Subject: Re: Convert ascii data to binary
Posted by rigby on Fri, 28 Jan 2005 16:29:00 GMT
View Forum Message <> Reply to Message

This routine might be useful. The heart of the code is from C. D. Pike
in 1993.

```
help, int2bin(byte('8f'x))
<Expression>    STRING    = '10001111'

--Wayne


;*******************************
FUNCTION Defined, var
;*******************************
;~ Returns 1 if VAR is defined, and 0 otherwise. [Programming]

On_Error, 2

if (n_params() NE 1) then $
message, /noname, "Usage: Defined(var)"

s = size(var)
return, (s[s[0]+1] NE 0)

end  ;; defined


;+
;************************************************
FUNCTION Int2Bin, data, $
nibble=nibble, $
nb = nb, $
bitarray=bitarray, $
formatStr=formatStr
;************************************************
;~ Converts integer-type DATA into its binary representation.
[Conversion, String]
;
;  Data must be an integer-type scalar or 1D array.
;
;  By default, a string is returned consisting of the binary
;  representation of DATA, in groups of eight bits, most significant
;  bit leftmost.  If NIBBLE is set, the bits are instead displayed in
;  groups of 4.
;
```

```
;  You can restrict the result to NB bytes (or, if NIBBLE is set, NB
nibbles) by
;  setting NB to a positive integer value. (This option is not
available currently
;  if BITARRAY is set.)
;
;  If BITARRAY is set, a byte array is returned, in which each
;  element is the corresponding binary bit of DATA, BUT IN REVERSE
ORDER.
;  The array has 8, 16, 32 or 64 elements, when DATA is a byte,
integer,
;  integer long or long-long, respectively.
;
;  You can create a string representation of fixed-point integers,
e.g., "1100.01"
;  by supplying FORMATSTR in the form 'bN.M', where N is the total
number of binary
;  bits returned and M is the number of bits displayed to the right of
the binary point.
;  The result is padded with leading 0's to exactly N binary digits, if
necessary.
;
;  Examples:
;
;  help, Int2Bin('8f'x)
;   <Expression>    STRING    = '00000000 10001111'
;
;  print, int2bin('8f'x, nb=1)
;   10001111
;
;  help, Int2Bin(fix('8f'x), /bitarray)
;    <Expression>    BYTE      = Array[16]
;  print, Int2Bin(fix('8f'x), /bitarray)
;    0   0   0   0   0   0   0   0   1   0   0   0   1   1   1   1
;
;  print, int2bin('8f'x, format='b5.1')
;   0111.1
;-

;  Modification history:
;  Dec 30 1993  W. Rigby.  Original by C. D. Pike, 7-Oct-93, with my
minor mods.
;  ...
;  Jan 14 2005  stripped down before posting.

On_Error, 2

MsgPrefix = "[Int2Bin] "
```

```
maxBits = 64

if (n_params() NE 1) then $
Message, /noname, "Usage: Int2Bin, data {[/nibble || /bitarray] [nb]}
|| {formatStr=}"

doBitArray = Keyword_Set(BITARRAY)
doNibble = Keyword_Set(NIBBLE)
doFormatStr= Defined(formatStr)
doNB = Defined(nb)

;;if (CheckArgument(data, "integerType", errMsg)) then $
;;  Message, /noname, MsgPrefix + "DATA" + errMsg
;;
;;if (CheckArgument(n, /optional, "integerType", /positive, errMsg))
then $
;;  Message, /noname, MsgPrefix + "N" + errMsg
;;
;;if (CheckArgument(formatStr, /optional, "string", /scalar, errMsg))
then $
;;  Message, /noname, MsgPrefix + "FORMATSTR" + errMsg
;;
;;if (CheckArgument(nb, /optional, "integertype", /scalar, minvalue=1,
maxValue=8, errMsg)) then $
;;  Message, /noname, MsgPrefix + "NB" + errMsg

if (doNibble AND doBitArray) then $
Message, /noname, MsgPrefix + "BITARRAY can't be set when NIBBLE is."

if (doNB and doBitArray) then $
Message, /noname, MsgPrefix + "The keyword NB not supported when
BITARRAY is set"

if (doFormatStr) then begin
if (doBitArray) then $
Message, /noname, MsgPrefix + "BITARRAY can't be set when
FORMATSTR is supplied."
if (doNibble) then $
Message, /noname, MsgPrefix + "NIBBLE can't be set when FORMATSTR
is supplied."
if (Defined(nB)) then $
Message, /noname, MsgPrefix + "NB can't be set when FORMATSTR is
supplied."

msg = "FORMATSTR must be in the form 'bn.m' with (n GT 0) and (0 LE
m LE n)"
```

```
if (StrLowCase(StrMid(formatStr, 0, 1)) NE 'b') then $
Message, /noname, MsgPrefix + msg

p = StrPos(formatStr, ".")  ;;
if (p LT 2) then $
Message, /noname, MsgPrefix + msg
n = StrLen(formatStr)
if (p EQ n-1) then $
Message, /noname, MsgPrefix + msg

width = StrMid(formatStr, 1, p-1)
;;   if (MyNot(StrIsNumber(width, /int))) then $
;;     Message, /noname, MsgPrefix + msg

nBinary = StrMid(formatStr, p+1, n-1-(p+1)+1)
;;   if (MyNot(StrIsNumber(nBinary, /int))) then $
;;     Message, /noname, MsgPrefix + msg

width = fix(width)
nBinary = fix(nBinary)

if ((width LE 0) OR (nbinary LT 0)) then $
Message, /noname, MsgPrefix + msg

if (width LT nBinary) then $
Message, /noname, MsgPrefix + msg

doBitArray = 0   ;; force a string result
endif

;;nd = ndim(data)
nd = (size(data))[0]
case (nd) of
0: ;; scalars OK
1: ;; 1D arrays OK
else: $
Message, /noname, MsgPrefix + "DATA must be a scalar or a
one-dimensional array."
