCESS, tab 1,
  Organization Of A Source Program
  Punching Punch Card Source Program Records
  Punching Paper Tape Source Program Records

- NEAT/3 REFERENCE MANUAL, COMPILATION PROCESS, tab 2,
  Overview Of The Compilation Process
  Related Software
  Creating A Production Program

- NEAT/3 REFERENCE MANUAL, COMPILATION PROCESS, tab 3,
  Compiler Output Printing
  Error Comment Directory

- NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1,
  Peripheral Type Codes
  Data Format Codes
  Symbolic Unit Designators

- NEAT/3 REFERENCE MANUAL, APPENDIX, tab 2,
  Compiler Control Instructions
  Source Program Organization

- OPERATING SYSTEM REFERENCE MANUAL, MONITOR, Description & Functions,
  General Description
  The System Control String
  The Override Control String
  Punched Media Input To Monitor

- OPERATING SYSTEM REFERENCE MANUAL, MONITOR, Control Instructions,
  HEADCS Control Instruction

- UTILITY ROUTINES REFERENCE MANUAL, PROGRAM ASSOCIATED,
  Language Directory List Routine
  Source Program Utility Routines (SPUR)
  NCR Century OPUR
  Symbolic Debug Routine
  Flowrite
  Production Debug Routine

- TECHNIQUES AND PROCEDURES REFERENCE MANUAL, COMPILER RELATED,
  Independent Overlay Compilation
  Compiling Program Exceeding 12,000 Source Statements

RELATED SOFTWARE

The software for the NCR Century Series is highly interdependent. For instance, no compiled NEAT/3 program can function without the use of the Input/Output Executive which handles all the details of data transfers between units of the system. The Input/Output Executive finds all the information needed for its operation in the compiled NEAT/3 program.

The Monitor is another major software item. It is used to call into memory all necessary programs, including the Source Program Utility Routines and the NEAT/3 Compiler. This section briefly describes those areas of the Monitor that are used in the compilation process.

MONITOR

The primary functions of the Monitor are to load programs and to automatically link system runs (programs) as predetermined by the programmer. Automatic program linking provides a streamlined operation with a minimum need for operator intervention. The programmer has complete flexibility in establishing the most efficient program sequence. He may designate programs to be run conditionally, depending on calendar dates, day of week, or other information in memory; or he may designate programs to be run unconditionally.

The Monitor is concerned with programs on disc only. It cannot directly load programs from any other media.

The Control String

At the beginning of a series of programs or after a program has been run, the Monitor takes control and reads the first or next segment of a control string to determine what to do next.

A control string is basically a sequence of Monitor control instructions and program names. The sequence in which these program names appear in the control string reflects the order in which the programs are to be run.

A typical control string is shown below.

PAGE	LINE		REFERENCE	OPERATION	OPERANDS
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 40 41 42 43 44 45 46 47 48 49 50
0 0 0	0 3 0	C	A C C T S R C V	H E A D C S	
0 0 0	0 6 0	C		N E X T D O	R U N 1
0 0 0	0 9 0	C		N E X T D O	R U N 2
0 0 0	1 2 0	C		N E X T D O	R U N 3
0 0 0	1 5 0	C		N E X T D O	R U N 4
0 0 0	1 8 0	C		N E X T D O	R U N 5
0 0 0	2 1 0	C		N E X T D O	R U N 6
0 0 0	2 4 0	C		N E X T D O	R U N 7
0 0 0	2 7 0	C		F I N I S H	

The name of this control string is ACCTSRCV (Accounts Receivable). Its function is to automatically link seven individual programs to completely perform an accounts receivable task.

In this example the individual program names reflect the sequence in which the programs are executed. (Normally the function, rather than a sequence, is reflected by a program name.)

The coding of control string items is similar to the coding of NEAT/3 source lines.

Control strings may be stored on punched cards, punched paper tape, or on a disc. The programmer may place control strings on any disc by using the Source Program Utility Routines (SPUR).

Monitor Control Instructions

The following are some Monitor control instructions that the programmer may use in the process of compiling a source program. A more detailed explanation of all the Monitor Control Instructions is contained in a separate publication on the Monitor.

- DATE

PAGE	LINE		REFERENCE	OPERATION	OPERANDS
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 40 41 42 43 44 45 46 47 48 49 50
X X X	X X X	C		D A T E	2 5 / 0 6 / 6 8 / T U E , 2 5 / 0 6 / 6 8 / T U E

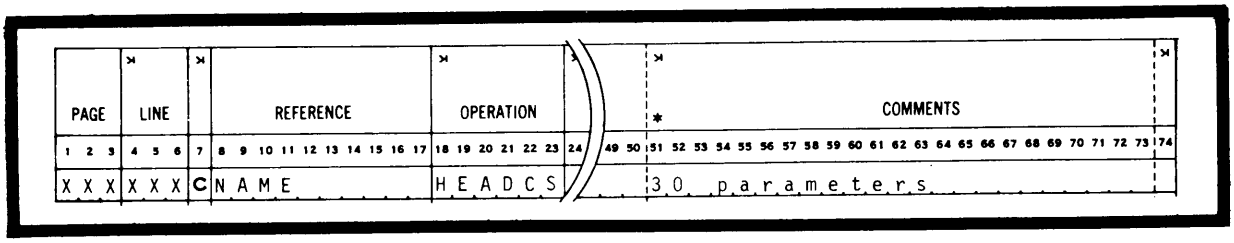
The DATE control instruction causes the Monitor to store the virtual and actual dates in the reserved memory area. The DATE control instruction also sets the 30 parameter characters in the reserved memory area to zero.

The virtual date is the date for which the running of a program is scheduled; the actual date is today's calendar date. For instance, if a string of programs is scheduled to be run on a Monday, but is actually run on a Tuesday, the virtual date is Monday's calendar date and the actual date is Tuesday's calendar date.

The actual date can only be stored in the reserved memory by the COT Start Routine (part of Monitor). If the DATE instruction is read by the Monitor at any other time, only the virtual date (positions 24-31) can be changes.

- Positions 1-3 contain the page number.
- Positions 4-6 contain the line number.
- Position 7 contains C.
- Positions 18-21 contain DATE to indicate a date control.
- Positions 24-47 contain the virtual day, month, year, and day of week by the actual day, month, year, and day of week.

- HEADCS

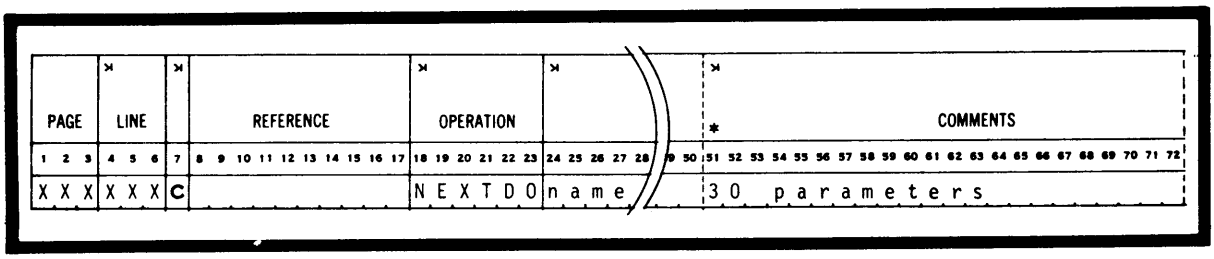


PAGE	LINE		REFERENCE	OPERATION		COMMENTS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18 19 20 21 22 23	24	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
X X X	X X X	C	N A M E	H E A D C S		3 0 p a r a m e t e r s

The HEADCS control instruction denotes the beginning of a control string. The name in the reference column of the HEADCS instruction becomes the name of the control string. When the Monitor encounters this control instruction, it moves the 30 parameter characters (see explanation below), into the reserved memory area, and then reads in the next control item in the string.

- Positions 1-3 contain the page number.
- Positions 4-6 contain the line number.
- Position 7 contains C.
- Positions 8-15 contain the name that the control string assumes when it is placed on disc by the Source Program Utility Routines. Positions 16 and 17 contain the version number of the control string, if applicable.
- Positions 18-23 contain HEADCS to indicate that this is the header for a control string.
- Positions 51-80 contain 20 parameter characters which permit a wide range of options. These options are not needed for a simple compilation of a source program. All the options are fully explained in a separate publication dealing with the Monitor.

- NEXTDO



PAGE	LINE		REFERENCE	OPERATION		* COMMENTS
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 ... 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
X X X	X X X	C		N E X T D O	n a m e	3 0 p a r a m e t e r s

The Monitor control instruction NEXTDO calls a program into memory. The operands column contains either a program name or the name of a control string.

If the name in the operands column is the name of a program, the Monitor calls this program into memory.

If the name in the operands column is the name of a control string, the Monitor reads the first NEXTDO instruction in this control string and calls the program named in this NEXTDO instruction into memory. After this program has been run, the Monitor sequentially reads the other NEXTDO instructions in the control string and calls the named programs into memory until all the programs named in the control string have been run.

The Monitor searches the program directories on both the current and the alternate system discs to locate the latest version of the program named by the NEXTDO instruction.

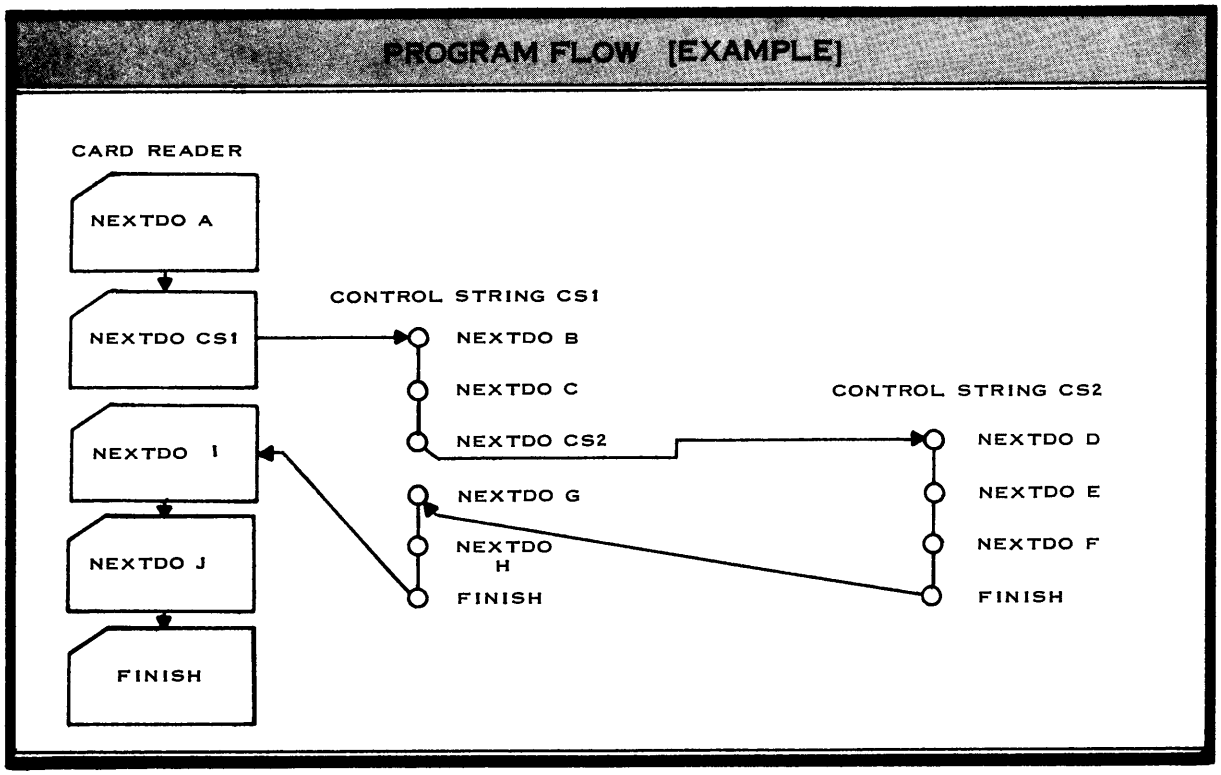
- Positions 1-3 contain the page number.
- Positions 4-6 contain the line number.
- Position 7 contains C.
- Positions 18-23 contain NEXTDO to indicate that a new program is to be called into memory.
- Positions 24-50 contain the name of the program or control string to be run.
- Positions 51-80 contain 30 characters which permit certain options.

For full details, refer to the publication describing the Monitor.

- FINISH

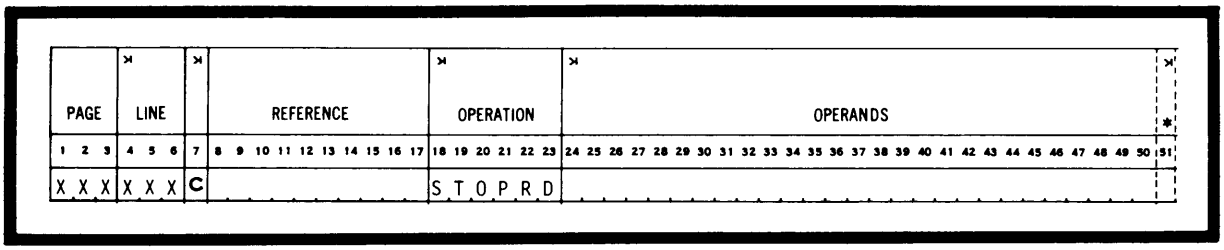
PAGE	LINE		REFERENCE	OPERATION	OPERANDS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18 19 20 21 22 23	24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
X X X	X X X	C		F I N I S H	

The FINISH instruction indicates the end of a control string. When the Monitor reads this instruction it links back to the source from which the present control string was accessed and reads the next item from that source. The source is usually the integrated punched card or paper tape reader, but it may also be another control string. If the Monitor reads a FINISH instruction in the punched card or paper tape reader, the processor halts.



