A FORTRAN SUBROUTINE FOR UNPACKING AND PACKING BINARY DATA

Gary W. Phillips

Naval Research Laboratory
Washington, D.C.

December 1974
A Fortran Subroutine for Unpacking and Packing Binary Data

GARY W. PHILLIPS

Nuclear Science Division

December 1974

NAVAL RESEARCH LABORATORY
Washington, D.C.

Approved for public release; distribution unlimited.
**REPORT DOCUMENTATION PAGE**

1. **REPORT NUMBER**
   - NRL Memorandum Report 2951
   - NRL Computer Bulletin 41

2. **GOVT ACCESSION NO.**
   - ADIA-400186

3. **RECIPIENT'S CATALOG NUMBER**

4. **TITLE (and Subtitle)**
   - A FORTRAN SUBROUTINE FOR UNPACKING AND PACKING BINARY DATA

5. **TYPE OF REPORT & PERIOD COVERED**
   - A final report on one phase of the problem.

6. **PERFORMING ORG. REPORT NUMBER**

7. **AUTHOR(S)**
   - Gary W. Phillips

8. **CONTRACT OR GRANT NUMBER(S)**

9. **PERFORMING ORGANIZATION NAME AND ADDRESS**
   - Naval Research Laboratory
   - Washington, D.C. 20375

10. **PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS**
    - NRL Problem 66H01-48

11. **CONTROLLING OFFICE NAME AND ADDRESS**
    - Department of the Navy
      - (Office of Naval Research)
    - Washington, D.C. 20360

12. **REPORT DATE**
    - December 1974

13. **NUMBER OF PAGES**
    - 14

14. **MONITORING AGENCY NAME & ADDRESS (IF different from Controlling Office)**

15. **SECURITY CLASS. (OF THIS REPORT)**
    - Unclassified

16. **DISTRIBUTION STATEMENT (OF THIS REPORT)**
    - Approved for public release; distribution unlimited.

17. **DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)**

18. **SUPPLEMENTARY NOTES**

19. **KEY WORDS (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)**
    - Computer subroutine
    - Partial word manipulation
    - Packing
    - Unpacking

20. **ABSTRACT (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)**
    - This is a general purpose routine to unpack data stored in core in packed binary format or to pack binary data stored word for word in an array. The data must be stored in bytes which are a multiple of three bits in length with a minimum length of 3 bits and a maximum of 48 bits. It is useful for unpacking data read in packed binary form and sorting it into an array so as to be convenient for further processing by a Fortran program, or for preparing data from an array for writing out in a compact form, or possibly for intermediate storage of large arrays during execution of a program in order to save core space.
## CONTENTS

1.0 IDENTIFICATION 1
2.0 PURPOSE 2
3.0 USAGE 3
4.0 METHOD OR ALGORITHM 6
5.0 SOURCE LANGUAGE LISTING 7
6.0 COMPARISON 9
7.0 TEST METHOD AND RESULTS 9
1.0 IDENTIFICATION

1.1 Title
Unpacking and Packing of Binary Data

1.2 Identification Name
M2-NRL-THREEBIT

1.3 Classification Code
M2 - Data Handling, Conversion and/or Scaling

1.4 RCC Identification Number
M2002000

1.5 Entry Points
THREEBIT, UNPACK, PACK

1.6 Programming Language
Language: 3600/3800 FORTRAN
Routine Type: Subroutine
Operating System: DRUM SCOPE 2.1

1.7 Computer and Configuration
CDC 3800

1.8 Contributor or Programmer
Gary W. Phillips, Code 6693M, Consultant Staff,
Nuclear Sciences Division

1.9 Contributing Organization
NRL - Naval Research Laboratory,
Washington, D. C. 20375

1.10 Program Availability
1.10.1 Submittal: Program write-up, Fortran
source deck, source listing

1.10.2 On File: RCC Program Library
1.11 Verification

The routine has been successfully tested in packing and unpacking binary data in bytes of length 3, 6, ..., 48 bits. In addition it has been used extensively to unpack data read in from 7-track tapes in binary format with 9, 12, 15 and 24 bit bytes.

1.12 Date

1 July 1974

2.0 PURPOSE

2.1 Description of the Routine

This is a general purpose routine to unpack data stored in core in packed binary format or to pack binary data stored word for word in an array. The data must be stored in bytes which are a multiple of three bits in length with a minimum length of 3 bits and a maximum of 48 bits.

