ECE3120: Computer Systems
Chapter 4: D-Bug12 Functions

Manjeera Jeedigunta
http://blogs.cae.tntech.edu/msjeedigun21
Email: msjeedigun21@tntech.edu
Tel: 931-372-6181, Prescott Hall 120

- Prev
  - Subroutines
- Today
  - Using the D-Bug 12 Functions to perform I/O Operations

Using the D-Bug12 Functions to Perform I/O Operations

Function	Description	Pointer Address
`far main()`	Start of D-Bug12	$EE80
`getchar()`	Get a character from SCI0 or SCI1	$EE84
`putchar()`	Send a character out SCI0 or SCI1	$EE86
`printf()`	Formatted Output - Translates binary values to characters	$EE88
`far GetCmdLine()`	Obtain a line of input from the user	$EE8A
`far sscanhex()`	Convert an ASCII hexadecimal string to a binary integer	$EE8E
`isxdigit()`	Checks for membership in the set [0..9, a..f, A..F]	$EE92
`toupper()`	Converts lower case characters to upper case	$EE94
`isalpha()`	Checks for membership in the set [a..z, A..Z]	$EE96
`strlen()`	Returns the length of a null terminated string	$EE98
`strcpy()`	Copies a null terminated string	$EE9A
`far out2hex()`	Displays 8-bit number as 2 ASCII hex characters	$EE9C
`far out4hex()`	Displays 16-bit number as 4 ASCII hex characters	$EEA0
`SetUserVector()`	Setup user interrupt service routine	$EEA4
`far WriteEEByte()`	Write a data byte to on-chip EEPROM	$EEA6
`far EraseEE()`	Bulk erase on-chip EEPROM	$EEAA
`far ReadMem()`	Read data from the M68HC12 memory map	$EEAE
`far WriteMem()`	Write data to the M68HC12 memory map	$EEB2

- All functions listed in Table 4.2 are written in C language.

- The first parameter to the function must be passed in accumulator D. The remaining parameters must be pushed onto the stack in the reverse order they are listed in the function declaration.
- Parameters of type **char** will occupy the lower order byte of a word pushed onto the stack.
- Parameters pushed onto the stack before the function is called remain on the stack when the function returns. It is the responsibility of the caller to remove passed parameters from the stack.
- All 8- and 16-bit values are returned in accumulator D. A returned value of type **char** is returned in accumulator B.

int getchar (void)

Pointer address:	$EE84
Function:	retrieve a single character from the control terminal.
Incoming parameter:	none
Returned value:	8-bit character in accumulator B

Adding the following instruction sequence will read a character from SCI0 port:

```
getchar   equ    $EE84
          …
          jsr    [getchar,PCR]
```

int putchar(int)

Pointer address:	$EE86
Function:	outputs a single character to the control terminal
Incoming parameter:	character to be output in accumulator B
Returned value:	the character that was sent (in B)

- This function outputs a character to serial communication port SCI0.
- The character to be output should be placed in accumulator B.
- The following instruction sequence will output the character **A** to serial port SCI0

```
putchar   equ    $EE86
          …
          ldab   #$41; place the ASCII code of 'A' in accumulator B
          jsr    [putchar,PCR]
          …
```

int printf(char *format,…)

Pointer address:	$EE88
Function:	convert, format, and print its arguments on the standard output. Incoming parameters: zero or more integer data to be output on the stack, D contains the address of the format string. The format string must be terminated with a zero.
Returned value:	number of characters printed in D.

- This function is used to convert, format, and print its argument as standard output under the control of the format string pointed to by *format*.
- All except floating-point data types are supported.
- The format string contains two basic types of objects:

1. ASCII characters which will be copied directly to the display device.
2. Conversion specifications. Each conversion specification begins with a percent sign (%).
3. Optional formatting characters may appear between the percent sign and ends with a single conversion character in the following order:

[-][<FieldWidth>][.][<Precision>][h | l]

The Meaning of Optional Characters

Table 4.3 Optional formatting characters

Character	Description
-(minus sign)	Left justifies the converted argument.
FieldWidth	Integer number that specifies the minimum field width for the converted arguemnt. The argument will be displayed in a field at least this wide. The displayed argument will be padded on the left or right if necessary.
. (period)	Separates the field width from the precision.
Precision	Integer number that specifies the maximum number of characters to display from a string or the minimum number of digits for an intger.
h	To have an integer displayed as a short.
l(letter ell)	To have an integer displayed as a long.

Formatting Characters Supported by the *printf() function:*

Table 4.4 Printf() conversion characters

character	Argument type; displayed as
d, i	int; signed decimal number
o	int; unsigned octal number (without a leading zero)
x	int; unsigned hex number using abcdef for 10...15
X	int; unsigned hex number using ABCDEF for 10...15
u	int; unsigned decimal number
c	int; single character
s	char *; display from the string until a '\0' (NULL)
p	void *; pointer (implementation-dependent representation)
%	no argument is converted; print a %

Example for outputting a message (**Flight simulation**):

```
CR      equ     $0D
LF      equ     $0A
printf  equ     $EE88
        …
        ldd     #prompt
        jsr     [printf,PCR]
        …
prompt  db      “Flight simulation”,CR,LF,0
```

Example for outputting three numbers **m, n,** and **gcd** along with a message:

```
CR      equ     $0D
LF      equ     $0A
printf  equ     $EE88
        …
        ldd     gcd
        pshd
        ldd     n
        pshd
        ldd     m
        pshd
        ldd     #prompt
        jsr     [printf,PCR]
        leas    6,sp
        …
prompt  db      “The greatest common divisor of %d and %d is %d”,CR,LF,0
```

int far GetCmdLine(char *CmdLineStr, int CmdLineLen)

Pointer address:	$EE8A
Incoming parameters:	a pointer to the buffer where the input string is to be stored and the maximum number of characters that will be accepted by this function.
Returned value:	a string from the user (usually from the keyboard)

- This function is used to obtain a line of input from the user.
- The reception of an ASCII carriage return ($0D) terminates the reception of characters from the user.
- The caller of this function normally would output a message so that the user knows to enter a message. The example in the next page illustrates this interaction.

Example: GetCmdLine

```
printf      equ   $EE88
GetCmdLine  equ   $EE8A
cmdlinelen  equ   100          ;no.of characters to accept
CR          equ   $0D
LF          equ   $0A
            …
prompt      db    “Please enter a string: “,CR,LF,0
            …
inbuf       rmb   100
            …
            ldd   #prompt      ; output a prompt to remind the user to
            jsr   [printf,PCR] ; enter a string
            ldd   #cmdlinelen  ; push the CmdLineLen
            pshd               ;          “
            ldd   #inbuf       ;address to store the string
            call  [GetCmdLine,PCR]   ; read a string from the keyboard
            puld               ; clean up the stack
```