- Positions 1-3 contain the page number.
- Positions 4-6 contain the line number.
- Position 7 contains C.
- Positions 18-23 contain FINISH which indicates the end of a control string.

- STOPRD



PAGE	LINE		REFERENCE	OPERATION	OPERANDS	
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 40 41 42 43 44 45 46 47 48 49 50	51
X X X	X X X	C		S T O P R D		

If the Monitor reads STOPRD from a punched card or from punched paper tape, the next record (on punched card or paper tape) is either data input to the program presently being loaded by the Monitor or a control instruction for the program that is to be called next.

- Positions 1-3 contain the page number.
- Positions 4-6 contain the line number.
- Position 7 contains C.
- Positions 18-23 contain STOPRD.

Placing a Control String on Disc

The following is a hypothetical example that illustrates how a control string may be placed on a disc and how the programs within that control string may be subsequently run. This example has been chosen to give the reader a better understanding of some of the Monitor functions. Normally, the operator would place the control string on a disc at one time and run the programs specified by this control string at another time. Also, the operator would use the Monitor Boot only after a reset from the console or following a shut-down of the system after power has been turned on again. The DATE instruction is needed only if a date change is required.

The following sequence of punched card or paper tape records may be used to place a control string on a disc and to start the execution of the programs within that control string.

» TABS RECORD NUMBER	PAGE	» LINE	»	REFERENCE	» OPERATION	» OPERANDS
	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 40 41 42 43 44
1			C	(MONITOR	BOOT	)
2	000	060	C		DATE	28/03/69/TUE
3	000	090	C		NEXTDO	SPUR
4	000	120	C		STOPRD	
5	000	000	C	ACCTSRCV	HEADCS	N Y N
6	000	060	C		NEXTDO	RUN1
7	000	090	C		NEXTDO	RUN2
8	000	120	C		NEXTDO	RUN3
9	000	150	C		NEXTDO	RUN4
10	000	180	C		NEXTDO	RUN5
11	000	210	C		NEXTDO	RUN6
12	000	240	C		NEXTDO	RUN7
13	000	270	C		FINISH	
14	END	$	C			
15	000	000	C		EXITTO	
16	END	$	C			
17	000	150	C		NEXTDO	ACCTSRCV
18	000	180	C		STOPRD	
19	000	210	C		FINISH	

- Record No. 1

  The operator uses the console switches to read the Monitor Boot into memory. The Monitor Boot then calls the Monitor from disc into memory.

- Record No. 2

  The Monitor reads the DATE instruction and stores the virtual date in the reserved memory area and on disc.

- Record No. 3

  The Monitor reads the NEXTDO instruction which calls in SPUR.

- Record No. 4

  The Monitor reads the STOPRD instruction, stops reading records, and loads SPUR into memory.

- Record No. 5

  SPUR now reads the HEADCS instruction which indicates to SPUR that a control string is to be placed on a disc. The name to be given to this control string appears in the reference column of the HEADCS instruction. This name may be a maximum of eight characters (the remaining two are reserved for a version number that is assigned by SPUR when applicable).

  An N in position 24 specifies that this is the initial construction of a control string. A Y in position 35 of the HEADCS instruction specifies that SPUR will sequence-check the page-line numbers of the following control string item (records). If these items are found to be out of sequence, SPUR sorts them before placing them on a disc. N in position 36 specifies that the page-line numbers of the control string items are not to be changed. See the OPERATING SYSTEM REFERENCE MANUAL, MONITOR, control instructions tab, "HEADCS Control Instruction".

- Records No. 6-13

  SPUR reads these control string items and places the control string on disc. (See Record No. 5 for option to sort and renumber these records.)

- Record No. 14

  END$ indicates to SPUR that this is the end of the control string. This record does not beoome part of the control string.

- Record No. 15

  SPUR reads the control instruction EXITTO which instructs SPUR to return control to the Monitor.

- Record No. 16

  The software that handles punched card input requires that the EXITTO control instruction be followed by END$.

- Record No. 17

  The Monitor reads this NEXTDO instruction which indicates that the ACCTSRCV control string (RUN1) is to be called into memory.

- Record No. 18

  The Monitor reads this STOPRD instruction which indicates that no more records are to be read by the Monitor at this time. Control is transferred to the RUN1 program.

  When RUN1 is finished, the Monitor is called into memory, where it reads the next item from the ACCTSRCV control string on disc. Since this next item specifies NEXTDO RUN2, the Monitor reads the program for RUN2 into memory.

  This action repeats itself until all seven runs in the ACCTSRCV control string are completed.

  When the Monitor reads the FINISH instruction at the end of the ACCTSRCV control string on disc, the Monitor links back to the punched card or paper tape reader from which the ACCTSRCV was accessed.

- Record No. 19

  The Monitor reads the FINISH instruction in the punched card or paper tape reader. This FINISH instruction halts the processor.

★ ★ ★ ★

CREATING A PRODUCTION PROGRAM

This section describes the normal sequence of steps for creating a production program from a newly prepared source program. Also, the operational steps required for running a compilation on an NCR Century System are briefly explained in this section.

** FROM SOURCE PROGRAM TO PRODUCTION PROGRAM

The flowchart on the following page illustrates the necessary steps for creating a production program from a newly written source program. A short explanation of each step follows.

Desk-Check the Source Program

Desk-checking the source program is the first requirement. Before taking the program to the computer, the programmer must review the logic flow of his program to ensure that it will precisely perform the desired tasks. The programmer should also organize the various types of data to be used by his program and, if possible, he should check the source program for punching errors.

Run the Appropriate Software Program and Correct Errors

The desk-check does not usually uncover all errors. Therefore, the programmer should specify on the Compilation Control Statement that this program is to be compiled in the debug mode.

Following the SPUR run, the programmer should use the SPUR listing to correct possible format errors. SPUR should be rerun after every correction until all format errors have been eliminated from the source program.

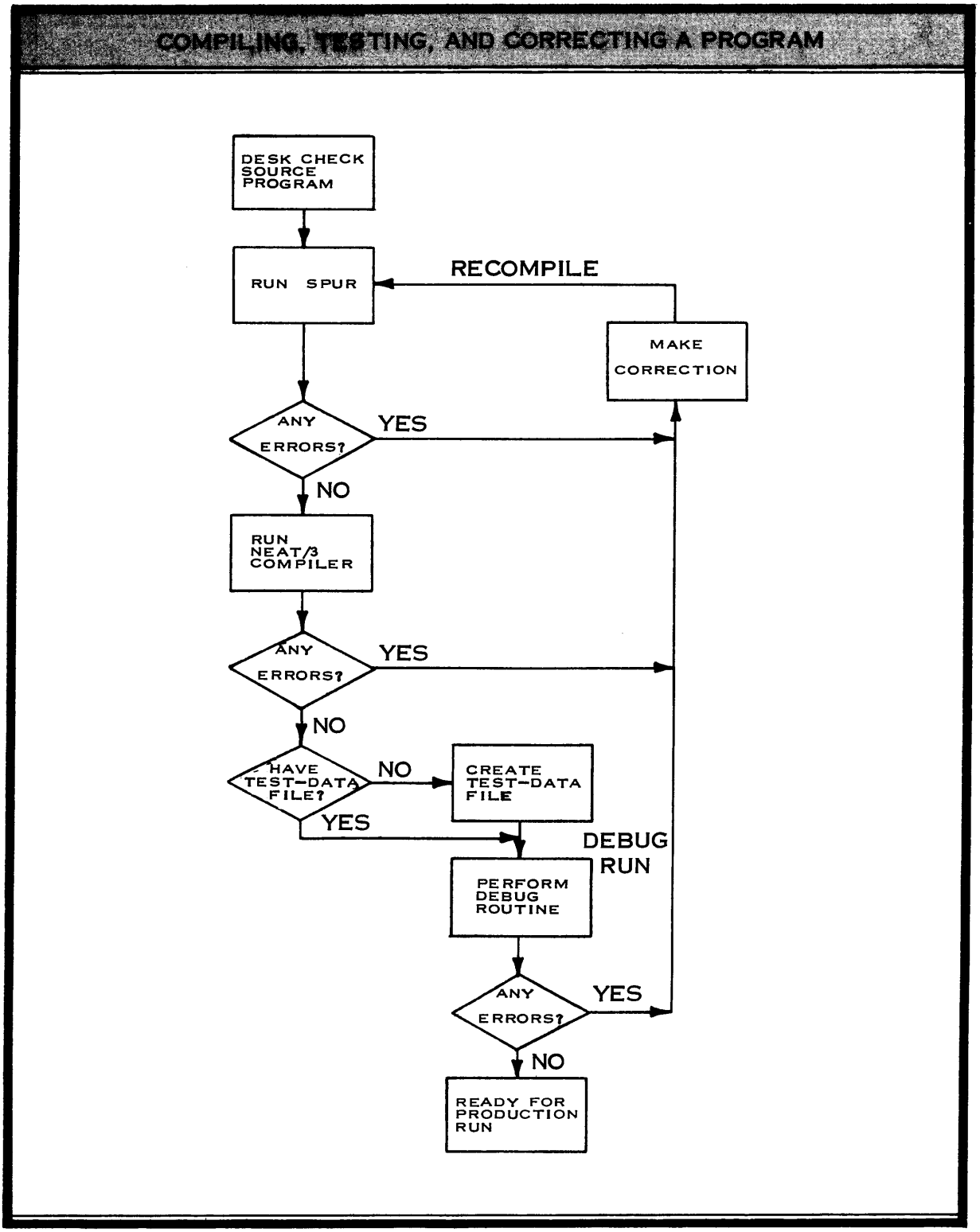
The NEAT/3 Compiler must be run next. The programmer must now check the NEAT/3 Compiler listing for error notations, make the necessary corrections in the source program, and recompile.

Recompile the Program

The programmer recompiles a program by entering only the corrected source records, the Compilation Control Statement, and the END sentinel (END$). The corrected source records must follow the order described under COMPILATION PROCESS, tab 1, "Organization of a Source Program" and must have page and line numbers corresponding to those of the incorrect source records.

The recompilation begins by rerunning the corrected source records through SPUR. This produces a corrected source program (recompilation master). If

SPUR does not detect any errors in this run, the source program may then be run through the NEAT/3 Compiler. If any errors are detected by the NEAT/3 Compiler, the recompilation process must be repeated. This process must be continued until no errors are detected by either SPUR or the NEAT/3 Compiler.



Test the Program and Debug

After correcting all the errors detected by SPUR and the NEAT/3 Compiler, the programmer is now ready to test his program on the computer. The data used for this test should not be voluminous, but should represent every type of record that may be processed, so that every path in the program may be tested.

NCR supplies software to aid the programmer in debugging his program. Two of these software aids are the Datawriter Program and the Symbolic Debug System which are fully described in the Utility Routines Reference Manual.

- Datawriter Program

  Before a program can be debugged, it must have data to process. This data is referred to in the flowchart as the Test-Data file. The Datawriter uses punched card or punched paper tape input to generate a data file on magnetic media.

- Symbolic Debug System

  As shown on the flowchart, the programmer now uses the Symbolic Debug System when running the program. At the programmer's option, the Symbolic Debug System prints information regarding program branches, records, fields, constants, and work areas whenever a specified condition exists.

After running the Debug Routine the programmer should review the printed listings and make the necessary corrections. Corrections at this point require a recompilation - again beginning with SPUR.

Run the Program

As a final step, the programmer places the debugged production program on the appropriate disc(s) for production work.

COMPILING A PROGRAM ON THE NCR CENTURY SYSTEM

Following is a description of one of several procedures that may be used for compiling an object program on the NCR Century System. In this example, the programmer uses a software control string which automatically links the necessary software routines to read NEAT/3 source statements and to compile a program.

