HONEYWELL EDP

HARDWARE BULLETIN

# SERIES 200

## SERIES 200 PROGRAMMERS' REFERENCE MANUAL, MODELS 200/1200/2200 ADDENDUM #2

| | |
|---|---|
| SUBJECT: | Description of New Language Features Available to Users of Easycoder Assembler D. |
| SPECIAL INSTRUCTIONS: | This bulletin is an addendum to the Series 200 Programmers' Reference Manual, Models 200/1200/2200 (file number 113.0005.0000.00.00). The information contained herein will be incorporated into the manual at the time of the next revision. |
| DATE: November 20, 1965 | FILE NO. 113.0005.0000.00.00 |

8701
31165
5166
Printed in U. S. A.

## FOREWORD

An additional assembly program is available to the Series 200 user. Called Easycoder Assembler D, it is part of the SERIES 200/OPERATING SYSTEM - MOD 1. Easycoder D operates in a system having a minimum of 16,384 characters of main memory. (Additional memory may be used to advantage.)

All information pertaining to Easycoder Assembler C and contained either in the Series 200 Programmers' Reference Manual, Models 200/1200/2200 or in Addendum No. 1 to that manual pertains also to Easycoder D. The additional features offered the Series 200 user by Easycoder Assembler D lie mainly in two areas:

1. Easycoder D processes symbolic tags of from one to ten characters in length;
2. Easycoder D processes floating-point constants and the machine instructions provided by the scientific unit (Feature 1100).

Questions and comments regarding this publication should be addressed to:

Honeywell Electronic Data Processing
Information Services
60 Walnut Street
Wellesley Hills, Massachusetts 02181

# ADDENDUM TO SERIES 200 PROGRAMMERS' REFERENCE MANUAL

## EASYCODER PROGRAMMING

The following information applies to Section 5 of the reference manual.

### Coding Form (Alternate Format)

Symbolic tags of up to ten characters in length may be employed with Easycoder Assembler D. For symbolic tags consisting of six characters or less, the standard coding form is used. However, if tags of from seven to ten characters are used, the location, operation code, and operands fields must be altered (to accommodate this increase in tag size).

To denote that the alternate format is used, the program header (PROG) card, which is always coded in the standard format, must contain an A in column 75. All cards following the program header, up to and including the END card, must have the following format:

| FIELD | CARD COLUMNS |
|---|---|
| Card Number | 1-5 (unchanged) |
| Type | 6 (unchanged) |
| Mark | 7 (unchanged) |
| Location | 8-18 |
| Operation Code | 19-24 |
| Operands | 25-80 |

1. LOCATION (Card Columns 8-14): This field is changed to card columns 8-18 when the alternate format is specified. The same programming conventions which apply to six-character tags apply also to ten-character tags.
2. OPERATION CODE (Card Columns 15-20): This field is changed to card columns 19-24 when the alternate format is specified. The method of coding mnemonic operation codes remains the same. If octal operation codes are used, they are written in columns 23 and 24; columns 19 to 22 are left blank.
3. OPERANDS (Card Columns 21-80): This field is changed to card columns 25-80 when the alternate format is specified. The method of coding entries and remarks remains the same.

NOTE: Symbolic tags of more than six characters in length may not be used if the input is to be in the form of paper tape.

### Address Modification Codes

INDEXED

Indexed addressing is performed by appending to the address being modified a code which specifies the index register to be used. When an index register in the range from 1 through

15[1] is to be used, the code consists of a plus sign followed by an X and the decimal number of the desired index register. An index register in the range from 16 through 30[2] is referred to by a code consisting of a plus sign followed by a Y and the decimal number of the desired index register minus 15, as follows:

| Index Register | Code |
|---|---|
| 16 | +Y1 |
| 17 | +Y2 |
| 18 | +Y3 |
| ⋮ | ⋮ |
| 29 | +Y14 |
| 30 | +Y15 |

## DATA FORMATTING STATEMENTS

The following information applies to Section 6 of the reference manual. With the exception of floating-point constants (Easycoder D only), this information applies to Easycoder C as well as to Easycoder D.

### Numeric Constants

Numeric constants may take any one of three forms: decimal, binary, or octal. The maximum length of the storage field which can be occupied by a numeric constant is 63 characters.

#### DECIMAL CONSTANTS

The coding of decimal constants remains as stated in the reference manual.

#### BINARY CONSTANTS

A binary constant is actually written as a decimal entry which is then automatically converted to a binary value by the Assembly Program. The binary value is stored (right-justified) in the constant field.

