HONEYWELL EDP

HARDWARE BULLETIN

SERIES 200

ADDENDUM # 1 TO SERIES 200 PROGRAMMERS' REFERENCE MANUAL, MODELS 200/1200/2200.

SUBJECT:	Description of New Language Features Available to Users of Easycoder Assembler C.
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: September 17, 1965 FILE NO. 113.0005.0000.00.00

8477
10965
Printed in U.S.A.

ADDENDUM NO. 1 TO SERIES 200 PROGRAMMERS' REFERENCE MANUAL, MODELS 200/1200/2200

EASYCODER PROGRAMMING

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

Coding Form

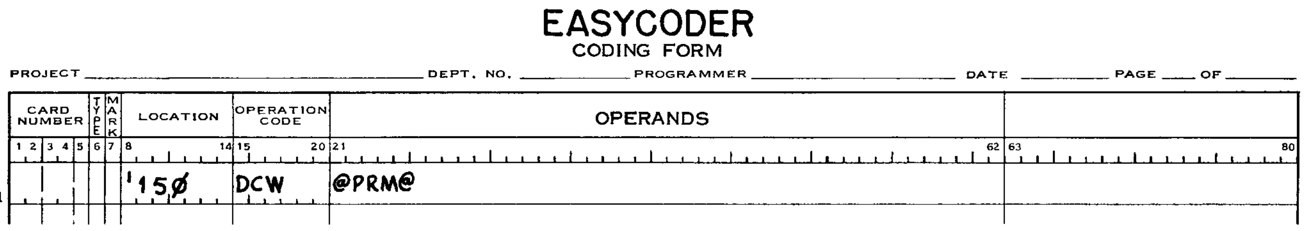
Several new options in the use of the coding form are available to Easycoder C users. These options are as follows.

1. TYPE (Card Column 6): This column may also contain the letter T to designate a temporary remarks card, or the letter D to designate a data card. If the programmer wishes to enter remarks lines anywhere in the source program but does not want these remarks to become a permanent part of the source program, a T instead of an asterisk (*) is placed in column 6. Remarks lines inserted in this manner are used only on the first assembly (i.e., when the program is being "inserted") and are subsequently deleted from the symbolic program tape by the Assembly Program. A temporary (like a permanent) remarks statement, while it appears in the program listing, does not appear in the object program.

   A D in column 6 indicates a data card. All data cards must be contained in segments consisting only of data cards. In addition, any data card (or group of data cards) must be immediately preceded by a SEG card and immediately followed by either an EX, XFR, or END card. When a data card is encountered by the Assembly Program, the contents of columns 8 through 80 are reproduced, unaltered, on the binary run tape or machine-language punched deck.

2. LOCATION (Card Columns 8-14): The location field may also contain an apostrophe (')[1] followed by a decimal number to indicate an address relative to the out-of-sequence base (OSB). The out-of-sequence base is a value, maintained by the Assembly Program, which can be set by means of the XBASE instruction (see page 5). The Assembly Program assigns to the corresponding statement an address equal to the sum of the decimal number and the current value of the OSB. (Leading zeros may be omitted from the decimal number.) The allocation of any succeeding instructions is not affected.

   Assume, for example, that the OSB has been set to the value 500 by an XBASE instruction when the following DCW statement is encountered.



EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ 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
				'15Ø	DCW	@PRM@	

[1] Card code 8, 2 (octal 12).

The constant PRM is assigned, by the Assembly Program, to locations 648 through 650. (The value of the OSB remains 500.)

NOTE: If the apostrophe appears in column 8, an out-of-sequence address refers to the leftmost character of an instruction or to the rightmost character of a constant or reserved area. If the apostrophe appears in column 9, these conventions are reversed: An out-of-sequence address refers to the rightmost character of an instruction or to the leftmost character of a constant or reserved area.

3. OPERANDS (Card Columns 21-80): For Easycoder C users, the operands field extends to column 80. However, the method of coding entries in this field remains the same. Remarks may be entered following the terminating space.

Address Codes

The valid address codes also include the special symbol apostrophe (printer '; keypunch 8, 2; octal 12). This symbol is an element whose value is equal to the current value of the out-of-sequence base (OSB). It is followed by an address modifier to specify the address of the desired operand. The OSB is set by means of the XBASE instruction (see page 5).