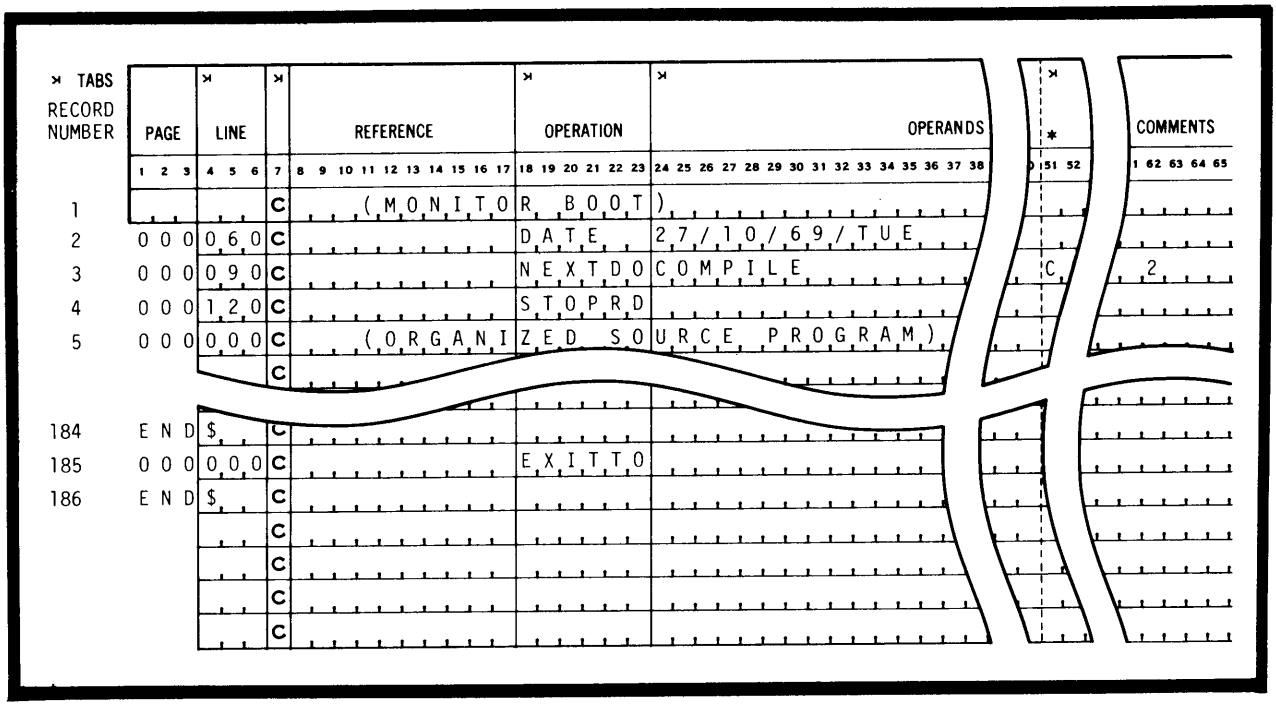
COMPILE Control String

The compiler disc contains a COMPILE control string which the programmer may use to read NEAT/3 source statements and to compile a program. The COMPILE control string first calls SPUR to read the source statements. Then the COMPILE control string calls the NEAT/3 compiler to compile the program.

Compilation Procedure

Compiling a NEAT/3 program requires two discs: a compiler disc with the necessary software, and a program disc for storing the source statements and the object program.

In the following example, the operator uses the illustrated sequence of punch card records to access the COMPILE control string and to compile the program. The use of the Monitor Boot (records 1 and 2) is only necessary following a reset from the console or following a system shut-down after the power is turned on again. The DATE instruction (record 3) is only needed if a date change is required.



- Record No. 1

  The Monitor Boot records call the Monitor into memory.

- Record No. 2

  The Monitor reads the DATE instruction and stores the virtual date in the reserved areas in memory and on disc.

- Record No. 3

  The Monitor reads the NEXTDO instruction and then accesses the COMPILE control string. The character in position 51 clears the Monitor flags. The numeric digit in position 62 of the NEXTDO instruction affects SPUR, which is called by the COMPILE control string. The 2 in position 62 indicates that SPUR will read NEAT/3 source statements from the card reader.

  The other possible codes in position 62 of the NEXTDO instruction are fully explained in the UTILITY ROUTINES reference manual.

- Record No. 4

  The Monitor reads the STOPRD instruction which indicates that no more records are to be read by the Monitor at this time. Control is transferred to the COMPILE control string which calls for SPUR.

- Records No. 5-183

  SPUR reads the organized NEAT/3 source program (records number 5 through 183) and writes it on the program disc.

- Record No. 184

  SPUR reads the END$ instruction which indicates the end of the source statements. Control is again transferred to the Monitor, which accesses the COMPILE control string and calls the NEAT/3 Compiler into memory to compile the program.

  NOTE

  After the program has been compiled, the COMPILE control string loops back to its beginning (SPUR) to compile another program unless an EXITTO instruction initiates an exit from the COMPILE control string.

- Records No. 185-186

  SPUR reads EXITTO and END$, which initiate an exit from the COMPILE control string after the program has been compiled.

As a result of the previous EXITTO instruction the COMPILE control string now calls for a printout of the program directory on the program disc. This directory includes the name of the newly compiled program. At the end of the printout routine, control is again transferred to Monitor.

- Record No. 186

  SPUR reads the END$ record which indicates that no more records are to be read at this time.

  As a result of the previous EXITTO instruction, the COMPILE control string now calls for a printout of the program directory on the program disc. This directory includes the name of the newly compiled program. At the end of the printout routine, control is again transferred to the Monitor.

- Record No. 187

  The Monitor reads the FINISH instruction which halts the processor.

* * * *

COMPILER OUTPUT PRINTING

GENERAL DESCRIPTION

A final function of compiler output is the printing of the compilation listing, a line-by-line document of the user's source program. This listing is of particular usefulness in providing the programmer with the following aids:

- A printed copy of the source program.
- A map (or listing) of source line locations in memory and a summary of major portions of the program.
- A classification of all compiler-detected errors.
- A cross-reference index, if requested.
- The object coding created for each source line, if specified.

The compilation listing consists of four divisions, or sublistings, that present in differing formats the compiler input and output data. The first division, compiler control statements, and the second division, data and coding statements, form the program listing. The third division, a program memory map, and an optional fourth division, a cross-reference index, form the supplementary listing.

Each division of the compilation listing is printed on a new page with a division heading consisting of two printed lines. A main header line, similar for all divisions but identifying each one, is printed on the first line (printline 4). Following each main header line is printed a subheader line (printline 6) which defines the data to be presented within the division. When an end-of-page condition is encountered for those divisions which extend beyond one page, line spacing to the top of the next page occurs and the division heading is repeated before printing of the text continues. Essentially the header lines are self-explanatory; however, they are considered briefly under the discussion of each division.

The compiler for the NCR Century 100 uses only the 51 characters contained in graphics code A when printing the compilation listing.

GRAPHICS CODE A									
0	5	A	F	K	P	U	Z	:	<
1	6	B	G	L	Q	V	$ or £	=	>
2	7	C	H	M	R	W	*	'	.
3	8	D	I	N	S	X	/	(	,
4	9	E	J	O	T	Y	+	)	-

The pound sign (£) and the dollar sign ($) are interchangeable depending on the installation. The 51st character is the space (Ø). Any character not included in the graphics code A is considered untranslatable and printed as a space.

Printing is controlled by options available to the programmer on the compiler specification worksheet. A review of these options is included in the following chart.

DIVISION	SHOULD OBJECT CODING BE LISTED?	SHOULD LISTING BE DOUBLE SPACED?	SHOULD CROSS REFERENCE INDEX BE LISTED?	DELETE DIGIT
COMPILER CONTROL STATEMENTS [FIXED FORMAT]	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.	ONLY THE FIRST CONTROL STATEMENT CANNOT BE DELETED SINCE IT SETS THE DELETE DIGIT.
DATA AND CODING STATEMENTS [FIXED FORMAT]	ENTRY [N] STATEMENTS ARE PRINTED. NO OBJECT CODING IS PRINTED. ENTRY [Y] ALL STATEMENTS CONTAINED IN THIS DIVISION ARE PRINTED WITH OBJECT CODING FOR EACH STATEMENT.	ENTRY [N] STATEMENTS LINES ARE SINGLE SPACED. ENTRY [Y] STATEMENTS LINES ARE DOUBLE SPACED.	NOT AFFECTED BY THIS OPTION.	STATEMENTS SUPPRESSED BY THE DELETE DIGIT ARE PRINTED AS THOUGH THEY WERE COMMENT LINES.
PROGRAM MEMORY MAP [FIXED FORMAT]	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.
CROSS REFERENCE INDEX [OPTIONAL LISTING]	NOT AFFECTED BY THIS OPTION.	NOT AFFECTED BY THIS OPTION.	ENTRY [N] THIS DIVISION IS NOT PRINTED. ENTRY [P] THIS DIVISION IS PRINTED IN PRESENTATION SEQUENCE [PAGE AND LINE]. ENTRY [A] THIS DIVISION IS PRINTED IN ALPHABETICAL SEQUENCE. ENTRY [B] BOTH ALPHABETICAL AND PRESENTATION SEQUENCE LISTINGS ARE PRINTED.	NOT AFFECTED BY THIS OPTION.

PROGRAM LISTING

The program listing is the printed copy of the source program and consists of two divisions. The first division, the compiler control statements, is the record of control information and the options requested on the compiler specification worksheet. The second division contains the data and coding statements which were prepared on the data and coding worksheets and input to the NEAT/3 compiler.

Compiler Control Statements Division

The compiler control statements division of the program listing is normally printed as a single page and may include three statements.

- The Control Statement

  This mandatory statement is the record of control information defined on the first page of the compiler specification worksheet (excluding the author statement at the bottom of the first page of the worksheet.) This statement is always printed in a fixed format and is not affected by the print options, nor can it be deleted by the delete digit, since it is this statement which sets the delete digit.

- The Author Statement

  This optional statement records the author's or programmer's name as entered at the bottom of the first page of the compiler specification worksheet.

- The Option Statement

  This optional statement is the record of control information defined on the second page of the compiler specification worksheet.

The preestablished format for this division is not affected by print options requested on the compiler specification worksheet. The text of the control statement is double-spaced and presented in two columns on the page, with approximately 1 inch spacing between the two columns of information. The author statement, when present, is printed as a single line across the page. The option statement, when present, follows the same format as the control statement.

Normally, the 1-page format for control statements satisfies most requirements. However, when additional pages are warrented, this format can be expanded as follows:

- Placing asterisks in columns 8 and 9 of either the author statement or option statement causes the statement line to be printed at the top of a new page and treated as a comment line. Comment lines are printed in a collapsed format that does not follow columnar field spacing and do not become part of the object coding.

- Inserting a separate dummy control statement with asterisks in columns 8 and 9 to precede either the author statement or option statement causes the dummy statement to be printed at the top of a new page and to be treated as a comment line. The author or option control statement is printed as the second line on the new page. In this instance the author and/or option statements are included in the object coding.

- Division Heading

  The division heading appears at the top of each page with the main header line printed on printline 4 and the subheader line printed on printline 6.

PROGRAM LISTING — CONTROL STATEMENTS DIVISION HEADING

NCR NEAT/3 000	MAIN PROGRAM	PROGRAM XXXXXXXX00		CONTROL STATEMENTS	DATE 00/00/00	PAGE 000
P/L CD	STATEMENT	COL STATED	ASSUMED	STATEMENT	COL STATED	ASSUMED

The contents of these two lines are considered separately in the following charts.

MAIN HEADER LINE — CONTROL STATEMENTS DIVISION	
Printer Image	Meaning
NCR NEAT/3 000	Compiler name and version number
MAIN PROGRAM	Defines page as part of main program, rather than an overlay
PROGRAM XXXXXXXX00	Program name and version number
CONTROL STATEMENTS	Division name
DATE 00/00/00	Date of compilation
PAGE 000	Page number

SUBHEADER LINE — CONTROL STATEMENTS DIVISION	
Columnar Title	Meaning
P/L	Page and line number
CD	Worksheet code (P)
STATEMENT*	Source statement
COL*	Source column number
STATED*	Source entry
ASSUMED*	Assumed entry whenever source entry is left blank
* These columnar titles are presented two to the page.	

- Error Comment And Compiler Comment Lines

  Error comment and compiler comment lines appear on a single-spaced printline immediately preceding the first printline of the statement. Both error comment and compiler comment lines are further explained in the next section, data and coding statements division. Error comment and compiler comment coding are discussed in further detail under separate publication in this manual. See COMPILATION PROCESS, tab 3, "Language Directory".

Data and Coding Statements Division

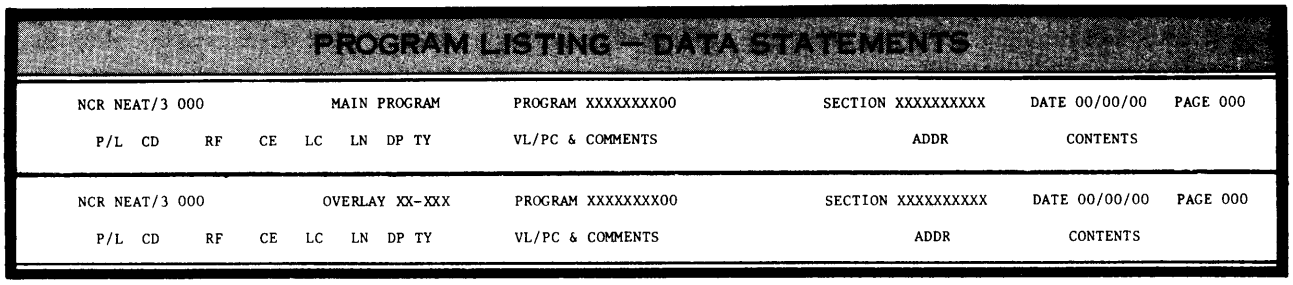
This division of the program listing is printed by overlay in the order that source statements are presented to the compiler. Data statements are printed first, followed by coding statements for the main program and each overlay. The first data statement is printed at the top of a new page, when the first coding statement is encountered, automatic line spacing to the top of a new page occurs before the coding statements are printed. Within the listing of either data or coding statements, additional new pages are printed whenever an end of page is encountered, an asterisk appears in columns 8 and 9, indicating a comment statement, or a section or overlay control instruction is encountered.

- Data Statements

  The first part of this division presents a listing of all data statements as prepared on the data worksheets.

  - Division Heading

    The division heading appears at the top of each page in this section with the main header line printed on printline 4. On printline 6, the subheader line is printed for data statements and includes mnemonic columnar titles to correspond with field definitions on the data worksheet.



