Tech Series

Working With Binary Integer Data in REXX

This article will provide an example of working with a file that contains [at least] one binary integer field. The assumption is made that you know how to read and write records, using record I/O (EXECIO) or stream I/O (linein, charin, linout, charout, and so on), so the focus here will be on the actual processing of binary numbers.

Binary integer data is numeric data stored as powers of two, typically in 2 bytes, 4 bytes, or 8 bytes.

Suppose there is an inventory file containing 100-byte records and that locations 41-44 contain the quantity on hand for an item in a warehouse. If there is a value there of X'00000032', this indicates that there are 50 of these items in stock. (X'32' = 3 X 16 + 2 in decimal).

Suppose that an ISPF dialog driven by a REXX exec takes online orders and that a customer places a request to buy some quantity of a particular item, say 12 Wacky Widgets. The REXX code needs to determine if there is an adequate number on hand to satisfy the customer's request. Assume that the quantity ordered is in the REXX variable named qty and the item description is in a field called desc.

The REXX EXEC is able to locate the relevant inventory record and read it into a variable called inrecord. The next step is to convert the data in positions 41-44 to REXX-type numeric data.

The tool to use here is the REXX built-in function named C2D ("Character-to-Decimal"). This function takes any string as input and treats the digits as a binary number, returning a decimal value. For example, C2D('b') would return the value 130, since a character 'b' is X'82'. In the inventory example, the conversion code would look something like this:

```
qty_on_hand = C2D(substr(inrecord,41,4))
```

This places a decimal 50 into the REXX variable named qty_on_hand (based upon a binary value of X'00000032'). The next thing that the REXX code would have to do is determine if there are enough Wacky Widgets to satisfy the customer's request. If there are enough to ship, the program would subtract the qty number from the just-converted qty_on_hand and then update the record on disk. If there is a shortage, a call to some other routine that deals with this contingency must be made. Sample REXX code might look like this:

continued next page

```
qty_on_hand = C2D(substr(inrecord,41,4))
if qty > qty_on_hand
then call under_stocked
else do
qty_on_hand = qty_on_hand - qty
inrecord = overlay(D2C(qty_on_hand,4),inrecord,41)
call process_order
end
```

Note the use of the D2C built-in function to convert a decimal number back to binary. D2C functions exactly opposite of C2D. Also note the use of the REXX overlay function to replace the binary data at positions 41-44 within inrecord.

C2D and D2C, as well as C2X discussed in the packed-decimal article, are somewhat counter-intuitive in how they work. It is important to test and experiment a lot when using them. But these are the functions that enable the processing of binary integer and packed-decimal fields in REXX EXECs.

This article was contributed by:

Steve Comstock
The Trainer's Friend, Inc.