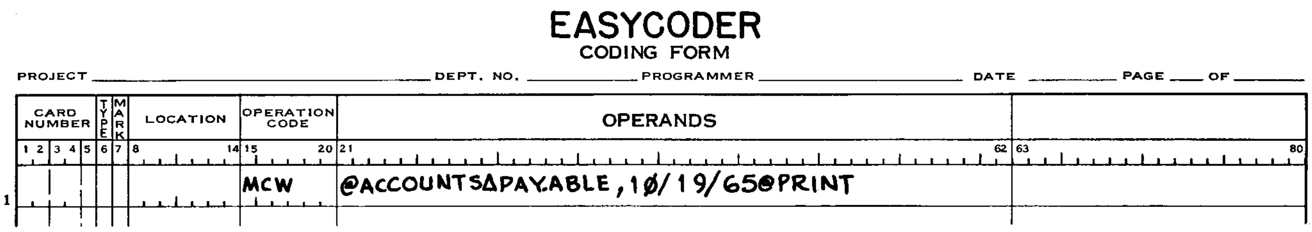
Literals

All literals (including binary) can be coded with a maximum of 63 characters. If the constant generated from a literal occupies from one to six storage locations, it is assigned a storage address only once in the program, regardless of the number of times the literal appears in the program. A constant that exceeds six characters is assigned a storage address each time the corresponding literal appears in the source program. The latter condition can be avoided by using a DCW statement whenever a long literal is to be used more than once in the source program.

ALPHANUMERIC LITERALS

Alphanumeric literals may be written in one of four ways.

1. The special symbol @ is written before and after the literal. The alphanumeric literal may contain any valid Series 200 character (including blanks) except the @ symbol.



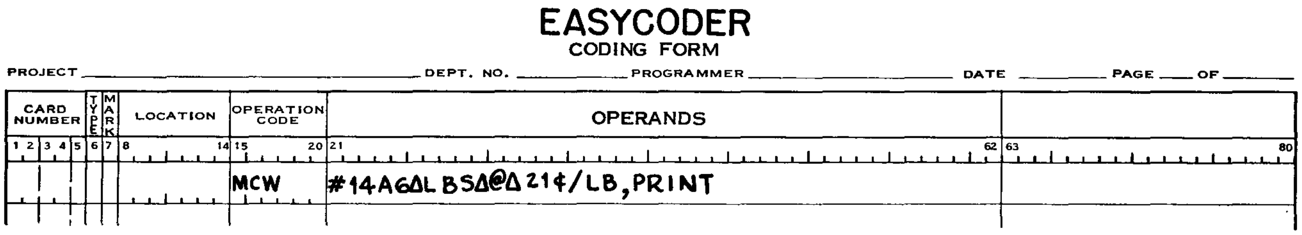
EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ______

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				MCW	@ACCOUNTSΔPAYABLE,1Ø/19/65@PRINT	

The statement above illustrates the use of an alphanumeric literal. The instruction causes the information contained within the @ symbols to be moved to the field tagged PRINT.

2. A number sign (#) is followed by a number from 1 through 63 which specifies the number of characters in the literal; this number is, in turn, followed by the letter A and the literal.



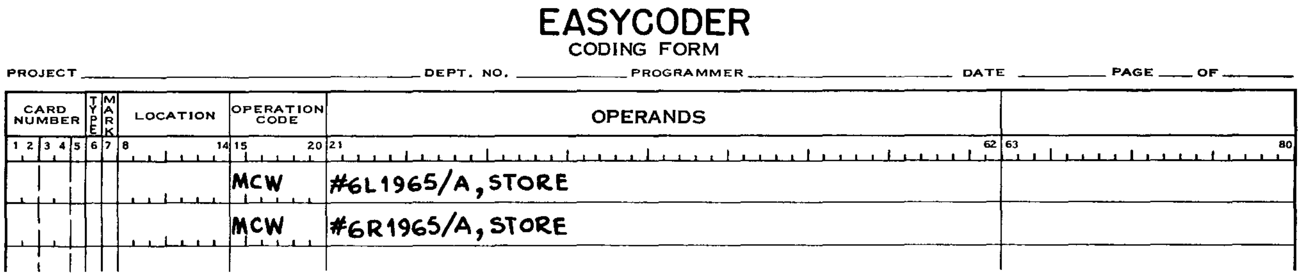
EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ 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					MCW	#14A6ΔLBSΔ@Δ21¢/LB,PRINT	

In the above example there are 14 characters in the literal. The instruction causes these 14 characters to be moved to the field tagged PRINT.

3. If it is desired to set an <u>item mark</u> (in addition to a work mark) in the <u>leftmost</u> position of the literal constant field, a number sign (#) is followed by a number from 1 through 63 which specifies the number of characters in the literal; following this number is the letter L and the literal (see the first example below).



EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ 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					MCW	#6L1965/A,STORE	
2					MCW	#6R1965/A,STORE	

4. If it is desired to set an <u>item mark</u> in the <u>rightmost</u> position of the literal constant field, a number sign (#) is followed by a number from 1 through 63 which specifies the number of characters in the literal; following this number is the letter R and the literal (see the second example above).

NOTE: In forms (1) and (2) alphanumeric literals of six characters or less are stored in a literal table and duplicates are eliminated. The duplicates are not, however, eliminated in forms (3) and (4).

DATA FORMATTING STATEMENTS

The following information applies to Section 6 of the reference manual.