The contents of these two lines are considered separately in the following charts.

MAIN HEADER LINE — DATA STATEMENTS	
Printer Image	Meaning
NCR NEAT/3 000	Compiler name and version number
MAIN PROGRAM or	Defines page as part of main program,
OVERLAY 00-000	or overlay group and number
PROGRAM XXXXXXXX00	Program name and version number
SECTION XXXXXXXXXX	Section name
DATE 00/00/00	Date of compilation
PAGE 000	Page number

SUBHEADER LINE — DATA STATEMENTS	
Columnar Title	Meaning
P/L	Page and line number of source statement
CD	Worksheet code (D)
RF	Reference name
CE	Definition code (R, A, I, or F)
LC	Defined location
LN	Defined length
DP	Defined decimal point
TY	Data type
VL/PC	Data value or data picture
COMMENTS	Programmer's remarks, if any
ADDR	Memory location
CONTENTS	Object coding, when requested

- Statement Format

The printline created for each data statement is in the same format and sequence as the information was entered initially on the data worksheet. Data fields are printed under their appropriate columnar titles in the subheader line. Comment statements are printed in a collapsed format without regard for columnar headings.

- Programmer Comments

When a partial-line comment has been included in a source line, the comment is preceded by an asterisk and printed under the comments field. If the comment is too long for the printspace available, a comment overflow is carried to the next line. When both comment overflow and object coding overflow occur, both overflow portions use the same line and are printed under their columnar fields.

Statements which are suppressed by the delete-digit logic are printed as though they were comment statements, in the collapsed-line format.

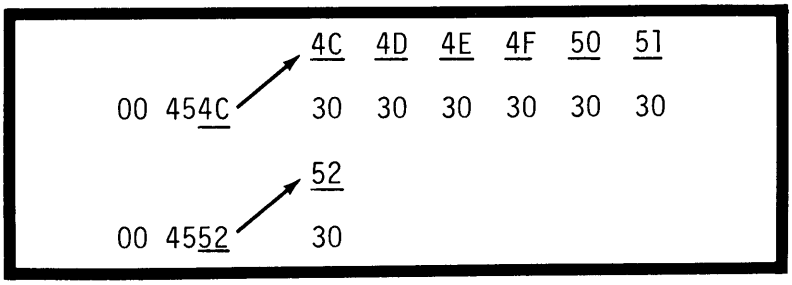
- Memory Location

All memory addresses for related source statements are carried in hexadecimal and printed under the ADDR field. Addresses are determined by hexadecimal counting as shown in the following examples.

```
 P/L  CD     RF     CE   LC   LN   DP TY        VL/PC & COMMENTS                ADDR          CONTENTS
009240 D SAVE       A        0006  0 U 000000                                 00 454C 303030303030
009270 D COUNTER    A        0001  0 U 0                                      00 4552 30
```

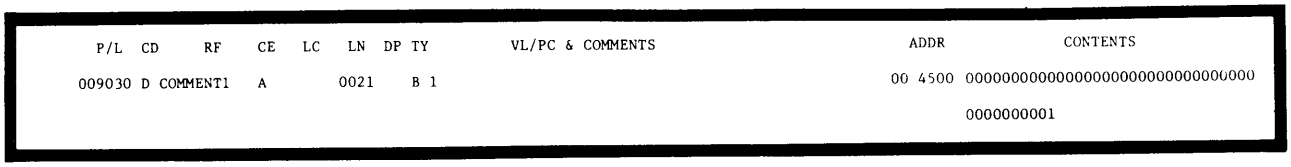
- The first statement defines an area, and shows its address as 00 454C.
- The length of the statement is six characters.
- The second statement also defines an area, and shows its address as 00 4552.

An expanded view of the ADDR and CONTENTS fields illustrates how this counting is accomplished.



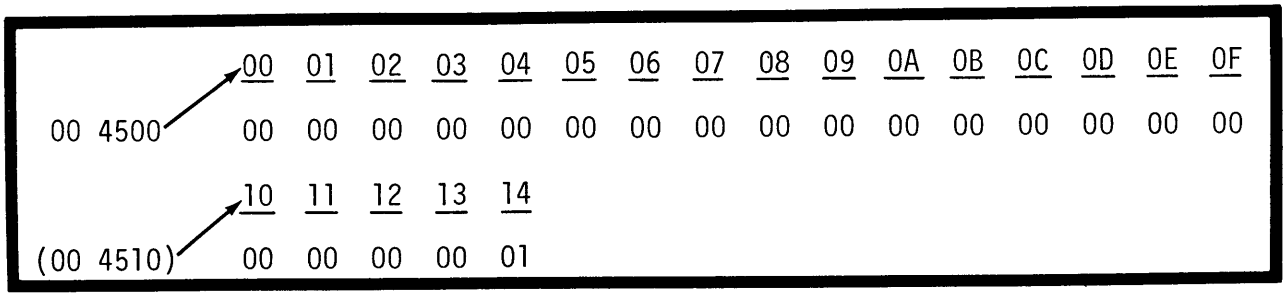
- The address, 00 454C, is the beginning address assigned to the first character of the statement.
- Each subsequent character increments the address by 1 hex digit to the end of the statement, ending at memory location 00 4551.
- The next statement is assigned the next available memory address, 00 4552.

In the next example, the object coding for the statement is too long to fit in the available printspace for the line. This condition causes the object to overflow to the next printline.



- The statement defines an area, and shows its address as 00 4500.
- The length of the statement is 21 characters.

An expanded view of this example is shown below.



- The address, 00 45<u>00</u>, is the beginning address of the first character of the statement.

- Each following character increments the address by 1 hex digit until the end of the printline (32 print positions) is reached.

- The object coding overflows to the next printline (the complete address of the first overflow character is shown in parentheses) and each character continues incrementing the address by 1 hex digit to the end of the statement, ending at memory location 00 45<u>14</u>.

In a final example, two statements are shown having the same memory address. This is possible when the second statement is a field definition within the first statement, and the field's relative location within the area definition is 0.

P/L	CD	RF	CE	LC	LN	DP	TY	VL/PC & COMMENTS	ADDR	CONTENTS
009300	D	CONTROL1	A		0004		X		00 4558	
009330	D		F		0003		X N		00 4558	4E2020
009360	D		F		0001		B 255		00 455B	FF
009390	D	CONTROL2	A		0004		X		00 455C	

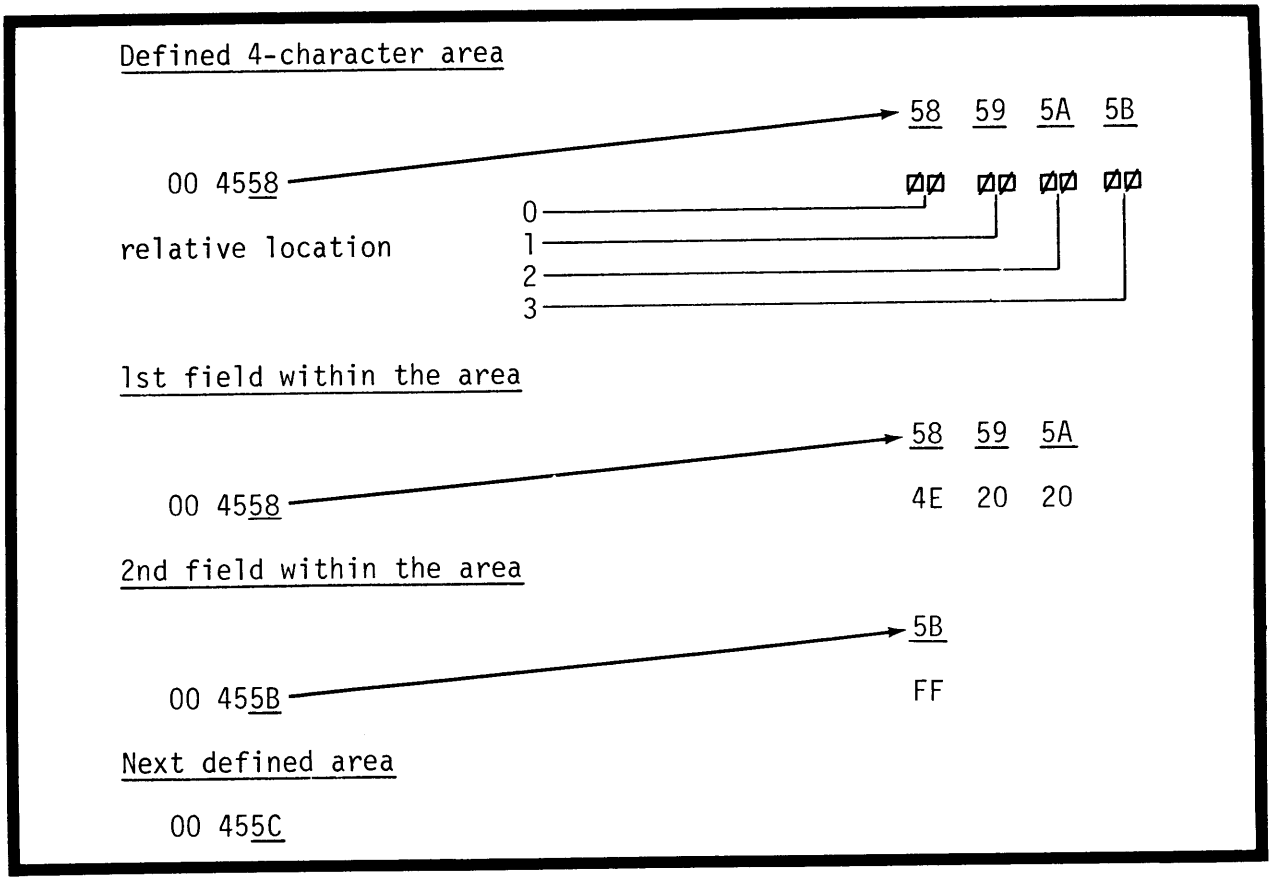
- The first statement defines an area, and shows its address as 00 45<u>58</u>. This is a 4-character, X-type area. Since no value entry is present, no object coding is printed for this line.

- The next statement is a 3-character X-type field with a relative location of 0 within the area. Therefore, the first character of the field is the first character of the area at memory address 00 45<u>58</u>.

- The third statement completes the area definition. It is a B-type field one character long (the remaining character within the area) and is at relative location 3 within the area. Its memory address is 00 45<u>5B</u>.

- Since the area definition is complete, the next area statement is assigned the next available address, 00 45<u>5C</u>.

An expanded view of this example is shown below.



- Object Coding

  When the programmer requests an object listing for the source program, both source and object appear side by side on the same line, with the object coding printed under the contents field.

  Object values for all data statements are interpreted in hexadecimal coded USASI, with 2 hex digits representing each 8-bit memory location. For a more detailed discussion of translating NCR Century internal code into hexadecimal, see NEAT/3 MANUAL, APPENDIX, tab 1, "Hexadecimal Code to NCR Century Internal Code."

  Consider the following examples.

```
P/L  CD     RF     CE   LC   LN  DP TY      VL/PC & COMMENTS          ADDR          CONTENTS
009090 D COMMENT2  A        0006  0 U 000000                          00 4515 303030303030
```

- The object coding for the value of this statement is 303030303030.
- 30 is the hexadecimal coding for 0.

Data object is left-justified, beginning in print position 101, and continuing through print position 132. Therefore, a total of sixteen 2-character hex codes can be printed on each line. In the instance where object coding exceeds the available printspace, object overflow occurs on the following line(s), as seen in the next example.

```
P/L  CD     RF    CE   LC   LN  DP TY        VL/PC & COMMENTS                    ADDR          CONTENTS
009120 D COMMENT3   A        0021    X ACCOUNT CREDITED WITH                  00 451B 4143434F554E54204352454449544544
                                                                                          2057495448
```

- The object coding for this statement is too long for one printline and causes object overflow. The meaning of the object coding is shown below.

```
41 43 43 4F 55 4E 54 20 43 52 45 44 49 54 45 44 20 57 49 54 48
 A  C  C  O  U  N  T     C  R  E  D  I  T  E  D     W  I  T  H
```

- Coding Statements

This part of the program listing presents a record of all coding statements as entered on the coding worksheets.

- Division Heading

The division heading appears at the top of each page in this section with the main header line printed on printline 4. On printline 6, the subheader line is printed for coding statements and includes mnemonic columnar titles to correspond with field definitions on the coding worksheets. The content of the main header line is the same format as previously described for data statements.

```
                    PROGRAM LISTING — CODING STATEMENTS

NCR NEAT/3 000        MAIN PROGRAM     PROGRAM XXXXXXXX00     SECTION XXXXXXXXXX   DATE 00/00/00   PAGE 000
P/L  CD    RF    OP                    OPERANDS & COMMENTS                ADDR          CONTENTS
```

The content of the subheader line is considered in the following chart.

SUBHEADER LINE — CODING STATEMENTS	
Columnar Title	Meaning
P/L	Page and line number of source statement
CD	Worksheet code (C)
RF	Reference name
OP	Operation code
OPERANDS	Operand expression
COMMENTS	Programmer's remarks, if any
ADDR	Memory location
CONTENTS	Object coding, when requested

- Statement Format

The printline created for each coding statement is in the same format and sequence as the coding worksheet, with the statement fields printed under the appropriate columnar titles in the subheader line.

- Programmer Comments

When a partial-line comment has been included in a source line, the comment is preceded by an asterisk and printed under the comments field. Comments beyond printer column 84 are carried to an overflow line as described for data statements.

