Mosaic Industries

8364 Visibility Sensor

APPLICATION NOTE
MI-AN-017

Summary

The following codes shows how the QED Board receives 8364 Visibility Sensor messages, assists access to constitutent parts of the message, and coverts the message to another representation better suited for additional manipulation.

```
ANEW 8364.COMM
\ These words demonstrate how the QED Board can receive 8364 Visibility
\ Sensor messages, facilitate easy access to constituent parts of the
\ messages, convert these message components to alternative representation
\ suitable for additional manipulation (ie text to floating point format).
\
\ The code defines a record data structure with fields corresponding to
\ the constituant parts of the 8364 message. Memory is then allocated for
\ one of these record structures. Words are used to monitor the serial port
\ and store incoming message to the buffer. Once a complete message has
\ been received, utility words may be executed to access and convert the
\ received information. Not all possible conversions have been supplied
\ and have been left as an exercise for the user.   : )
\
\ This code is NOT a complete implementation of the 8364 protocol, and is
\ provided without any expressed or implied warrenty. The user of this code
\ assumes all risk in its use and implementation.
\
\ Questions regarding this code should be directed to:
\   Melody Liu            Mosaic Industries, Inc.          510/790-8222
\

HEX               \ Express all numbers as hexadecimal values in this code file.
9600 0 DP X!      \ Move dictionary up to allow room for many names.

\ The following constants define field lengths for each component of the
\ incoming 8364 message:
3     CONSTANT    DAY.CNT
8 1+  CONSTANT    TIME.CNT           \ The 1+ explicitly accounts for the space
6 1+  CONSTANT    OUTPUT.CNT         \ delimiter.
4 1+  CONSTANT    1.STATUS.CNT
4 1+  CONSTANT    2.STATUS.CNT
D 1+  CONSTANT    1.RESERVED.CNT
6 1+  CONSTANT    00.AVERAGE.CNT
6 1+  CONSTANT    01.AVERAGE.CNT
6 1+  CONSTANT    10.AVERAGE.CNT
6 1+  CONSTANT    11.AVERAGE.CNT
4 1+  CONSTANT    2.RESERVED.CNT
```

```
2      CONSTANT    PACKET.LENGTH.CNT
4      CONSTANT    SOURCE.ADDR.CNT
4      CONSTANT    DEST.ADDR.CNT
2      CONSTANT    PACKET.NUM.CNT
2      CONSTANT    INSTRUCTION.CNT
4      CONSTANT    RESERVED.CNT
2      CONSTANT    MSB.CRC.CNT
2      CONSTANT    LSB.CRC.CNT


\ 8364.DATA.FIELD is a sub-structure field of the more complete 8364.PACKET
\ structure defined below. Information in 8364.DATA.FIELD is the essential
\ sensor information.
STRUCTURE.BEGIN:  8364.DATA.FIELD
      DAY.CNT                 BYTES->     +DAY
      TIME.CNT          BYTES->     +TIME
      OUTPUT.CNT        BYTES->     +OUTPUT
      1.STATUS.CNT            BYTES->     +1.STATUS
      2.STATUS.CNT            BYTES->     +2.STATUS
      1.RESERVED.CNT    BYTES->     +1.RESERVED
      00.AVERAGE.CNT    BYTES->     +00.AVERAGE
      01.AVERAGE.CNT    BYTES->     +01.AVERAGE
      10.AVERAGE.CNT    BYTES->     +10.AVERAGE
      11.AVERAGE.CNT    BYTES->     +11.AVERAGE
      2.RESERVED.CNT    BYTES->     +2.RESERVED
STRUCTURE.END

\ 8364.PACKET defines the structure of 8364 message packets and include an
\ 8364.DATA.FIELD sub-structure field.
STRUCTURE.BEGIN: 8364.PACKET
       PACKET.LENGTH.CNT      BYTES->     +PACKET.LENGTH
      SOURCE.ADDR.CNT         BYTES->     +SOURCE.ADDR
      DEST.ADDR.CNT           BYTES->     +DEST.ADDR
      PACKET.NUM.CNT          BYTES->     +PACKET.NUM
      INSTRUCTION.CNT         BYTES->     +INSTRUCTION
      RESERVED.CNT            BYTES->     +RESERVED