To code a binary constant, the programmer writes the following: (1) a # sign (in the first column of the operands field); (2) a number from 1 to 63 which designates the number of six-bit characters needed to store the resulting binary value; (3) the letter B; (4) the decimal representation of the desired binary constant. Note that if the decimal representation of the

[1] Recall that index registers 7 through 15 are available only with the following central processors: 201-1, 201-2, 1201, 2201, and 4201.

[2] Index registers 16-30 are available only with the following central processors: 1201, 2201, and 4201.

binary constant is preceded by a minus sign, the Assembly Program stores the binary constant in twos-complement form.

The maximum value allowed for a binary constant is 999999.

**EASYCODER**
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

| | CARD NUMBER | TYPE | MARK | LOCATION | OPERATION CODE | OPERANDS | |
|---|---|---|---|---|---|---|---|
| | 1 2 3 4 5 | 6 | 7 | 8 14 | 15 20 | 21 62 | 63 80 |
| 1 | | | | CON1 | DCW | #1B5Ø | |
| 2 | | | | CON2 | DCW | #3B-5 | |
| 3 | | | | CON3 | DCW | #63B1 | |
| 4 | | | | | | | |

The first statement above results in a one-character constant with an octal value of 62. The result of the second statement is a three-character constant with an octal value of 777773. The last statement results in a 63-character constant which has a 1 in its rightmost position preceded by 62 zeros.

## OCTAL CONSTANTS

Octal constants are coded in octal notation. To code an octal constant the programmer writes the following: (1) a # sign (in the first column of the operands field); (2) a number from 1 to 63 which specifies the number of six-bit characters required to store the octal constant; (3) the letter C; (4) the constant value. Note that the value stored by the Assembly Program is always left-justified in the storage field.

**EASYCODER**
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

| | CARD NUMBER | TYPE | MARK | LOCATION | OPERATION CODE | OPERANDS | |
|---|---|---|---|---|---|---|---|
| | 1 2 3 4 5 | 6 | 7 | 8 14 | 15 20 | 21 62 | 63 80 |
| 1 | | | | OCT1 | DCW | #2C7777 | |
| 2 | | | | OCT2 | DCW | #63C6 | |
| 3 | | | | | | | |

The first statement above results in a two-character constant with an octal value of 7777. The second statement results in a 63-character constant composed of octal 60 followed by 62 zeros.

## Alphanumeric Constants

The three methods of coding alphanumeric constants remain as stated in the reference manual. However, it should be noted that the maximum number of alphanumeric characters which can be contained in the constant depends on the number of card columns available in the operand field to express the constant.

The first method (constant value enclosed in @ symbols) and the second method (constant value enclosed in legal characters[1]) require two card columns to format the constant. Therefore 58 card columns are available (54 if using the alternate card format) to express the constant. In these cases the alphanumeric constant may contain 58 (or 54) alphanumeric characters.

The third method requires a maximum of four card columns to format the constant. The four columns are occupied by the # sign, and the two digits which specify the maximum number of alphanumeric characters contained in the constant, and the letter A. Therefore 56 card columns are available (52 if using the alternate card format) to express the constant. Using this method, the alphanumeric constant may contain 56 (or 52) alphanumeric characters.

## Blank Constants

The manner of coding blank constants remains the same. However, the decimal value which indicates the number of blank storage positions desired may be from 1 to 63.

## Floating-Point Constants[2]

A floating-point constant is written as a decimal entry which is then automatically converted by the Assembly Program to a fixed-length floating-point value, i. e. , a six-character binary mantissa followed by a two-character power-of-two exponent.

To code a floating-point constant the programmer writes the following:

1. The letter F.
2. A decimal number, the mantissa, which may be signed or unsigned and which may contain a maximum of eleven digits with or without a decimal point.
3. The letter E.
4. A decimal number, the exponent, which must be between 0 and 616 and may be signed or unsigned.

If an exponent of zero is desired, the letter E and the decimal number which follows it are not required.

NOTE: If the mantissa and/or the exponent is preceded by a minus sign, the Assembly Program stores the corresponding value in twos-complement form.

[1]Legal characters are any characters other than blank, +, -, #, F, or the numerics 0-9.

[2]Floating-Point Constants may be used only with the scientific unit (Feature 1100). See the bulletin entitled Scientific Unit for Models 1200 and 2200 (Feature 1100).

**EASYCODER**
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

| | CARD NUMBER 1 2 3 4 5 | TYPE 6 | MARK 7 | LOCATION 8–14 | OPERATION CODE 15–20 | OPERANDS 21–62 | 63–80 |
|---|---|---|---|---|---|---|---|
| 1 | | | | FCON1 | DCW | F4-359E2 | |
| 2 | | | | FCON2 | DCW | F+4359E-1 | |
| 3 | | | | FCON3 | DCW | F1 | |
| 4 | | | | FCON4 | DCW | F-.ØØØ1 | |
| 5 | | | | FCON5 | DCW | F-1E-4 | |
| 6 | | | | | | | |

The first two entries above (FCON1 and FCON2) result in the same floating-point value when converted by the Assembly Program. FCON1 uses a decimal point while FCON2 arrives at the same result by using a negative exponent. This is also true for FCON4 and FCON5.