- Memory Location

Memory locations for coding statements are carried in hexadecimal and printed under the ADDR field. Addresses are determined by hexadecimal counting as described for data statements, and because of hardware command formats, are normally incremented by 4 or 8.

- Object Coding

Object values for coding statements are left-justified, beginning in print position 101, and are printed in hexadecimal groups: two characters per memory location, 4 memory locations (8 characters) per group, with a space between groups. This grouping is created for ease of reference.

- Major Functions

When the coding of a source program specifies a major function, the generated coding (associated with the function) is not printed in the program listing. What does appear in the listing is the function definition as it was generated by answers to parameter questions. The questions and answers appear as comment lines, printed in two columns and following the card image format of the parameter questions.

The comments defining the function appear in the program listing in all cases except where the minimum number of parameters is not present. Generation of a major function is not attempted if the basic set of parameters for the function is not present. A comment indicating that the function was not generated will precede the parameters.

In addition, in the following instances, the basic set of parameters is present, but the function cannot be generated.

- An input or output data area is not indicated on the parameters, or the data definition for this area does not exist.
- The name of the function is not indicated, or one of the required exits has not been indicated.
- A key question for a particular function has not been answered.

When the generation of a function is aborted, the comments defining the function are preceded by a comment line indicating that the function was not generated.

When errors are detected in answers to parameter questions, whether or not the function is aborted, either one or two asterisks will appear to the left of the comment defining the particular parameter questions.

- One asterisk is used when an illegal answer is found, but a substitute answer can be generated. When a substitute answer is made, the substituted or assumed answer is printed in parentheses to the right of the user's illegal answer.

P/L	CD	RF	OP	OPERANDS & COMMENTS
100A05	C	* *	INDICATIVE FROM REC--G (F)	EXCESS O/P AREA TO BE--Z

NOTE

If the user's answer to this question is other than F or L, an F is assumed.

- Two asterisks are used when an incomplete answer is found, and due to the conditions of the user's answer, the logic associated with this particular answer is not generated.

```
    P/L CD       RF      OP                          OPERANDS & COMMENTS
100A14 C * **    I/P TOTAL 02--SALARY                O/P TOTAL 02--
```

NOTE

The name of field in the output record for the Total 2 Control is incomplete.

All comments with asterisks indicating an error are preceded by an error comment line which points to specific error comments listed in the language directory.

Below is a typical printout of the Accumulate function as it might appear in the program listing.

```
 P/L  CD      RF       OP                          OPERANDS & COMMENTS
100A01 C *    ACCUMULATE FUNCTION   PAGE NO--100      NAME--ACCFUNCT
100A02 C *      I/P AREA--WORK1                 O/P AREA--WORK2
100A03 C *      GETMORE (EX1)--CARDREG          NEXTDO (EX2)--REPORTER
100A04 C *      END OF DATA (EX3)--CLOSERT
100A05 C *      INDICATIVE FROM REC--F          EXCESS O/P AREA TO BE--Z
100A06 C *      NO. ADDED FLDS--2
100A07 C *      REC. CNT. FIELD--RECCNT
100A08 C *      REC. O/P IF ALL ADDED FLDS. ZERO--Y
100A09 C *        UNIOUE CHAR. OVR. FLD--OVRFIELD       UNIQUE CHAR--$
100A10 C *      NO. CONTROL KEYS--2
100A11 C *        KEY 1--NAME
100A12 C *        KEY 2--EMPLNO
100A13 C *      I/P TOTAL 01--HRSWORKEDA          O/P TOTAL 01--HRSWORKEDB
100A14 C *      I/P TOTAL 02--SALARY              O/P TOTAL 02--SALARYTD
```

- Options

  Certain options are available on the compiler control worksheet. If requested, these options affect or appear in the format of the data and coding statements division.

  - Line Spacing

    Normally printlines are single-spaced unless the double-space option is requested. Conventional single-spacing or double-spacing occurs throughout the division listing with the exception of error comment and compiler comment lines.

  - Object Coding

    When requested, the object coding is printed under the contents field for each statement in the collapsed line format.

  - Delete Digit

    Statements suppressed by the delete digit are printed as though they were comment lines.

- Error Comment and Compiler Comment Lines

  When either the error comment line or compiler comment line is created, it always precedes the printline of the statement to which it refers.

  An error comment line is distinguished by an E in the first print position on the line. This line contains one or more 4-character error codes which point to specific error comments listed in the language directory.

  A compiler comment line is distinguished by a C in the first print position on the line. The 4-character codes point to specific compiler comments listed in the language directory. These comments are used to bring a specific situation in the source program to the attention of the programmer. When a source statement contains both an error comment and compiler comment, they are flagged as an error comment line.

  When printlines are single-spaced, an error comment or compiler comment line is preceded by a double-spaced line.

```
   P/L  CD     RF    CE   LC   LN  DP TY        VL/PC & COMMENTS                ADDR            CONTENTS

 009030 D COMMENT1   A        0021    X      PREVIOUS BALANCE                00 4500 2020202020505245564 94F5553204241
                                                                                     4C414E4345
 009060 D COMMENT2   A        0021    X       PAY THIS AMOUNT                00 4515 2020202020205041592 0544849532041
                                                                                     4D4F554E54

E    *RF
 009090 D COMMENT 3  A        0021    X ACCOUNT CREDITED WITH                00 452A 4143434F554E5420435 2454449544544
                                                                                     2057495448
 009120 D ACCUMCHRG  A        0007 02 D +0000.00                             00 453F 3030303030302B
 009150 D            F  00000 0006 02 U 0000.00                              00 4546 303030303030
 009180 D ACCUMPAY   A        0007 02 D +0000.00                             00 454C 3030303030302B
 009210 D            F  00000 0006 02 U 0000.00                              00 4553 303030303030
```

When printlines are double-spaced, an error comment or compiler comment line is followed by a single-spaced line.

```
  P/L  CD     RF     CE   LC    LN   DP TY        VL/PC & COMMENTS                ADDR          CONTENTS
009030 D COMMENT1   A         0021   0 X      PREVIOUS BALANCE                 00 4500 2020202020505245564 94F5553204241
                                                                                        4C414E4345
009060 D COMMENT2   A         0021     X      PAY THIS AMOUNT                  00 4515 2020202020205041592054484953204 1
                                                                                        4D4F554E54
E    *RF
009090 D COMMENT 3  A         0021     X ACCOUNT CREDITED WITH                 00 452A 4143434F554E5420435245444954454 4
                                                                                        2057495448
```

As an additional aid to the programmer, these special comments are printed at the bottom left of the page.

- The page on which an error comment or compiler comment line occurs will contain, EXAMINE THIS PAGE.

  - If the next consecutive page is without an error comment or compiler comment line it will contain, EXAMINE PRIOR PAGE.

  - If an error comment or compiler comment line occurs on two consecutive pages, the second page will contain, EXAMINE THIS PAGE AND PRIOR PAGE.

This insures that one or the other of these comments is visible on the outward edge of the folded printer listing.

- Nonformat Statements

Comment statements are remarks that the programmer makes to document his program. These may be partial-line comments, that are part of the source-line in which they occur, these comments do not become part of the object program.

In addition, the programmer may indicate, by an asterisk in columns 8 and 9, that an entire line is a comment and is for documentation of the program. A complete-line comment is printed in collapsed-line format without regard for columnar spacing of the statement. Source statements which are deleted by the delete digit option are printed as complete-line comments. See INTRODUCTION AND DATA, tab 3, "Programming Worksheets."

Other nonformat statements include flowcharting and error statements.

- First Print Column Flags

Error comment and compiler comment lines that appear in the program listing are flagged by single characters in the first print column. The letter E denotes an error comment line; and the letter C denotes a compiler comment line.

SUPPLEMENTARY LISTING

The supplementary listing is the printed copy of certain special classifications of the object program and for format purposes consists of two divisions. The first division, program memory map, shows that portion of memory assigned to the compilation. Since the second division (the cross-reference index) is an optional listing, it is only printed when requested by the programmer to show what page/lines use each reference.

Program Memory Map Division

The program memory map division presents a summary of the memory requirements of the object program and includes that portion of memory used by software over which the programmer has no direct control.

- Division Heading

  The division heading appears at the top of the page with the main header line printed on printline 4 and the subheader line printed on printline 6.

SUPPLEMENTARY LISTING — PROGRAM MEMORY MAP DIVISION HEADING

```
NCR NEAT/3 000                    PROGRAM XXXXXXXX00          MEMORY MAP          DATE 00/00/00   PAGE 000

ALLOCATION   BEG ADD   END ADD   DEC LEN   SUBROUTINE NAME AND DECIMAL LENGTH OF COMPILER INCLUDED SUBROUTINES
```

The contents of these two lines are considered separately in the following charts.

MAIN HEADER LINE — PROGRAM MEMORY MAP DIVISION

Printer Image	Meaning
NCR NEAT/3 000	Compiler name and version number
PROGRAM XXXXXXXX00	Program name and version number.
MEMORY MAP	Division name
DATE 00/00/00	Date of compilation
PAGE 000	Page number

SUBHEADER LINE — PROGRAM MEMORY MAP DIVISION	
Columnar Title	Meaning
ALLOCATION	Allocation name or overlay
BEG ADD	Beginning address
END ADD	Ending address
DEC LEN	Decimal length
SUBROUTINE NAME AND DECIMAL LENGTH OF COMPILER INCLUDED SUBROUTINES	Up to five subroutines listed per line.

- Map Text

  The memory map accounts for the executive and program allocations and, when applicable, the overlay directory, buffers, data, file tables, coding, literals, and subroutines within the main program and each overlay. The name of each compiler included subroutine is printed. The length of major portions of the program and subroutine are printed in decimal. The starting addresses and ending addresses are printed in hexadecimal.

Cross Reference Index Division

The cross-reference index division, when requested by the programmer, is printed from the cross-reference file created during data and coding statement processing of the compilation.

- Division Heading

  The division heading appears at the top of the page with the main header line on printline 4 and the subheader line on printline 6.

SUPPLEMENTARY LISTING — CROSS REFERENCE INDEX DIVISION					
NCR NEAT/3 000		PROGRAM XXXXXXXX00	CROSS REFERENCE	DATE 00/00/00	PAGE 000
P/L RF	ADDR	PAGE AND LINE OF REFERENCING STATEMENTS			

The contents of these two lines are considered separately in the following charts.

MAIN HEADER LINE — CROSS-REFERENCE INDEX DIVISION	
Printer Image	Meaning
NCR NEAT/3 000	Compiler name and version number
PROGRAM XXXXXXXX00	Program name and version number
CROSS REFERENCE	Division name
DATE 00/00/00	Date of compilation
PAGE 000	Page number

SUBHEADER LINE — CROSS-REFERENCE INDEX DIVISION PRESENTATION SEQUENCE	
Columnar Title	Meaning
P/L	Page and line number
RF	Reference name
ADDR	Memory location
PAGE AND LINE OF REFERENCING STATEMENTS (see NOTE)	

SUBHEADER LINE — CROSS-REFERENCE INDEX DIVISION ALPHABETICAL SEQUENCE	
Columnar Title	Meaning
RF	Reference name
P/L	Page and line number
ADDR	Memory location
PAGE AND LINE OF REFERENCING STATEMENTS (see NOTE)	

NOTE

Page/line entries of referencing statements from overlays different from that of the reference name will be preceded by an asterisk.

- Index text

  The cross-reference index may be printed in either or both of the following formats, as designated on the compiler specification worksheet.

  - Presentation Sequence

    Each reference name is listed by page and line number in descending order, together with its object address. The page and line numbers of all statements in which the reference name appears as an operand are listed in the order the statements are presented to the compiler.

  - Alphabetical Sequence

    The reference names are alphabetically sorted prior to being printed, and are listed in that order, together with their object addresses. The page and line numbers of all statements in which the reference name appears as an operand are listed in the order the statements are presented.

- Options

  The cross-reference division may be requested to be listed in either presentation or alphabetical sequence, or both.

LANGUAGE DIRECTORY

INTRODUCTION

The language directory is part of the system software and exists as a data file on the compiler disc pack. The directory is copied from disc to printer by the NEAT/3 directory list routine. Any number of copies of the directory can be printed, as needed; these copies are generally retained as standard reference documents. In this way, the directory does not have to be printed following every compilation.

Changes and/or additions to the language directory are performed as required by the NEAT/3 directory management routine. Minor changes and/or additions can be made in the printed directory as hand corrections, until the number of changes and additions warrant the printing of a new edition of the directory or until a new version of the NEAT/3 Compiler is released.

Because the language directory has an extract of all formatting and procedural rules, it serves as a grammatical text governing the use of the NEAT/3 language.

SECTIONS

The language directory is printed in five sections.

Section I - NEAT/3 Release Notifications

This section contains the version number of the corresponding NEAT/3 Compiler and comments pertaining to current NEAT/3 language usage and restrictions including release notifications for future applications.

Section II - NEAT/3 System Tags

This section contains a listing of two types of reference tags used in the NCR Century system:

- General system tags, identified by the prefix >EXEC., are presented first. Each tag is briefly defined.
- File system tags, identified by the prefix FR.$, are presented next and are likewise briefly defined.

Section III - SPUR Comments

The comments found in this section are precompilation errors detected during input to the source program utility routines and may appear in the remarks column of the SPUR exception report. Each comment is briefly defined.

