P-TYPE

PACKED DECIMAL DATA

Packed decimal data fields are created by specifying a “P” for the type in a DC or DS declarative. In this format, when assembled, each byte in the field contains 2 decimal digits except the right-most byte in the field which contains a decimal digit followed by a sign. The valid positive signs are the hexadecimal characters A, C, E, and F. The valid negative signs are the hexadecimal characters B and D. The “preferred” signs, those which are generated by the machine or chosen by the assembler, are positive “C” and negative “D”. It is standard practice to use “C” and “D”, but be aware that the other signs are recognized as well. Consider the two fields defined below,

```
AMOUNT1      DC    PL3’+20’
AMOUNT2      DC    PL2’-34’
```

In the first example the assembler generates x’00020C’ for AMOUNT1 and x’034D’ for AMOUNT2.

A packed decimal field can be created using either of the following formats,

```
          name       DC     dPLn’constant’
   or
          name       DS     dPLn
```

where ‘name’ is an optional field name
‘d’ is a duplication factor used to create consecutive copies of the field (default = 1copy)
‘P’ represents the packed decimal data type
‘L’ represents an optional length (default = 1 byte)
‘n’ is the number of bytes in the field
‘constant’ is the initial value of the field in decimal format. Optionally, a decimal point and a + or - sign can be coded but the decimal point is not assembled.

It is important to note that if a name for a field is specified, it will represent the address of the first byte of the field. Additionally, the name of the field will be associated with a length attribute which equals the number of bytes in the field. If the “Ln” construction is omitted in a DC, the length will be just large enough to accommodate the digits specified in the constant (along with a padded 0 on the left if the number of digits in the constant is even). In a DS declarative, “Ln” must be coded. The length of a packed field defined using a DC or DS is limited to a maximum of 16 bytes. This limits the size of decimal fields to 31 digits.

During assembly, fields that were defined consecutively in the source with DC’s or DS’s are assigned consecutive storage locations in the program according to the value of the location counter which is maintained by the assembler. For example, in the fields below, if PKD1 is associated with address x’1000’, then PKD2 is located at x’1008’ and PKD3 is located at x’1011’.

LOCATION

```
1000       PKD1      DS     PL8
1008       PKD2      DC     PL9’20’
```

```
         1011       PKD3     DS    PL3
```

Examples

Some Typical DS's and DC's:

```
   R        DS    PL8    An 8-byte field reserved for packed data
   S        DS    PL16   The maximum size of a packed field (31 digits)
   T        DS    PL1    The minimum size of a packed field (1 digit)
   U        DS   3PL4    Three consecutive packed fields (U refers to
                         the first field)
   V        DC    P’123’    V = x’123C’, field is assumed positive
   W        DC    P’-123’   W = x’123D’
   X        DC    P’12’     X = x’012C’, field padded with a 0 to fill
                            up 2 bytes
   Y        DC    PL4’12’   Y = x’0000012C’, constant shorter than the
                            field length - padded
   Z        DC    PL2’12345’   Z = x’345C’, constant too big,
truncation
                               occurs on the left
   DOG      DC    P’123.45’  DOG = x’12345C’, decimal is treated as a
                             comment, not assembled
```

Tips

1. Be sure and use DC instead of DS when providing a constant. Consider the example below,

```
                   AMOUNT     DS     PL4’1234’
```

The assembler will not complain about you providing a constant on a DS statement. It will, however, ignore the constant and treat it as a comment. The field remains uninitialized.

2. Be sure to make any packed fields you define large enough to hold any possible values that might be developed in them. You cannot err by overestimating the size of a packed field. On the other hand, it is disastrous to underestimate the size of a packed field - this will result in truncation of digits in arithmetic operations (decimal overflow exception).

3. The specification of “P” as the type of data does not ensure that the data is packed. In fact, many kinds of data might be stored in a packed field.