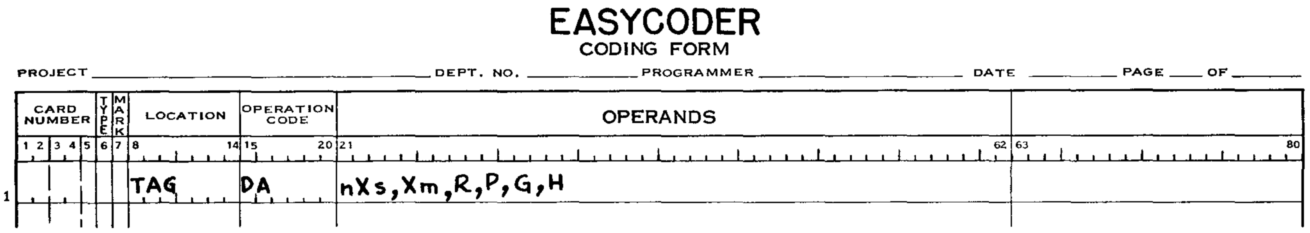
Define Area - DA

When used with Easycoder C, the DA statement may make use of the following parameters (in addition to the n, s, Xm, and R parameters already specified).

1. The character P: Coding this character in the heading line of a DA statement causes the special character 72_8, together with an item mark, to be placed at the end of each area as an additional character.
2. The character G: Coding this character in the heading line causes the special character 32_8, together with a record mark, to be placed one position to the right of the last area.
3. The character H: Coding this character in the heading line instructs the Assembly Program to associate the specified index register (Xm) with each reference to the tag in the location field of the DA statement, as well as with each reference to a field or subfield within the reserved area(s).

   NOTE: If a symbolic tag is used it is not automatically indexed by the specified index register (Xm) unless parameter H is employed. This parameter is meaningless if no index register is specified.

The format of a DA statement heading line employing all parameters is illustrated below.



EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ 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			TAG	DA	nXs,Xm,R,P,G,H	

ASSEMBLY CONTROL STATEMENTS

The following information applies to Section 7 of the reference manual. Five statements (SEG, EX, ORG, MORG, and LITORG) are modified somewhat; two new statements (XFR and XBASE) are also described.

Segment Header - SEG

If used, a SEG statement must follow the program header (PROG), each Execute (EX) statement, and each Transfer (XFR) statement.

Execute - EX

Each segment except the last must end with either an EX or an XFR statement. When an EX statement is encountered, all literals preceding the EX statement which have not been allocated to memory are allocated in sequence, and the literal table is cleared.

Origin - ORG

A symbolic tag may be written in the location field. If this tag begins in column 8, it is assigned to the address written in the operands field. If it begins in column 9, the tag is assigned to the location at which the next instruction would have begun had the ORG statement not been present.

EASYCODER
CODING FORM

PROJECT ______________ DEPT. NO. ________ 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				IDENT	ORG	78ØØ	

In the example above, assume that the instruction preceding the ORG statement was assigned to locations 5000 through 5007. The next instruction would normally begin at location 5008. The tag IDENT, since it begins in column 9, is thus assigned to location 5008, and the next instruction is stored beginning at location 7800.

Modular Origin - MORG

A symbolic tag may be written in the location field. If this tag begins in column 8, it is assigned to the address written in the operands field. If it begins in column 9, the tag is assigned to the location at which the next instruction would have begun had the MORG statement not been present (seethe example given for the ORG statement).

Literal Origin - LITORG

In the absence of a LITORG statement, all of the generated coding associated with a memory load — except for a memory load terminated by an XFR statement — is allocated immediately following the in-line coding.

A symbolic tag may be written in the location field. If this tag begins in column 8, it is assigned to the address written in the operands field. If it begins in column 9, the tag is assigned to the location at which the next instruction would have begun had the LITORG statement not been present (see the example given for the ORG statement).

Transfer - XFR

This statement performs essentially the same functions as the Ex statement; the one exception is that use of the XFR statement does not result in the allocation of literals or in the clearing of the literal table.

Set Out-of-Sequence Base - XBASE

The XBASE statement establishes the out-of-sequence base (OSB). As its name implies, the OSB is a base address for the storage of out-of-sequence coding. Such coding is allocated or referred to by means of the address code ' (apostrophe) in the location or operands field, respectively (see above).

The letters XBASE are written in the op code field. The operands field contains the value (absolute or symbolic) to which the assembly program is directed to set the out-of-sequence base (OSB). If a symbolic tag appears in the operands field it must have appeared in the location field of a previous source program entry.

EASYCODER
CODING FORM

PROJECT ______ DEPT. NO. ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					XBASE	5ØØ	
2				'275	DCW	@CON@	

In the above example, the out-of-sequence base (OSB) is set to 500 by the XBASE statement. When the second entry is encountered, the Assembly Program assigns the rightmost character of the constant CON to location 775 (500 + 275).

HONEYWELL
ELECTRONIC
DATA
PROCESSING
WELLESLEY HILLS,
MASSACHUSETTS 02181