Section IV - Format and Expression Error Comments

This section, which is indexed by a series of 4-character error codes with mnemonic significance, generally relates to elementary errors of invalid format or expressions of information as entered in the source program worksheets. Statements and instructions flagged with error comments from this section require a correction to the referenced statement or operand expression.

- Statement Error Comments

  These error comments (listed below) are indexed alphabetically using the 2-character field mnemonics assigned as column titles in the program listing.

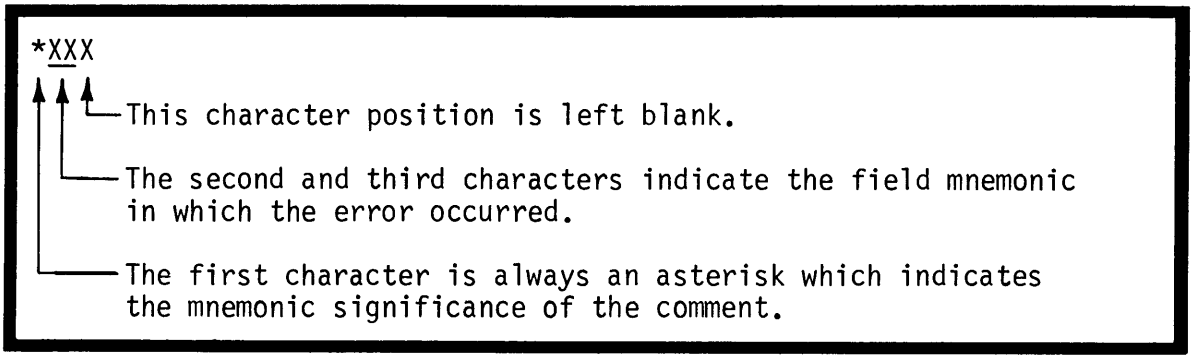
Field Mnemonics	Source Field
CD	Presentation code format
CE	Definition code format
DP	Decimal point field format
LC	Location field format
LN	Length field format
*LO	Length/Offset format
*NA	Index register assignment
OP	Operation code format
PC	Picture field format
*SK	Skipped field format
RF	Reference field format
TY	Type field format
VL	Value field format

*These error codes do not represent specific source fields, but do reflect error classifications detected in source statements.

In addition, certain miscellaneous field mnemonics as assigned, as listed below.

Field Mnemonic	Title
CC	Control statements
PS	Presentation sequence error

The following illustration shows the format of the statement error comments.



Example:

```
    P/L  CD    RF    CE   LC   LN  DP  TY        VL/PC & COMMENTS

E     *CE
  009300 D CONTROL1  S        0004      X
```

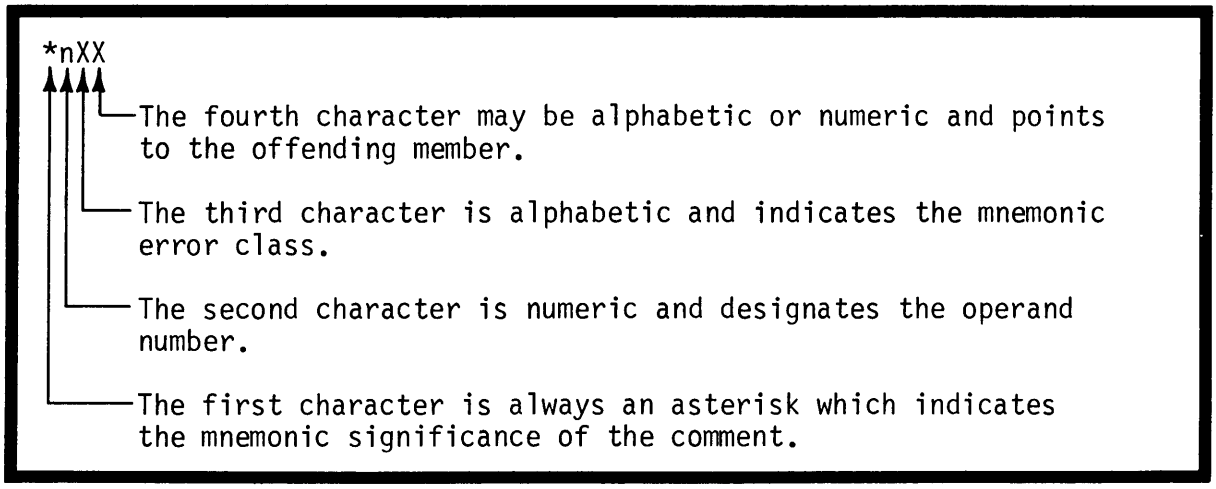
Using the error code, *CE , the corresponding entry in the error comment directory indicates the following:

```
*CE   *DEFINITION CODE FORMAT.
           THE DEFINITION CODE FORMAT
      (COLUMN 18) WILL CONTAIN
      AN R, FOR RECORD DEFINITIONS WHICH
      ARE VALID ONLY IN CONTEXT WITH FILE
      CONTROL SPECIFICATIONS;
      AN A, FOR AREA DEFINITIONS WHICH ARE
      USED TO RESERVE OR DEFINE AN AREA IN
      MEMORY;
      AN I, FOR ITEM DEFINITIONS WHICH ARE
      VALID ONLY IN CONTEXT WITH TABLE
      CONTROL SPECIFICATIONS;
      AN F, FOR FIELD DEFINITIONS WHICH ARE
      USED AS SUBGROUPS OF THE ABOVE
      DEFINITION FORMAT CODES.
```

Since the definition code must be an R, A, I, or F, the S entry is invalid and must be corrected.

- Operand Error Comments

These error comments are indexed alphabetically on the 3rd character by three error classes: D for duplicate, I for illegal, and U for undefined. Each class is further indexed numerically and alphabetically on the 4th character. The following illustration shows the format for operand error comments.



Example:

```
    P/L  CD   RF      OP                    OPERANDS & COMMENTS

E     *1I$
  011210 C          BR    $30
```

Using the error code, *1I$, the corresponding entry in the error comment directory would indicate the following:

```
*-I$  *ILLEGAL LOCAL TAG.
           OPERANDS EXPRESSING LOCAL TAGS
      MUST BE AT LEAST TWO CHARACTERS AND
      NOT MORE THAN THREE CHARACTERS IN
      LENGTH BEGINNING WITH THE $ FOLLOWED
      BY NUMERICS AND MAY NOT EXCEED THE
      RANGE $00 THRU $24.
```

Section V - Procedural Error Comments and Compiler Comments

This section of the directory, which is indexed by a series of 4-character error codes with no mnemonic significance, generally relates to errors discovered by the compiler with regard to procedural usage of the NEAT/3 language. Error classifications are arranged in the following alphabetical order:

A - general error comments.

B - compiler usage comments. These comments are not expressions of source language errors, but rather, indicate the compiler has encountered internal difficulty in accessing or processing a specific command. Since the compiler indicates this command cannot be generated, other coding must be devised for a successful compilation. If an error comment occurs in this classification, it should be reported to your NCR representative.

C - compiler comment codes. These comments refer to some specific application used in the source program. The application is governed by a conditional restriction. The compiler creates these comments to bring the situation to the attention of the programmer.

D - arithmetic and manipulative error comments. These comments reflect arithmetic and data manipulative errors.

E - general error comments.

F - file control error comments. These comments reflect invalid or conflicting entries detected in file specifications.

G, H, J - input/output error comments, and linking error comments.

K - reserved for NCR usage.

L - literal error comments.

M, N, P, Q, R - reserved for NCR usage.

S - subroutine error comments.

T - table control error comments.

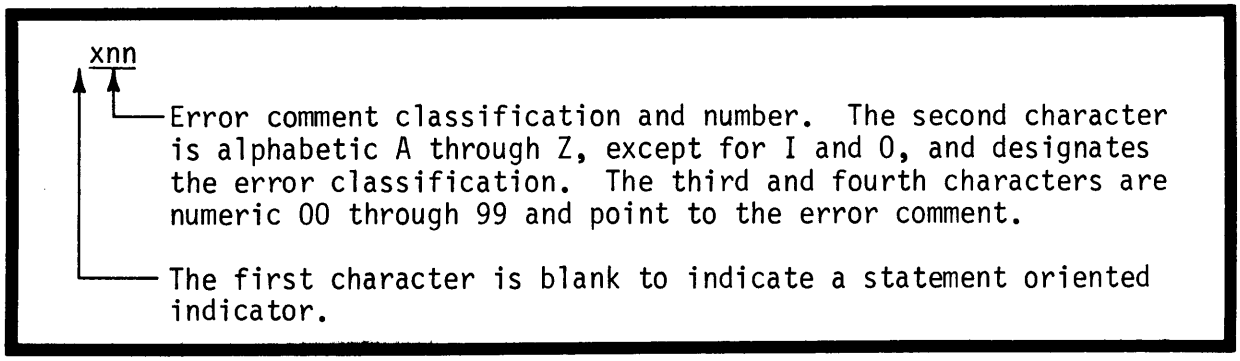
U, V, W, X, Y, Z - reserved for user macro-generator error comments.

Error comments are further arranged within each error classification in numeric order, ranging from 00 through 99 inclusive. Statements and instructions flagged with error comments from this section must be redefined or correctly expressed before a successful compilation can be created.

Compiler comments from this section relate to conditional situations that require the attention of the programmer.

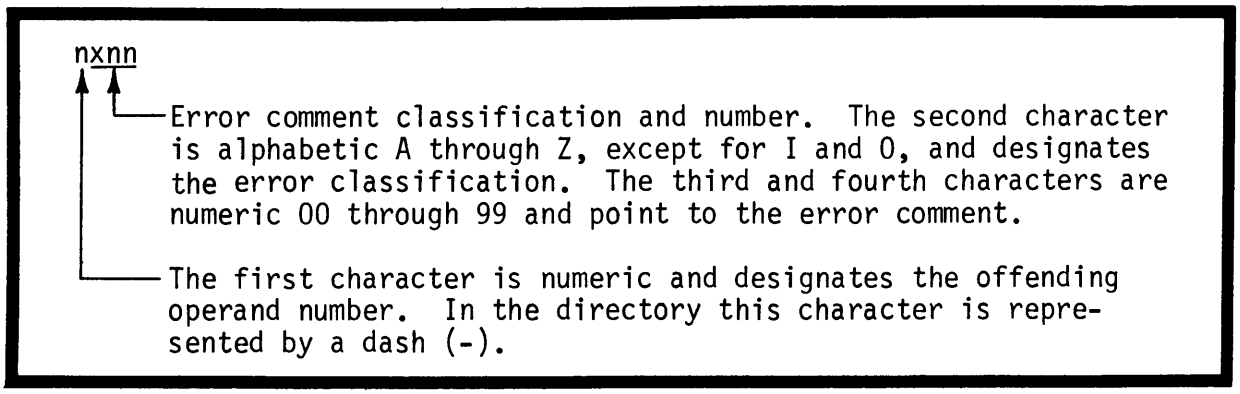
- Statement Error Comments

These error comments are indexed alphabetically by error classification on the second character and numerically by comment on the third and fourth characters. The following illustration shows the format of statement error comments.



- Operand Error Comments

These error comments are indexed alphabetically by error classification on the second character and numerically by comment on the third and fourth characters. The format of an operand error comment is shown below.



NOTE

Generally the first character is replaced by an operand number when printed in the error comment line, i.e., 1A07; however, in some instances, the error code in the error comment line may appear as expressed in the directory, i.e., -A07.

TABLE OF NEAT/3 INSTRUCTIONS

Instr.	Operands	Description
ADD	A , B	(A) + (B) → B
ADD	A , B , C	(A) + (B) → C
ADDC	A , B , Z	If overflow, branches to Z.
ADDC	A , B , C , Z	
ADDL	A , B , Z	If overflow, links to Z.
ADDL	A , B , C , Z	
ADDR	A , B ,	Rounds off decimal places.
ADDR	A , B , C	
ADDRC	A , B , Z	Rounds off decimal places; if overflow, branches to Z.
ADDRC	A , B , C , Z	
ADDRL	A , B , Z	Rounds off decimal places; if overflow, links to Z.
ADDRL	A , B , C , Z	
BEGDBG		Establishes point at which debugging begins.
ENDDBG		Establishes point at which debugging ends.
BLKCHK	FR, A , Z	Branches to Z if block length is ≥ (A).
BLKOUT	FR	Outputs short block.
BR	Z	Branches; stores no link.
BRE	Z	
BRG	Z	
BRL	Z	
BRGE	Z	
BRLE	Z	
BRU	Z	
CLOSE	FR	Closes named file.
CLOSEO	FR	Closes and obsoletes named magnetic media file.
CNSOUT	A	Relays (A) to the operator.
CNSIN	A . B	Relays (A) to the operator and requires an input to B in hexadecimal.
CNSINA	A , B	Relays (A) to the operator and requires an input to B in either alpha or numeric.
COMP	A , B	Compares (A) to (B).
COND	A , B	(A) → B; length of B must be one-half the length of A.
COPYA	A	Copies entire source program. (A = PROGNAME)
COPYP	A , B , C	Copies partial source program. (B = starting page/line #, C = ending page/line # or END$)
COPYR	A , B , C	Copies partial source program. (B = starting reference name, C = ending reference name or END$)
DEFALT	FR1,...,FR4	Closes current section; opens next section.