It is useful for unpacking data read in packed binary form and sorting it into an array so as to be convenient for further processing by a Fortran program, or for preparing data from an array for writing out in a compact form, or possibly for intermediate storage of large arrays during execution of a program in order to save core space.

2.2 Problem Background

The program was originally written to unpack data read in from 7-track tapes in packed binary format of 9, 12, 15, and 24 bits in length. An existing program in the RCC Library, M2 UCSD BYTES, was found to be inapplicable as it could pack and unpack bytes in lengths only of 1, 6, 12, and 24 bits. Thus it could not be used for 9 bit and 15 bit lengths. Also, the logic was set up to handle only one byte at a time rather than handling the data as an array.
3.0 USAGE

3.1 Calling Sequence or Operational Procedure

UNPACK (NA, ARRAY, NT, TEMPLATE, NW, IWORK, NBITS, NRETRN)
PACK (NA, ARRAY, NT, TEMPLATE, NW, IWORK, NBITS, NRETRN)

3.2 Arguments, Parameters, and/or Initial Conditions

Entry UNPACK calls for unpacking data from ARRAY into TEMPLATE.

Entry PACK calls for packing data from TEMPLATE into ARRAY.

NA is the dimension of ARRAY.

ARRAY is the array used for the packed data.

NT is the dimension of TEMPLATE.

TEMPLATE is the array used for the unpacked data.

NW is the dimension of IWORK and must be at least 2*NA.

IWORK is working space used by the routine.

NBITS is the length in bits of the bytes.

NRETRN is a parameter returned by the routine to indicate the length of the unpacked or packed data returned. If negative, it indicates an error condition (see Section 3.5).

3.3 Space Required (Decimal and Octal)

3.3.1 Unique Storage:

777 octal (511 decimal) locations exclusive of computer system library functions

3.3.2 Common Blocks: None

3.3.3 Temporary Storage: IWORK(NW)
3.4 **Messages and Instructions to the Operator**

None

3.5 **Error Returns, Messages, and Codes**

Several error messages are printed on the standard output unit. A negative value for NRETRN indicates the data was not processed and the reason is printed.

a. **NBITS = (I3) IS NOT AN INTEGER MULTIPLE OF 3**
   Data not processed, NBITS must be evenly divisible by 3, NRETRN = -1.

b. **NBITS = (I3) IS GREATER THAN 48**
   Data not processed, NBITS cannot exceed 48, NRETRN = -2.

c. **NW = (I5) MUST BE AT LEAST TWICE NA = (I5)**
   Data not processed, NW is less than 2*NA, NRETRN = -3.

The calling program should take appropriate action when a negative NRETRN is received, e.g. terminate the program or go on to process the next data set.

3.6 **Informative Messages to the User**

The following messages indicate only part of the data could be processed.

a. **NUMBER OF BYTES = (I5) EXCEEDS NT = (I5)**
   NT BYTES WILL BE PROCESSED
   ARRAY could not be completely unpacked because NT was less than (NA*48)/NBITS, the total number of bytes of length NBITS contained in ARRAY.

b. **PACKED LENGTH = (I10) EXCEEDS NA = (I5)**
   THE FIRST (I5) BYTES WILL BE PROCESSED
   NT bytes of length NBITS would occupy a packed length of (NT*NBITS+47)/48 words, which exceeds NA. Only (NA*48)/NBITS bytes will be packed.

Upon return from the above two cases or upon a normal return from a call to UNPACK or PACK, NRETRN will contain the length in words of the unpacked (TEMPLATE) or packed (ARRAY) data, respectively. Partial bytes
will not be processed. If the packed data ends with a partial word it will be zero filled on the right. Elements of TEMPLATE or ARRAY with index greater than NRETRN will contain their previous values. Consequently, after return from THREEBIT the calling program should take care not to process elements of the data with index greater than NRETRN.

3.7 Input
None

3.8 Output
None other than the output described in Sections 3.5 and 3.6.

3.9 Formats
Not applicable

3.10 External Routines and Symbols
XMØDF

3.11 Timing
No timing estimates were made; the timing depends on the data length and byte length. The example in Section 7.0 took five seconds, excluding compilation time.