      8364.DATA.FIELD         STRUCT->    +DATA
      MSB.CRC.CNT             BYTES->     +MSB.CRC
      LSB.CRC.CNT                   BYTES->     +LSB.CRC
STRUCTURE.END

\ Create and allocate an instance of 8364.PACKET. This data structure is
\ used as the message receiving buffer in addition to providing access to
\ its data fields.

8364.PACKET V.INSTANCE: MESSAGE
\ Define words to manage the message buffer.

: CLEAR.MESSAGE.BUFFER ( -- | Fill all MESSAGE mem locations with 0 )
MESSAGE SIZE.OF MESSAGE ERASE
;

0D  CONSTANT RET  \ ASCII carriage return character
0A  CONSTANT LF   \ ASCII line feed character
```

```
\ RECEIVE.MESSAGE stores 98 characters in memory beginning at MESSAGE.
\ Msg.rcvd.tflag is TRUE if 98 chars are received followed by a CR,
\ otherwise FALSE is returned. The tflag does not refer to the validity
\ of the data received.
: RECEIVE.MESSAGE ( -- msg.rcvd.tflag )
  0 SIZE.OF MESSAGE LOCALS{ &buffer.size &cnt }
 BEGIN
        KEY                    \ Get the next character on the comm port
\    KEY2                      \ Substitute this line for the prev to use comm2
      DUP RET <>               \ Is the char a carriage return?
       &cnt &buffer.size < AND\ Are we at the end of the buffer?
      WHILE                    \ If no to each of these...
        MESSAGE &cnt XN+ C!    \ Store the current char in MESSAGE
        &cnt 1+ TO &cnt        \ Increment count
 REPEAT
RET =                          \ Did we end on a carriage return?
&cnt &buffer.size =            \ Did we fill the buffer?
AND                            \ Both are required to return a TRUE
;

\                 General data type conversion words:
\ TEXT>PAD$ moves the given text string and count to the temporary memory
\ buffer named PAD (system defined) in the standard counted string format.
: TEXT>PAD$ ( text.xaddr\cnt -- PAD )
      LOCALS{ &cnt x&text }
      PAD &cnt 2+ 20 FILL               \ Fill PAD with cnt+2 spaces
      x&text PAD 1XN+ &cnt CMOVE &cnt PAD C!    \ Move text to PAD and store cnt
      PAD                               \ Return the PAD address
;
\ CONTENT>NUMBER converts a number text string given a character count to
\ an integer, double int, or floating point number. The converted value is pushe
\ onto the data stack accompanied by a type indicator value: 1 = int, 2 = double
\ 3 = floating point, 0 = text could not be converted.
: CONTENT>NUMBER ( text.xaddr\cnt -- [0] or [n\1] or [d\2] or [f\3] )
      BL SKIP LOCALS{ &cnt x&text.addr }
      x&text.addr &cnt TEXT>PAD$   NUMBER   ?DUP   0=
      IF    x&text.addr &cnt TEXT>PAD$   FNUMBER  -3 *
      ENDIF
;

\ Specific information conversion words:
: PACKET.LENGTH ( -- [0] or [n\1] or [d\2] or [f\-1] )
      MESSAGE +PACKET.LENGTH PACKET.LENGTH.CNT CONTENT>NUMBER
;

: SOURCE.ADDRESS ( -- [0] or [n\1] or [d\2] or [f\-1] )
      MESSAGE +SOURCE.ADDR SOURCE.ADDR.CNT CONTENT>NUMBER
;

: CRC ( -- [0] or [n\1] or [d\2] or [f\-1] )
      MESSAGE +MSB.CRC MSB.CRC.CNT CONTENT>NUMBER DROP 8 SCALE
      MESSAGE +LSB.CRC LSB.CRC.CNT CONTENT>NUMBER DROP OR 1
;
```

```
: DAY ( -- [0] or [n\1] or [d\2] or [f\-1] )
     MESSAGE +DATA +DAY DAY.CNT 1+ CONTENT>NUMBER
;

: TIME ( -- | prints time, no conversion )
     MESSAGE +DATA +TIME TIME.CNT 1+ TYPE
;

: OUTPUT ( -- [0] or [n\1] or [d\2] or [f\-1] )
     MESSAGE +DATA +OUTPUT OUTPUT.CNT 1+ CONTENT>NUMBER
;

\ To test these words execute:
\   CLEAR.MESSAGE.BUFFER      \ Zero the message buffer.
\   RECEIVE.MESSAGE           \ Receive message and store.
\   Execute any of the information conversion words:
\   OUTPUT
\   PACKET.LENGTH
\   SOURCE.ADDRESS
```