Instr.	Operands	Description
DELETE	FR	Removes current record from a source-destination chained file.
DELETE	FR, WA	Removes current record from file and stores record in workarea.
DIV	A , B , C	(B) ÷ (A) → C
DIVC	A , B , C , Z	If overflow, branches to Z.
DIVL	A , B , C , Z	If overflow, links to Z.
DIVR	A , B , C	Rounds off decimal places.
DIVRC	A , B , C , Z	Rounds off decimal places; if overflow, branches to Z.
DIVRL	A , B , C , Z	Rounds off decimal places; if overflow, links to Z.
DSCOFF	A	Informs operator to remove disc pack on unit referenced by (A).
END$		Indicates end of input data.
ENTRY		Establishes entry point into program or overlay.
FINISH		Returns control to Monitor.
GET	FR	Presents records sequentially.
GET	FR, WA	Presents records sequentially; moves them to a workarea.
LGET	FR	Reads a label from magnetic tape.
RGET	FR, A	Presents record addressed by (A).
RGET	FR, A , Z	Presents record addressed by (A); branches if null block.
SGET	FR, Z	Presents next record; branches if null block, E-O-B, E-O-S.
SGETC	FR, Z1, Z2	Presents next record; branches Z1 as SGET; branches Z2 if last record in block has been presented and record overflow flag is OFF.
SGETL	FR, Z1, Z2	Presents next record; branches Z1 as SGET; branches Z2 if the last record in the block has been presented.

Legend:

A , B , C	operands	FR	file reference
WA	workarea	TR	table reference
( )	contents	Z	branch address

Instr.	Operands	Description
INSERT	FR, WA	Places a record from workarea into current position in sorted file.
LINK	Z	Branches and stores a link.
LOG	A	Places (A) in the log.
MARK	FR, A	Stores in the A the file location of the current record in memory.
MOVE	A , B	Moves (A) to B.
MULT	A , B , C	(A) x (B) → C or (B) x (A) → C (smaller x larger).
MULTC	A , B , C , Z	If overflow, branches to Z.
MULTL	A , B , C , Z	If overflow, links to Z.
MULTR	A , B , C	Rounds off decimal places.
MULTRC	A , B , C , Z	Rounds off decimal places; branches if overflow.
MULTRL	A , B , C , Z	Rounds off decimal places; links if overflow.
* (RENAME)		Assigns reference without establishing new program region.
OMIT		Omits source line indicated in columns 1-6 during recompilation.
OMIT	PAGELINE	Omits source lines during recompilation; beginning page/line indicated in columns 1-6, ending page/line indicated by operand.
OPEN	FR	Opens named file.
OPENS	FR, A	Opens named file at section (and relative sector) specified by A.
OVRLAY		Indicates beginning of new program overlay.
OVRLAYG		Indicates beginning of new program overlay and new overlay group.
PUT	FR	Places a record in named file.
PUT	FR, WA	Places a record from named workarea into a file.
LPUT	FR	Places a label on magnetic tape.
RDUMP		Creates a rescue dump.
RFILE	FR, WA	Randomly places a record in a file.
RELINK		Removes address stored in link list during last LINK: returns control to that address.
RELINK	Z	Removes address stored in link list during last LINK; transfers control to the routine referenced by the operand.
RESET	FR, A	Restores into memory the record whose address was saved by MARK.
RETNOR		Causes a return to the sorting program to sort the record normally.
RETDEL		Causes a return to the sorting program to delete the current record.
RETADD	WA	Causes a return to the sorting program to insert a new record.

Instr.	Operands	Description
ROPENS	FR	Reopens named file as a source file.
ROPEND	FR	Reopens named file as a destination file.
ROPENR	FR	Reopens named file as a source-destination file.
ROPENP	FR	Reopens named file as a piggyback file.
SECT		Starts a program section. (Reference columns 8-17 = section name.)
SETPL	A	Causes SPUR to renumber the next source statement with the page and line number indicated by operand A.
SPREAD	A , B	Spreads A throughout B.
SUB	A , B	(B) - (A) → B
SUB	A , B , C	(B) - (A) → C
SUBC	A , B , Z	If overflow, branches
SUBC	A , B , C , Z	to Z.
SUBL	A , B , Z	If overflow, links
SUBL	A , B , C , Z	to Z.
SUBR	A , B	Rounds off decimal places.
SUBR	A , B , C	
SUBRC	A , B , Z	Rounds off decimal places;
SUBRC	A , B , C , Z	branches if overflow.
SUBRL	A , B , Z	Rounds off decimal places;
SUBRL	A , B , C , Z	links if overflow.
TBEGB	TR	Initializes the table to be built.
TBEGF	TR	Initializes the table to perform all other functions.
TBILDD	TR, A , B , Z	Builds or inserts (A) into the item location specified by (B); branches when location specified by (B) is beyond the limits of the table or when the specified item is active.
TBILDN	TR, A , Z	Builds (A) into the next location; branches when next location is beyond limits of the table.
TDEL	TR, , Z	Delets the current item; branches when no items remain in the table or when the current item is nonactive.
TDEL	TR, A , Z	Stores current item in A, then deletes current item from the table; branches as TDEL above.

Instr.	Operands	Description
TFINDB	TR, A , , Z	Performs a binary search for item whose key is (A); branches when the desired item either is beyond the range of the table (turns G flag on) or is within the range, but physically missing from the table (turns E flag on).
TFINDB	TR, A , B , Z	Performs a binary search for the item whose keys are (A) and (B); branches as TFINB above.
TFINDD	TR, A , Z	Accesses the item whose position is specified by (A); branches either when the specified location is beyond the range of the table or when the specified item is nonactive.
TFINDN	TR, Z	Accesses the next item; branches when the next item is beyond the limits of the table.
TFINDO	TR, A, , Z	Beginning where previous search ended, performs sequential search for item whose key is (A); branches when desired item is beyond the range of the table (turns G flag on) or is within the range but physically missing from the table (turns E flag on).
TFINDO	TR, A , B, Z	Beginning where previous search ended, performs search for item whose keys are (A) and (B); branches as TFINDO above.
TFINDP	TR, Z	Accesses the previous item; branches when the previous item is beyond the limits of the table.
TFINDR	TR, A , , Z	Performs a serial search for item whose key is (A); branches if no item exists whose key is (A).
TFINDR	TR, A , B , Z	Performs a serial search for item whose keys are (A) and (B); branches if no item exists whose keys are (A) and (B).
TFINDS	TR, A , , Z	Performs sequential search for item whose key is (A); branches when the desired item either is beyond the range of the table (turns G flag on) or is within the range but physically missing from the table (turns E flag on).
TFINDS	TR, A , B , Z	Performs sequential search for item whose keys are (A) and (B); branches as TFINDS above.
TJUMP	TR, A , Z	Calculates relative location of an item in a table and transfers control to the corresponding transfer-of-control instruction in a list whose base is referenced by A. Branches to Z if instruction is other than a LINK or BR or if table reference (TR) finds an off table condition.

Instr.	Operands	Description
TMARK	TR, A	Stores in A the address of the current item.
TPACK	TR, Z	Moves active items to beginning of table, inactive items to end; branches if table contains no active items.
TRESET	TR, A , Z	Makes accessible the item whose address is stored in A; branches if the address is no longer within the current limit of the table.
TSERT	TR, A , Z	Inserts (A) into the table at the current location.
TSHIFT	TR, A	Destroys last item; moves other items toward end of table; inserts (A) into first position.
TSORTA	TR	Sorts items in table into ascending sequence.
TSORTD	TR	Sorts items in table into descending sequence.
WRITEBI	FR	Causes software to write immediately the current block of a source-destination file.
WRITSP	FR	Causes the software to write the current block.
XPAND	A , B	(A) → B; the length of B must be twice the length of A.

PERIPHERAL TYPE CODES

PERIPHERAL DESCRIPTION	MODEL NUMBER	PERIPHERAL TYPE CODE
CARDS		
Integrated (300 cpm) Card Reader	682-100	00
Freestanding (750/100 cpm) Card Reader/Punch	686-100	01
Freestanding (750 cpm) Card Reader	686-201	02
Freestanding (100 cpm) Card Punch	686-301	03
Freestanding (300 cpm) Card Punch	688-301	04
PAPER TAPE		
Integrated (1000 cps) Paper Tape Reader	662-100	10
Freestanding (1500 cps) Paper Tape Reader	660-101	11
Freestanding (200 cps) Paper Tape Punch	665-101	12
PRINTERS		
Integrated		
132 column, 450 lpm, single numeric	640-102	20
132 column, 450 lpm, double numeric	640-102	21
132 column, 1500 lpm, single numeric	640-200	22
132 column, 1500 lpm, double numeric	640-200	23
160 column, 1500 lpm, single numeric	640-210	24
160 column, 1500 lpm, double numeric	640-210	25
132 column, 600 lpm, expanded character set	640-300	26
Freestanding		
132 column, 450 lpm, single numeric	640-102	60
132 column, 450 lpm, double numeric	640-102	61
132 column, 1500 lpm, single numeric	640-200	62
132 column, 1500 lpm, double numeric	640-200	63
160 column, 1500 lpm, single numeric	640-210	64
160 column, 1500 lpm, double numeric	640-210	65
132 column, 600 lpm, expanded character set	640-300	66

PERIPHERAL DESCRIPTION	MODEL NUMBER	PERIPHERAL TYPE CODE
DISCS		
108 kb Integrated NCR Century 100 Disc Unit	655-101	30
108 kb Second Integrated NCR Century 100 Disc Unit	655-102	31
108 kb Freestanding Disc Unit	655-201	32
180 kb Freestanding Disc Unit	655-202	33
MAGNETIC TAPES		
Phase Mode, 9 Channels - 1600 bpi		
80 kb Single Unit (50 ips)	633-111	40
80 kb Dual Unit (50 ips)	633-121	43
144 kb Single Unit (90 ips)	633-211	41
144 kb Dual Unit (90 ips)	633-221	44
240 kb Single Unit (150 ips)	633-311	42
NRZ Mode		
40 kc Unit , 9 Channels - 800 bpi	633-119	45
10-28-40 kc Unit , 7 or 9 Channels - 200/556/800 bpi	633-117	48
CRAM		
145 million character capacity	653-101	50
Console		
Input/Output Writer (integrated)		F2
Input/Output Writer (freestanding)		F3
MICR		
MICR Sorter (600 DPM, 11 Pockets)	670-101	80
MICR Sorter (1200 DPM, 18 Pockets)	671-101	81
OCR		
Optical Character Reader	420-1	70
Optical Character Reader	420-2	71
OCR connected to 329 Controller		72
ENCODERS		
Magnetic Tape Encoder	736	90
PLOTTER		
Calcomp Plotter		A0

PERIPHERAL DESCRIPTION	MODEL NUMBER	PERIPHERAL TYPE CODE
COMMUNICATIONS MULTIPLEXORS		
Communication Multiplexor	621-101	B0
Communication Multiplexor	621-102	B1
REMOTE TERMINALS		
Online Adapter		C0

DATA FORMAT CODES

EXTERNAL CODE SET	DATA FORMAT CODE
Punched Card Stacking Functions Not Used	
No Translation - 1 Character Per Column	00
Standard NCR Century H Set	01
Standard NCR Century A Set	02
315 Hollerith Set	03
Binary	04
Punched Card Stacking Functions Used	
No Translation - 1 Character Per Column	10
Standard NCR Century H Set	11
Standard NCR Century A Set	12
315 Hollerith Set	13
Binary	14
Punched Paper Tape	
NCR Century Standard Code Set -- USASI	20
User Defined	2D
Printer	
No Format Control	00
Standard Vertical Format Control	30
Reporter	31
Magnetic Tape	
NCR Century Internal Code -- USASI	00
IBM BCD Code	40
IBM EBCDIC Code	41
315 Internal Code	42
315 Internal Code and Label Format	43
735 Magnetic Tape Encoder (International)	
NCR Century Internal Code -- USASI	00
IBM BCD Code	40
315 Internal Code	42

736 Magnetic Tape Encoder (Domestic)

NCR Century Internal Code -- USASI	00
IBM EBCDIC Code	41

Disc and CRAM

NCR Century Internal Code -- USASI	00

SYSTEM TAGS

The following is a list of reference tags used in the NCR Century System. The NEAT/3 compiler automatically assigns these reference tags and their reserved memory area during a compilation run. The user may access these areas by using their reference tags as operands of instructions in his own program.

GENERAL SYSTEM TAGS

>EXEC.ACDATE

This tag refers to a 6-character memory field (X type) which contains the actual date expressed as day, month, year. (For instance, the actual date May 17, 1968 is expressed as 170568.) This field contains three 2-character fields: >EXEC.ACDATEDA, >EXEC.ACDATEMO, and >EXEC.ACDATEYR. The programmer may use these as source fields, never as destination fields.

>EXEC.VRDATE

This tag refers to a 6-character memory field (X type) which contains the virtual date expressed as day, month, year. (For instance, the virtual date May 17, 1968 is expressed as 170568.) This field contains three 2-character fields: >EXEC.VRDATEDA, >EXEC.VRDATEMO, and >EXEC.VRDATEYR. The programmer may use these as source fields, never as destination fields.

>EXEC.MF01 through >EXEC.MF30

These 30 tags refer to the 1-character fields (X type) which contain Monitor Flags 01 through 30. The information in these fields is initially specified by the programmer's entries in positions 51 through 80 of the NEXTDO, HEADCS, and NEXTBR instructions. During program execution, the programmer may alter the contents of these flags with a MOVE instruction. These fields may be used as source-destination fields.