3.12 Accuracy
Not applicable

3.13 Cautions to Users
See Sections 3.5 and 3.6
3.14 Program Deck Structure

7 JOB card

7 FTN card

main program deck (includes call to PACK or UNPACK)

SUBROUTINE THREEBIT

SCOPE card

7 LOAD card

7 RUN card

Data (if any)

EOF

3.15 References - Literature - Appendices

None

4.0 METHOD OR ALGORITHM

The Fortran statements DECODE and ENCODE are used in UNPACK to go from a packed binary format in ARRAY to a packed BCD format in IWORK to an unpacked one byte per word format in TEMPLATE. For entry PACK the reverse of the above is done. Variable formats and variable dimensions are used to make the routine as general as possible.
5.0 SOURCE LANGUAGE LISTING

SURROUTINE THREEBIT (NA, ARRAY, NT, TEMPLATE, NW, IWORK, NRT, NRETURN)

C IDENT NUMBER - M2002000
C TITLE - UNPACKING AND PACKING OF BINARY DATA
C IDENT NAME - M2 NRL THREEBIT
C LANGUAGE - 3600/3800 FORTRAN
C COMPUTER - CDC 3800
C CONTRIBUTOR - GARY W. PHILLIPS, CODE 6603
C CONSULTANT STAFF
C ORGANIZATION - NUCLEAR SCIENCES DIVISION
C ORGANIZATION - NAVAL RESEARCH LABORATORY
C DATE - 16 JUNE 1976

GENERAL PURPOSE PROGRAM TO PACK AND UNPACK BINARY DATA IN BYTES
WHICH ARE A MULTIPLE OF THREE BITS UP TO 26 BYTES MAXIMUM.
ENTRY UNPACK OPERATES ON PACKED DATA IN ARRAY, DIMENSION NA, AND
RETURNS THE UNPACKED DATA IN TEMPLATE, DIMENSION NT.
ENTRY PACK WILL PACK DATA FROM TEMPLATE INTO ARRAY.
A WORK AREA WORK IS REQUIRED OF DIMENSION NW AT LEAST TWICE NA.

DIMENSION ARRAY (NA), TEMPLATE (NT), IWORK (NW), IFORM (2), JFORM (2)
TYPE INTEGER ARRAY, TEMPLATE

ENTRY JPACK
NPACK = 0
GO TO 10
ENTRY PACK
NPACK = 1
10 IF (MOD (NRT, 3)) 20, 30
20 PRINT 21, NRITS
30 NRITS = NRITS + 13
IF (NRITS .EQ. NT) GO TO 40
RETURN
31 IF (NRITS .LT. 49) GO TO 40
PRINT 31, NRITS
RETURN
40 IF (NA .LE. NW/2) GO TO 50
PRINT 41, NW
RETURN
41 FORMAT (*) NRITS .LE. NT .AND. NT .LE. 15
50 IF (NPACK) GO TO 60
RETURN
50 IF (NW .LE. (4A4+8)/NRT) GO TO 100
RETURN
51 IF (NRITS .LT. 15)
RETURN
60 NRITS = NRITS + 1
GO TO 10
RETURN
C
60 NB=NT
NC=(NH*NBITS+47)/8
IF (NC.LE.NA) GO TO 100
NH=NA*4/NC
PRINT 51,NCG,NA,NB
61 FORMAT(*OPACKED LENGTH =#110# EXCEEDS NA =#15/
* THE FIRST#15# BYTES WILL BE PROCESSED*)
NC=(NH*NBITS+47)/8
C
100 IF (NA=256) 110,110,120
110 I=IA*NA
ENCOD(16,101,JFORM) NA
101 FORMAT(*#14014#*)
GO TO 130
120 I=IA/256
IF (MOD(NA*256)) IA=IA+1
I=IA*6096
ENCOD(16,121,JFORM) IA
121 FORMAT(*#120(25601))*)
130 IF (NA=256) 140,140,120
140 I=IWH#K
ENCOD(16,141,JFORM) NA#K
141 FORMAT(*#14014#*)
GO TO 200
150 I=IA/256
IF (MOD(IA*256)) IA=IA+1
I=IA*7656#K
ENCOD(16,151,JFORM) IA#K
151 FORMAT(*#120(256012#))*)
C
200 IF (NPACK) 240,210
210 ENCOD(I=JFORM,1#0K.(ARRAY(I),I=1,NA)
220 DECOD(I=JFORM,1#0K.(TEMPLATE(I),I=1,N#)
NHT=NA
RETURN
C
240 NC=2#NC
INC(NJ)=0
245 ENCOD(I=JFORM,1#0K.(TEMPLATE(I),I=1,N#)
250 DECOD(I=JFORM,1#0K.(ARRAY(I),I=1,NC)
NHT=NC
C
END
6.0 COMPARISON