endcase

; make a mask for each of the 64 possible bit positions
; mask[0] has the MSB set and mask[maxBits-1] has bit 0 set.
mask = Rotate(2ULL^indgen(maxBits), 2)

case (nd) of
0: out = ((ulong64(data) AND mask) NE 0)
1: begin
ndata = n_elements(data)
```

```
nmask = n_elements(mask)

result = strarr(ndata)
out = ulon64arr(ndata, nmask)       ;; out[i] is 1 if bit MSB-i
is set
for i = 0, nmask-1 do $             ;; thus out[0] <--> MSB, out[
out[*,i] = ((ulong64(data) AND mask[i]) NE 0)
end
endcase

;  trim output depending on input type
switch size(data, /tname) of
'UNDEFINED' : $
Message, /noname, MsgPrefix + "DATA is undefined!"

'BYTE': begin
ii1 = maxbits-1    ;; last element in OUT --> bit 0 in DATA, so
we want
ii0 = ii1 - 8 + 1  ;; the last 8 bits of OUT
if (doBitArray) then begin
case (nd) of
0: result = byte(out[ii0:ii1])
1: result = byte(out[*,ii0:ii1])
endcase
endif else begin
if (doNibble) then $
fmtstr = '(2(4I1, 1x))' $
else $
fmtstr = '(8I1)'

case (nd) of
0: result = String(format=fmtstr, out[ii0:ii1])
1: begin
;; don't know how to do this without a loop
for i = 0, ndata-1 do $
result[i] = String(format=fmtstr, out[i, ii0:ii1])
end
endcase
endelse
break
end

'UINT' :
'INT': begin
ii1 = maxbits-1
ii0 = ii1 - 16 + 1   ;; we want the last 16 bits of OUT
if (doBitArray) then begin
case (nd) of
```

```
0:  result = byte(out[ii0:ii1])
1:  result = byte(out[*, ii0:ii1])
endcase
endif else begin
if (doNibble) then $
fmtstr = '(4(4I1, 1x))' $
else $
fmtstr = '(2(8I1, 1x))'

case (nd) of
0: result = String( format=fmtstr, out[ii0:ii1])
1: for i = 0, ndata-1 do $
result[i] = String( format=fmtstr, out[i,ii0:ii1] )
endcase
endelse
break
end

'ULONG' :
'LONG': begin
ii1 = maxbits-1
ii0 = ii1 - 32 + 1   ;; we want the last 32 bits of OUT

if (doBitArray) then begin
case (nd) of
0: result = byte(out[ii0:ii1])
1: result = byte(out[*,ii0:ii1]) ; just to be explicit
about what's going on
endcase
endif else begin
if (doNibble) then $
fmtstr = '(8(4I1, 1x))' $
else $
fmtstr = '(4(8I1, 1x))'

case (nd) of
0: result = String(format=fmtstr, out[ii0:ii1] )
1: for i = 0, ndata-1 do $
result[i] = String(format=fmtstr, out[i,ii0:ii1])
endcase
endelse
break
end

'LONG64' :
'ULONG64' : begin
ii1 = maxbits-1
ii0 = ii1 - 64 + 1 ;; i.e., 0
```

```
if (doBitArray) then begin
case (nd) of
0: result = byte(out[ii0:ii1])
1: result = byte(out[*,ii0:ii1]) ; just to be explicit
about what's going on
endcase
endif else begin
if (doNibble) then $
fmtstr = '(16(4I1, 1x))' $
else $
fmtstr = '(8(8I1, 1x))'

case (nd) of
0: result = String( format=fmtstr, out)
1: for i = 0, ndata-1 do $
result[i] = String( format=fmtstr, out)
endcase
endelse
break
end

else: $
Message, /noname, MsgPrefix + "This can't happen."  ;; new integer
data type?
endswitch

if (doNB) then begin
if (doBitArray) then begin
;; easy to add, but no need currently
Message, /noname, MsgPrefix + "This can't happen."
endif else begin
;; result is a string, with bytes (or nibbles) separated by
spaces
nr = n_elements(result)        ;; = n_elements(data)

for j = 0, nr-1 do begin
;; strArr = SplitString(result[j])   ;; split at whitespace
strArr = StrSplit(result[j], " ", /extract)
n = n_elements(strArr)            ;; number of bytes or
nibbles in the string array

if (nb LT n)  then begin
r = strArr[n-1]         ;; least significant byte or
nibbles
for i = 1, nb-1 do begin
r = strArr[n-1-i] + " " + r  ;; add bytes or nibbles
from right to left
endfor
```

```
result[j] = r   ;; OK to index scalars (a=4, a[0] = 3) --
since when?
endif ;; otherwise just return the whole string
endfor
endelse
endif

if (doFormatStr) then begin
;; put the bitarray into n.m format.
;; remember, lsb is in the last element of RESULT

;; remove all the (byte or nibble) spaces first -- too messy to try
to keep them.
result = StrCompress(result, /remove_all)

;; trim or pad to WIDTH bits
n = StrLen(result)
if (width LE n) then $
temp = StrMid(result, n-width, width) $
else $
;;     temp = MakeString(width-n, char='0') + result
temp = string(make_array(n, value=(byte('0'))[0]))
n = width

result = StrMid(temp, 0, width-nbinary) + "." + StrMid(temp,
n-nbinary, nbinary)
endif

return, result

end   ;  Int2Bin
```