>EXEC.REMAINDER

This tag refers to a 20-character field (U-type) that contains the remainder resulting from a decimal-divide instruction in the user's program. This remainder is stored as an unsigned integer. The programmer may use this as a source field, never as a destination field.

>EXEC.SIMOPTSW

This tag refers to a 1-character binary field in memory. The bits M_1 through M_5 of this field simulate option switches for control of Monitor. See OPERATING SYSTEM, MONITOR, "The Monitor Simulated Option Switch" for the function of each of these bits.

>EXEC.VRJULDATE

This tag refers to a 5-character, X-type field that contains the Julian date written as YRDAY. The DAY is expressed sequentially, from 1 to 365.

>EXEC.RMPRGFINAD

This tag refers to a memory field that contains the final address of a program plus 1.

>EXEC.ZERO

This tag is used with the ORIGIN instruction to assign an absolute starting memory address to coding or data so that it remains relocatable.

>EXEC.SORTSUD

This tag refers to a 2-character field that contains the symbolic unit designator (SUD) of the output unit when it is assigned by the General Sort program. The programmer may use this field as a source field, never as a destination field.

FILE-ORIENTED SYSTEM TAGS

The following fields are compiled for each disc and CRAM file in the user's program. To access the information in them, the user codes an instruction with a qualified operand. The qualifier is the file reference name, and the second part of the operand specifies the field desired. For instance, to determine the status of the last block of data read from the chained master file, the programmer codes a COMP MASTERFILE.$NULLFLAG, '1' instruction.

FileReference.$NONRESTOR

This tag refers to a 1-character flag (B type) associated with each chained file on CRAM or disc. After an INSERT instruction has inserted a record into its buffer area and has pushed down all records necessary to accommodate the insertion, INSERT checks the status of the $NONRESTOR flag. Normally, this flag is OFF (binary 0); INSERT reads back into memory the block containing the newly inserted record and makes this record available. However, if the programmer intends to do a random access next and does not want software to restore into memory this newly inserted record (thereby saving execution time), he may set the $NONRESTOR flag ON (not binary 0) before he codes the INSERT. INSERT then bypasses the restoration of this record but sets the $NONRESTOR flag OFF for future insertion.

FileReference.$NULLFLAG

This tag refers to a 1-character field (B type) that contains information regarding the latest block of data read from a chained file. Reference to this field enables the programmer to minimize wasted space within the area allocated to a chained file. See the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "GET Instructions" for an explanation of the contents of this field.

FileReference.$CARDFLAG

This tag refers to a 1-character flag (B type) associated with each CRAM file. Normally this flag is set to binary zero indicating that CRAM cards accessed are to be released only after updating. When CRAM cards are not to be updated after reading, the programmer can save execution time by requesting their immediate release from the capstan. If the current card is not to be updated, the user may request the release of the current card only by setting the FileReference.$CARDFLAG to binary 2. If none of the cards accessed is to be updated, the user may request the release of all cards by setting the flag to binary 4.

FileReference.$BINARYTBL

This tag refers to an area associated with each disc and CRAM file. Before the execution of a random-access READ or WRITE instruction, the programmer must update this area, directly or indirectly, with the relative address of the desired block (see the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 3, "Level Two Supplement," for further details).

The BINARYTBL area associated with a disc file contains the following fields: FileReference.$SECTION, FileReference.$RFLAG, and FileReference.$SECTOR.

FileReference.$SECTION is a 1-character field (B type) containing the number of the section to be accessed.

FileReference.$RFLAG, a 1-character field (B type), is the random flag. Normally this flag is set to binary 1 to indicate that the next block in the file is to be accessed. The user must set this flag to binary zero each time this series of fields is reset for a random access.

FileReference.$SECTOR is a 2-character field (B type) containing the relative number of the sector to be accessed with the section specified by FileReference.$SECTION.

The BINARYTBL area associated with a CRAM file contains the following fields: FileReference.$SECTION, FileReference.$CARD, and FileReference.$TRACK.

FileReference.$SECTION is a 1-character field (B type) containing the number of the section to be accessed.

FileReference.$CARD is a 2-character field (B type) containing the relative number of the card to be accessed within the section specified in the FileReference.$SECTION field.

FileReference.$TRACK is a 1-character field (B type) which contains the relative number of the track to be accessed on the card specified in the FileReference.$CARD field.

The following fields are compiled for each paper tape file in the user's program. FileReference.$BADCHAR and FileReference.$REPLACHAR are also compiled for each punched card file.

FileReference.$BADCHAR

This field contains a binary count of the invalid characters encountered during translation of each record when reading punched paper tape or punched cards.

FileReference.$ORIGINATE

This field contains the control characters taken from input paper tape if record terminators were defined as preceding the records.

FileReference.$TERMINATE

This field contains the control characters taken from input paper tape if record terminators were defined as following the records.

FileReference.$RLENGTH

This field may be set to a binary value that alters the maximum record length designated on the file specification sheet to reflect the anticipated length of the next record. The field is restored to its original value prior to termination of the GET or PUT instruction. If this field is set to zero, the next record read by the GET instruction is ignored.

FileReference.$FILSTATUS

When the user detects end-of-media from information in the previous record (rather than detection of the end-of-media character), he may set this flag to binary 2, which calls Extremity to alternate sections of the file; or he may set it to binary 4, which calls Extremity to transfer control to the user's own source file routine. On outputting, if section alternation may occur after any record, the software automatically sets this flag to 2 when the impending end-of-media condition is detected.

FileReference.$ERRTYPE

The user may access this field to determine the type of error encountered by the Paper Tape Translation routine.

FileReference.$REPLACHAR

When an invalid character is detected in punched card or paper tape input, the user may set this field to any other internal character representation except hex FE (the configuraiton to which all undefined characters are set).

FileReference.$PTFLAG

When an excess record length condition is detected by the Paper Tape Translation routine, the user may set this flag to binary zero, which causes the excess characters to be ignored.

FileReference.$EXCEPCHAR

When a control character is found in an output record and no escape character has been defined in the code set, the user may move this field (which contains the control character) to $REPLACHAR, which retains the control character in the output record.

FileReference.$CHARCOUNT

This field contains a count of the data characters read into the designated workarea.

FileReference.$LSTLINE

This field contains the number of the last data line to be printed.

FileReference.$CURLNUM

This field contains the number of the current line to be printed.

FileReference.$LSTLINEXX

This field is used when a report is being printed from an interim file. This field contains the number of the last data line to be printed and the sequential number of the report, designated in XX. The sequential number, which is assigned by the software for use in the file table, corresponds to the order in which the report's associated printer file specification is input.

FileReference.$CURLNUMXX

This field is used when a report is being printed from an interim file. This field contains the number of the current line being printed and the sequential number of the report, designated in XX. The sequential number, which is assigned by the software for use in the file table, corresponds to the order in which the report's associated printer file specification is input.

TABLE—ORIENTED SYSTEM TAGS

TableReference.$TEMLEN

This field contains the length of an item in a table.

TableReference.$TOFFSET

This field contains the relative location of the first item in a table.

CONVERTING HEXADECIMAL CODE TO NCR CENTURY INTERNAL CODE

HEXADECIMAL REPRESENTATION OF NCR CENTURY DATA CHARACTERS

B_8–B_5 \ B_4–B_1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000 0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001 1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010 2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011 3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100 4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101 5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110 6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111 7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

To find the two proper hexadecimal characters for representing an NCR Century data character, locate the desired character in its non-shaded square on the above chart. The shaded box to the left of the data character (on same line) contains the left-hand hexadecimal character, and the shaded box above the data character (same column) contains the right-hand hexadecimal character.

Following are some examples of NCR Century data characters and their corresponding hexadecimal representation.

NUL = 00 | + = 2B | R = 52

SYMBOLIC UNIT DESIGNATORS

PERIPHERAL TYPE	SYMBOLIC UNIT DESIGNATORS
Online Adapters	A00 - AFF
CRAM	C01 - C99
Disc	D01 - D99
I/O Writer	I01 - I99
Magnetic Tape	M01 - M99
Printer	P01 - P09
Card Reader	P11 - P19
Card Punch	P21 - P29
Paper Tape Reader	P31 - P39
Paper Tape Punch	P41 - P49
Optical Character Reader	P51 - P59
MICR	P61 - P69
Calcomp Plotter	P71 - P79
736 Encoder	R01 - R99

NCR CENTURY EXECUTIVES

INTRODUCTION

The NCR Century operating systems software is modular in design and resides both on disc and in memory. The majority of the software is disc resident to save memory space.

- The memory-resident portion of the I/O Executive (resident executive) handles all normal operating functions such as setting up and executing I/O operations and normal verification of completed I/O functions. In addition, the resident executive supervises linkage between subroutines within the user's program and the system software.
- The disc-resident portion is read into memory only when needed by the resident executive. However, when larger memories are used, some of the more frequently used subroutines become memory resident, providing faster program execution.

Several operating systems, each containing one or more resident executive formats, are available to the NCR Century series. The user informs Monitor which available resident executive format is to be used. The various operating systems and their resident executives are listed below.

BASIC OPERATING SYSTEM (B1)

The basic operating system, B1, contains an input-output Executive, Monitor, a System Disc Log and Disc Management. The B2 and B3 systems, which provide online communications and multiprogramming support respectively, are extensions of the basic B1 system. The B1 system contains the following resident executives:

- B1-1 is the normal resident executive for the NCR Century 100. This executive occupies 4.0K of memory and has 16 control words available for I/O operations.
- B1-2 is the normal resident executive for the NCR Century 200 series. B1-2 occupies 5.5K of memory, and it has 32 control words available for I/O operations. This executive is loaded in the NCR Century 200 unless there is insufficient space for the executive and the user's program, in which case Monitor will load the B1-2A resident executive.
- B1-2A, a smaller version of B1-2, is loaded by Monitor when there is insufficient space in the NCR Century 200 for B1-2. The B1-2A resident executive occupies only 4.0K of memory and has 16 control words.

ONLINE OPERATING SYSTEM (S2/B2)

The online operating system, B2, contains all the features of the basic B1 system, plus the following resident executives:

- The B2 resident executive is a dedicated online or dual programming executive for use on the NCR Century 100 or 200. It will operate in a minimum memory configuration of 16K and occupies 8K to 11K of memory. The B2 executive has 32 to 96 control words, and the following online features:

  - Tasking - the ability to run individual portions of the user's program concurrently.
  - Queueing - the ability to stack requests for service on a list to await program action.
  - Dynamic Storage Allocation - the ability to dynamically allocate storage areas in memory from a central pool for use as I/O areas or work areas.
  - Dual Programming - the ability to process both an online and a batch program concurrently.
  - Chaining - the method of grouping together related memory areas, although these areas may be scattered throughout memory.
  - Unsolicited Input - the ability to input messages to the online program or the operating system through the I/O writer.

  Tasking, dual programming, and the number of control words used determines the physical size of the B2 resident executive.

- B2-2A is a dedicated online resident executive for the NCR Century 200. This executive is a B2 resident executive with restrictions which provide minimum software requirements for online operations. B2-2A occupies only 5.5K of memory and it will run in any memory size from 32K up. This executive has 16 control words, and has the following online features: queueing, dynamic storage allocation, chaining, unsolicited input.
- S2 is a dedicated online resident executive for the NCR Century 100 with a minimum memory size of 16K. S2 occupies 5.5K of memory, has 16 control words available and has the following online features: queueing, chaining, unsolicited input.

MULTIPROGRAMMING SYSTEM (B3)

The B3 operating system provides multiprogramming capability for the medium and large scale NCR Century 200 configurations that have the multiprogramming hardware options. The B3 system is a fixed partition system, providing for as many as nine programs to operate independently in separate partitions, sharing central processor time. Each partition has available its own basic B1 or B2 resident executive, which requires a minimum memory configuration of 16K. However, when partition sizes over 16K are required, they must be in multiples of 2K.

At the start of the day the B3 initializer divides memory into fixed partitions, with sizes designated by the user. The B3 initializer allocates peripherals to the partitions, and assigns priorities to the partitions in the order in which they are presented. The initializer then overlays itself with the B3 resident executive (Kernel and Satellite). The B3 operating system has only one resident executive, described below:

- The B3 resident executive occupies from 16K to 32K and requires a minimum memory configuration of 64K (the B3 executive occupies the first 16K leaving 48K for three 16K partitions). Included in the B3 executive are the B3 Kernel and the B3 Satellite:
  - The B3 Kernel controls overall operation of the multiprogramming environment.
  - The B3 Satellite controls operator communications and printer spooling. By creating a file on disc for output to the printer, spooling allows multiple partitions to share a common printer without any delay or interruption of the individual partition's operations.

OPERATING SYSTEM	RESIDENT EXECUTIVE	PROCESSOR	NUMBER OF CONTROL WORDS	MEMORY OCCUPIED BY EXECUTIVE	IDENTIFICATION NUMBER (POS.51-52 ON COMPILER SPEC. SHEET)
B1 Basic	B1-1	100	16	4K	00
	B1-2	200	32	5.5K	40
	B1-2A	200	16	4K	08
B2 Online	B2	100 or 200	32 to 96	8K to 11K	18 (C-100) 58 (C-200)
	B2-2A	200	16	5.5K	50
	S2	100	16	5.5K	10
B3 Multi-Programming	B3	200	32	16K to 32K	80

SUBJECT INDEX FOR NCR CENTURY SERIES NEAT/3

NCR
WORLDWIDE HEADQUARTERS
DAYTON, OHIO 45409

EP-9880-01 [988001] 0971