

```

/*****
/*
/*----- D U M P C -----*/
/*
/* Task : A filter, which reads in characters from the
/* standard input device and outputs them as a
/* hex and ASCII dump on the standard output device */
/*-----*/
/* Author : MICHAEL TISCHER */
/* Developed on : 08/14/87 */
/* Last update : 04/07/95 */
/*****
#include <stdio.h> /* Include header files */
#include <dos.h>

/*== Type definitions =====*/
typedef unsigned char BYTE; /* Handle as a byte */

/*== Constants =====*/
#define NUL 0 /* ASCII code for NULL character */
#define BEL 7 /* ASCII code for Bell */
#define BS 8 /* ASCII code for Backspace */
#define TAB 9 /* ASCII code for Tab */
#define LF 10 /* ASCII code for Linefeed */
#define CR 13 /* ASCII code for Carriage Return */
#define ESC 27 /* ASCII code for Escape */

/*== Macros =====*/
#define tohex(c) ( ((c)<10) ? ((c) | 48) : ((c) + 'A' - 10) )

/*****
/* GETSTDIN: Reads a certain number of characters from the standard
/* input device and places them in a buffer
/* Input : BUFFER = Pointer to buffer receiving characters
/* MAXCHAR = Maximum number of characters read at a time
/* Output : Number of characters read
/*****
int GetStdIn(char *Buffer, int MaxChar)
{
    union REGS Register; /* Register variable for interrupt call */
    struct SREGS Segment; /* Accepts the segment register */

    segread(&Segment); /* Read contents of segment register */
    Register.h.ah = 0x3F; /* Function number */
    Register.x.bx = 0; /* Standard input device is handle 0 */
    Register.x.cx = MaxChar; /* Number of bytes to be read */
    Register.x.dx = (unsigned int) Buffer; /* Offset address of buffer */
    intdosx(&Register, &Register, &Segment); /* Call Interrupt 21H */
    return(Register.x.ax); /* Number of bytes read to caller */
}

/*****
/* STRAP : Attach character to string
/* Input : STRING = Pointer to string to be appended
/* TEXTPOINTER = Pointer to string with additional text
/* Output : Pointer behind the last added character
/*****
char *Strap(char *String, char *Textpointer)
{
    while (*Textpointer) /* Repeat until '\0' detected */
        *String++ = *Textpointer++; /* Transmit character */
    return(String); /* Pass pointer to calling function */
}

/*****
/* DODUMP : Reads the characters in and outputs them as dump
/* Input : None
/* Output : None
/*****

void DoDump( void )
{
    char NineBytes[9], /* Accepts the characters read */
        DumpBuf[80], /* Accepts a line of DUMP */
        *NextAscii; /* Points to next ASCII character in the buffer */
    BYTE i, /* Loop counter */

```