The present routine will handle packed arrays with byte lengths of any multiple of three bits up to 48 bits. The routine M2 UCSD BYTES in the RCC Program Library will handle byte lengths of 1, 6, 12, and 24 bits, but only one byte at a time.

Except for the one bit case, it is usually preferable to unpack the data before further processing rather than handling it byte by byte.

7.0 TEST METHOD AND RESULTS

A test program, PCKUNPCK, was written to unpack from ARRAY to TEMPLATE and then pack from TEMPLATE into ARRAY, for NBITS equal to 6, 18, and 30. The listing of PCKUNPCK and the results follow.
PROGRAM PCKUNPCK
DIMENSION ARRAY(4), BARRAY(4), TEMPLATE(20), IWORK(8)
DATA (ARRAY(4), (12345678)12345678) *
   (NA=4), (NT=20), (NW=8), (NB=4)
TYPE INTEGER ARRAY, ARRAY, TEMPLATE
DO 20 I=1,13,12
   NBITS=1
   NCHAR=NBITS/3
   PRINT 101, NBITS, NCHAR
101 FORMAT ($('#', #', #', #'), #', #', #', #', #', #')
   PRINT 10, ARRAY
10 FORMAT ('/###', ARRAY(4)/1X4/1X8)
   DO 11 J=1, NT
      NBITS=0
      CALL UNPCK(NA, ARRAY(NT), TEMPLATE(3), NW, IWORK, NBITS, NL)
      PRINT 11, NL, (TEMPLATE(J), J=1, NL)
11 FORMAT ('/###', TEMPLATE(3)/1X4/1X8)
   DO 12 J=1, NB
      NCHAR=0
      CALL PACK(NA, ARRAY(NT), TEMPLATE(3), NW, IWORK, NCHAR, NL)
20 PRINT 12, NL, (ARRAY(J), J=1, NL)
12 FORMAT ('/###', ARRAY(NT)/1X4/1X8)
   END
- - - NBITS = 6 NCHAR = 2 - - -

ARRAY
1234567012345670 1234567012345670 1234567012345670 1234567012345670

NUMBER OF BYTES = 32 EXCEEDS NT = 20
NT BYTES WILL BE PROCESSED

TEMPLATE LENGTH = 20
0000000000000012 0000000000000034 0000000000000056 0000000000000070
0000000000000012 0000000000000034 0000000000000056 0000000000000070
0000000000000012 0000000000000034 0000000000000056 0000000000000070
0000000000000012 0000000000000034 0000000000000056 0000000000000070

ARRAY LENGTH = 3
1234567012345670 1234567012345670 1234567000000000

- - - NBITS = 18 NCHAR = 6 - - -

ARRAY
1234567012345670 1234567012345670 1234567012345670 1234567012345670

TEMPLATE LENGTH = 10
000000000000123456 00000000000701234 000000000000567012 0000000000345670
0000000000123456 00000000000701234 000000000000567012 0000000000345670
0000000000123456 0000000000701234

PACKED LENGTH = 8 EXCEEDS NA = 4
THE FIRST 10 BYTES WILL BE PROCESSED

ARRAY LENGTH = 4
1234567012345670 1234567012345670 1234567012345670 123456701234000

- - - NBITS = 30 NCHAR = 10 - - -

ARRAY
1234567012345670 1234567012345670 1234567012345670 1234567012345670

TEMPLATE LENGTH = 6
0000001234567012 0000003456701234 0000005670123456 0000007012345670
0000001234567012 0000003456701234

PACKED LENGTH = 13 EXCEEDS NA = 4
THE FIRST 6 BYTES WILL BE PROCESSED

ARRAY LENGTH = 4
1234567012345670 1234567012345670 1234567012345670 123456701234000