Technical Bulletin

Using the HIF ioctl Service for Nonblocking Reads

EPD Systems Engineering March 20, 1995

Purpose

The Host Interface (HIF) kernel of Release 3.0 or later of the MiniMON29K™ product adds support for nonblocking read operations. This new feature allows application programs to continue processing while waiting for input data to be transferred. (To use this feature, MiniMON29K Release 3.0 or later must be loaded on both the target and the host.)

This bulletin shows example code that demonstrates how a nonblocking read can be used to create an interactive menu that allows subsequent processing to continue while waiting for user input.

Affected Product

The information in this bulletin affects the following product:

Product	Release
MiniMON29K	3.0 or later

The Problem

The default input mode used by the HIF kernel of the MiniMON29K product is COOKED (0x0000), which blocks (suspends) the application program issuing a **read** operation from the standard input device (terminal) until the input data has been transferred.

While blocked mode may be effective for some applications, it makes it difficult to implement and debug interactive menu-driven applications that require processing to continue while waiting for user input.

The Solution

The HIF kernel of the MiniMON29K Release 3.0 (or later) product implements nonblocking read support for standard input devices (terminals). Using the HIF **ioctl** service, the input mode of the standard input device can be changed from COOKED to NBLOCK mode, which allows the application program to continue processing while waiting for input data to be transferred.

The HIF ioctl Service

The HIF **ioctl** service establishes the operation mode of a specified file or device. It is intended to be applied primarily to terminal-like devices; however, certain modes apply to mass-storage files or to other related input/output devices.

In COOKED (0x0000) mode (the default input mode), when a **read** operation for a terminal-like device is issued by the application program, the kernel blocks (suspends) any further execution of the application program until the data has been transferred. Using the HIF **ioctl** service, the input mode of the standard input device can be set to NBLOCK (0x0010) mode, which specifies that subsequent read operations be executed without suspending (blocking) the application program issuing the **read** request.

The HIF read Service

After setting standard input to NBLOCK mode, a HIF **read** operation on the standard input device returns immediately to the application program. The return value of the **read** service contains the number of characters currently available, or –1 if none are available. The application program examines the return value from the **read** service to determine if any input data is available. If the return value is –1, the application program continues other processing while waiting for the input data.

Example Code

Following is a short code example showing how the technique explained above can be used for the creation of an interactive menu. The code in boldface highlights the use of the HIF **ioctl** service.

```
#include <stdio.h>
#include <hif.h>
#include <stdlib.h>

showmen() {
/*******************************************************************************/
/* The following subroutine will display a menu to standard output. The _write()  */
/* HIF service was used rather than printf for the following reasons:              */
/*   - To show the usage of the _write() HIF service                               */
/*   - When characters are sent to standard output using printf, the output is    */
/*     buffered and will not be displayed until a '\n' is used. Therefore, the     */
/*     "Enter your selection ->" line would not appear on standard output if       */
/*     printf were used because there is no '\n' character at the end of the      */
/*     string.                                                                     */
/*******************************************************************************/
  _write(1,"\nMenu:\n",7);
  _write(1,"\t1) Selection #1\n",17);
  _write(1,"\t2) Selection #2\n",17);
  _write(1,"\t3) Exit\n",9);
  _write(1,"\nEnter your selection -> ",25);
}

void main() {

  /* Declare necessary variables */
  int terminate=0, proc=0;

  /* Create a 1-character buffer to hold input */
  char *input=(char *) malloc(sizeof(char));

  /* Set standard input to NBLOCK mode using the _ioctl() HIF service */
  _ioctl(0,0x0010);

  showmen();

  /*****************************************************************************/
  /* Following is the main loop where processing occurs. The exit condition is    */
  /* set by selecting the appropriate menu item.                                  */
  /*****************************************************************************/
  while(!terminate) {

    /***************************************************************************/
    /* Because standard input has been set to NBLOCK mode using the _ioctl() HIF  */
    /* service, the _read() HIF service will return a negative number if there is */
    /* no input available. If there is input available, _read() will return a non-*/
    /* negative number and the character read will be contained in the input      */
    /* buffer.                                                                    */
    /***************************************************************************/
    if(_read(0,input,1)>=0) {

      /* React to the input received */
      switch(input[0]) {
        case '1':
          printf("\n\nYou selected choice #1.\n");
          showmen();
          break;
        case '2':
          printf("\n\nYou selected choice #2.\n");
          showmen();
          break;
        case '3':
          /* Choice #3 is to exit, so set a flag to end the main loop */
          terminate++;
          break;
        default:
          printf("\n\nInvalid selection.\n");
          showmen();
          break;
      }
    }
    /*****************************************************************************/
    /* This is where the processing should be placed that is to occur while      */
    /* awaiting user input. This program simply increments the variable "proc" to */
    /* demonstrate that processing is not halted while awaiting input from       */
    /* standard input.                                                           */
    /*****************************************************************************/
    proc++;
  }

  /* We have left the main loop. Clean up and exit. */
  printf("\n\nYou have selected exit\n");
  printf("The proc variable was incremented %d times.\n",proc);
  printf("** Note**\n");
  printf("The increments took place while you were interacting with the menu.\n");
  printf("This demonstrates processing was not halted while waiting user input.\n");
}
```

In the above example, the statement to read the user input from the standard input device (STDIN) is placed within a “while” loop, followed by the code that should not be suspended.

The **read** statement is placed in an "if" clause. The return value from **read** determines what action takes place. If there is valid input from STDIN, the appropriate “case” statement is executed. If no information is available, a –1 is returned and the “proc++” statement is executed.

Conclusion

The number of times that the *proc* variable is incremented in the example should be substantially higher than the sum of the number of times that options #1 and #2 are selected. This demonstrates that the *proc* variable is incremented during the time that the interactive menu is presented.

If the value of the *proc* variable displayed is equal to the sum of the number of times that options #1 and #2 are selected, the most likely cause is that a version of MiniMON29K prior to Release 3.0 is being used on either the host or target system (see “Ordering Information” below for how to order the current release).

Suggested Reference

For more information, see the *Host Interface (HIF) Specification, PID# 11539C.*

If You Need Assistance

Product support for the 29K Family processors is available from our Embedded Processor Division (EPD) Technical Support Hotlines located in the U.S. and in the U.K.

Assistance is available in the U.S. from 9:00 A.M. to 6:00 P.M. central time, Monday through Friday (except major holidays). In Europe assistance is available during U.K. business hours. Contact us at one of the following numbers.

To reach the U.S. hotline

From	Call
U.S.	1-800-2929-AMD
Japan	0031-11-1163
Any other location	+1-512-602-4118†

†Toll applies.

To reach the European hotline

From	Call
U.K.	(0)256-811101
France	0590-8621
Germany	0130-813875
Italy	1678-77224
Any other location	+44-(0)256-811101†

†Toll applies.

Ordering Information

To purchase the current release of the MiniMON29K product, contact your local sales office. Portions of the code for this product are available on the Advanced Micro Devices (AMD) Embedded Processor Division (EPD) Bulletin Board Service (BBS). To call the AMD EPD BBS, set your modem to dial **18002929263,,,,,,1**. Note that product support and documentation are not included when using code from the BBS.