Author(s) : J. Zoll
Library: KERNLIB
Submitter :
Submitted: 01.06.1973
Language : Fortran or Assembler
Revised: 16.09.1991

Pack/Unpack Continuous Byte-strings

PKCHAR allows packing of integers into *continuous byte-strings* on zoned memory across word boundaries. The term *continuous byte-string* is used here to designate n-bit bytes, stored from left to right, as opposed to the objects handled by PKBYT (M422), which are stored right to left within each word. The inverse unpacking is performed by UPKCH. Leading and trailing bits of each zone can be ignored.

Structure:

SUBROUTINE subprograms
User Entry Names: PKCHAR, UPKCH
External References: JBYT (M421), SBYT (M421), CBYT (M421)
COMMON Block Names and Lengths: /SLATE/ NWU,DUMMY(39)

Usage:

```
CALL PKCHAR(INT,MPK,N,IPAR)
CALL UPKCH (MPK,INT,N,IPAR)
```

PKCHAR packs the N–word vector INT of integers into the continuous byte-string supported by the vector MPK according to the packing specifications given in IPAR.

UPKCH is the exact inverse of PKCHAR.

The packing parameters are given in the 5-element vector IPAR:

IPAR(1)	Number of bits per byte, must be ≥ 1.
IPAR(2)	Number of bytes to be used in each zone (starting with the left-most); if IPAR(2) = 0, the maximum possible number is used.
IPAR(3)	Number of bits per zone. If IPAR(3) = 0, a zone equals 1 word.
IPAR(4)	Number of leading bits of each zone to be ignored.
IPAR(5)	Each new word handled by PKCHAR is preloaded with IPAR(5).

MPK is seen as a continuous string of bits, starting with the most significant bit of MPK(1), ignoring word boundaries. This string is divided into a number of consecutive and contiguous zones, each of IPAR(3) bits; the first zone starts with the most significant bit of MPK(1). Each zone contains IPAR(4) leading bits, a number of bytes (each of IPAR(1) bits), and trailing bits, if any.

On return from either routine, NWU in COMMON block /SLATE/ indicates the number of words in MPK actually used. PKCHAR sets to IPAR(5) each word of MPK before filling it, but it does not clear any trailing unused words which logically belong to the last zone.

Examples:

1. To convert, on the CDC 7600, 6-bit Hollerith text to 7-bit ASCII–code, to be held in 36-bit words on the PDP10, with 5 characters per word.

   ```
   DATA IPACK6 /6,0,0,0,0/
   DATA IPACK7 /7,0,36,0,0/
   CALL UPKCH(HOLL,INT,N,IPACK6)
   ```

   unpacks the Hollerith string `HOLL` into `INT`, where `INT(I)` is a small integer giving the display-code value of the `I`-th character. After conversion to ASCII, one may pack:

   ```
   CALL PKCHAR(INT,MPK,N,IPACK7)
   ```

   giving the vector `MPK` ready to be written out. If for some reason one required the first and the last (5th) character in each 36-bit PDP10 word to be zero, one could use:

   ```
   DATA IPACK7 /7,3,36,7,0/
   ```

2. To unpack 8-character bytes read with the CDC 7600 from 9-track tapes:

   ```
   DATA IPACK /8,0,120,0,0/
   CALL UPKCH(A,INT,N,IPACK)
   ```

3. To unpack on the CRAY 32-bit integers, read one each into one 64-bit machine word, into 16-bit integers, one each in one machine word, right-justified with zero-fill:

   ```
   DATA IPACK /16,2,0,32,0/
   CALL UPKCH(I32,I16,N,IPACK)
   ```

   The same operation on the Apollo, which has 32-bit words, could be done with

   ```
   DATA IPACK /16,0,0,0,0/
   ```

4. The Fortran implementaion of `BLOW` (M426) executes:

   ```
   IPACK(1) = NBITS
   IPACK(2) = 0
   IPACK(3) = NBYTES*NBITS + 127
   IPACK(4) = 0
   IPACK(5) = 0
   CALL UPKCH(SOURCE,TARGET,NBYTES,IPACK)
   ```

•