## ASSEMBLY CONTROL STATEMENTS

The following information applies to Section 7 of the reference manual. Four statements (PROG, ADMODE, CEQU, and REP) are modified somewhat; one new statement (SETLIN) is also described. Information pertaining to the ADMODE, CEQU, REP, and SETLIN statements is common to Easycoder C as well as to Easycoder D. The information concerning the PROG statement is applicable only to Easycoder D.

### Program Header - PROG

If the programmer desires to use the alternate card format (which allows room for tags consisting of up to ten characters), column 75 of the program header card must contain the letter A. The PROG card is never coded in the alternate format: the letters PROG always appear in the op code field (columns 15 through 18), while the name of the program always appears beginning in column 21.

### Set Address Mode - ADMODE

The letters ADMODE are placed in the op code field. The operands field specifies the mode of address assembly and may contain either the numbers 2, 3, or 4, or a symbolic tag. If a symbolic tag is used, it must have been previously defined to have a value of 2, 3, or 4. The contents of the operands field specifies that all subsequent instructions are to be assembled in the 2-, 3-, or 4-character addressing mode.

### Control Equals - CEQU

The CEQU statement assigns the symbolic tag written in the location field to the value written in the operands field. Instructions which refer to the tag defined by the CEQU statement must not precede the CEQU statement.

The entry in the operands field must be a decimal, binary, octal, or alphanumeric constant (the octal format is most commonly used). Regardless of the constant used, however, the resultant value must not exceed four characters in length.

## Repeat - REP

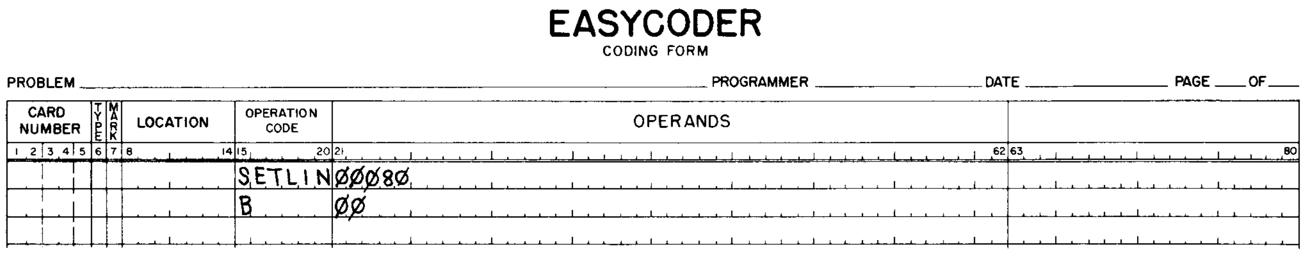
This statement directs the Assembly Program to repeat the following data formatting statement or machine instruction the number of times specified in the operands field. The number of times a statement is repeated includes the original statement and may not exceed 63. The Assembly Program repeats the statement without variation.

The letters REP are written in the op code field. The operands field designates the number of times the following statement is to be repeated (including the original statement).

## Set Line Number - SETLIN

This instruction is used to control the generation of line numbers by the Assembly Program.

The letters SETLIN are written in the op code field, while the first five columns of the operand field contain the desired line number. The Assembly Program replaces the contents of the line number generation counter with the number in the operands field of the SETLIN statement. This statement is effective only when the Assembly Program is generating line numbers. It is important to note that all of the first five columns in the operands field must be punched with a decimal number (i. e. , leading zeros are required).


**EASYCODER**
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

| | CARD NUMBER | TYPE | MARK | LOCATION | OPERATION CODE | OPERANDS | |
|---|---|---|---|---|---|---|---|
| | 1 2 3 4 5 | 6 | 7 | 8 14 | 15 20 | 21 62 | 63 80 |
| 1 | | | | | SETLIN | ØØØ8Ø | |
| 2 | | | | | B | ØØ | |
| 3 | | | | | | | |

In the example above, the SETLIN statement causes the instruction which follows it (B/00) to be assigned a line number of 00080.

## INSTRUCTIONS

The following information applies to Section 8 of the reference manual.

All machine instructions except LIB (Load Index/ Barricade Indicator), SIB (Store Index/ Barricade Indicator), and those associated with the scientific unit (Feature 1100) can be processed by Easycoder Assemblers A, B, and C. Easycoder D accepts all instructions.

CRP