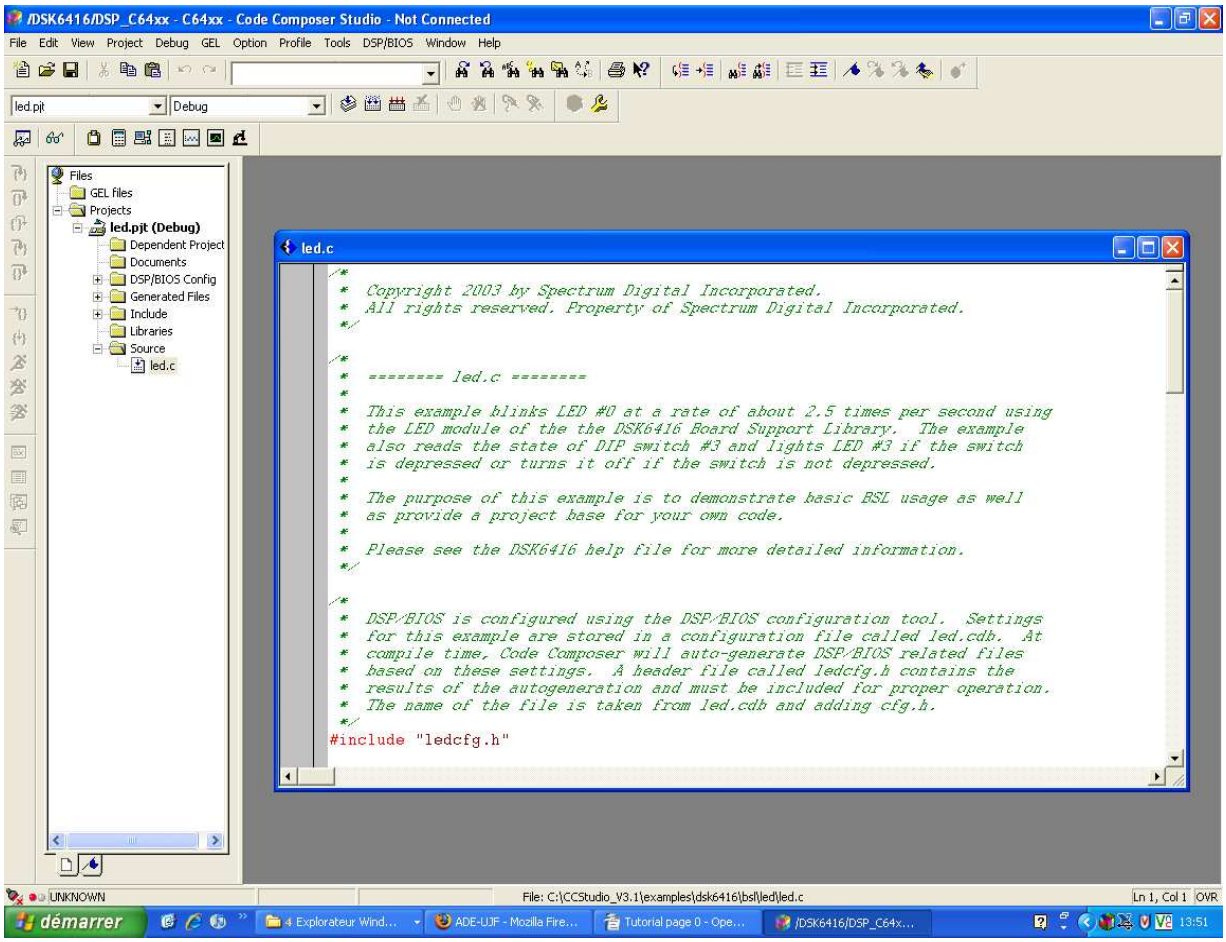
# Texas Instruments

Noyau Temps Réel

## *DSP BIOS TUTORIAL*

## *Kit DSP DSK 6416*




## *Laisser ce document sur place*

## C:\CCStudio_v3.1\tutorial\dsk6416\volume1\volume.c

```
#include <stdio.h>
#include "volume.h"
/* Global declarations */
int inp_buffer[BUFSIZE]; /* processing data buffers */
int out_buffer[BUFSIZE];
int gain = MINGAIN; /* volume control variable */
unsigned int processingLoad = BASELOAD; /* processing load */
struct PARMS str =
{
        2934, 9432, 213, 9432,
        &str
};
/* Functions */
extern void load(unsigned int loadValue);
static int processing(int *input, int *output);
static void dataIO(void);
/* ======== main ======== */
void main()
{
        int *input = &inp_buffer[0];
        int *output = &out_buffer[0];
        puts("volume example started\n");
        /* loop forever */
        while(TRUE)
        {
           /* Read using a Probe Point connected to a host file. */
           /* Write output to a graph connected through a probe-point. */
                dataIO();
                #ifdef FILEIO
                puts("begin processing");
                #endif
                /* apply gain */
                processing(input, output);
        }
}
/* ======== processing ======== *
* FUNCTION: apply signal processing transform to input signal.
* PARAMETERS: address of input and output buffers.
* RETURN VALUE: TRUE. */
static int processing(int *input, int *output)
{
        int size = BUFSIZE;
        while(size--){
        *output++ = *input++ * gain;
        }
        /* additional processing load */
        load(processingLoad);
        return(TRUE);
}

/* ======== dataIO ======== *
* FUNCTION: read input signal and write output signal.
* PARAMETERS: none.
* RETURN VALUE: none. */
static void dataIO()
{
        /* do data I/O */
        return;
}
```

- **testapp.c**

```
#include <stdio.h>
int main()
{
	int test_data[] = { 7, 13, 3, 25, 64, 15 };
	int max_value = maximumValue(test_data, 6);
	int min_value = minimumValue(test_data, 6);
	// call library functions
	printf( "The maximum value in the data is %d\n", max_value );
	printf( "The minimum value in the data is %d\n", min_value );
	return 0;
}
```

- **minimumvalue.c**

```
int minimumValue( int [],int);
// Requires: array_size equals the number of elements in the array, array_size > 0
// Returns: The minimum value contained in the integer array
int minimumValue( int values[], int array_size)
{
	int minimum_value;
	int i = 0;
	minimum_value = values[0];
	for (i = 0; i < array_size; i++)
{
	if (values[i] < minimum_value)
	minimum_value = values[i];
}
	return minimum_value;
}
```

- **maximumvalue.c**

```
int maximumValue( int [], int);
// Requires: array_size equals the number of elements in the array, array_size > 0
// Returns: The maximum value contained in the integer array
int maximumValue( int values[], int array_size)
{
	int maximum_value;
	int i = 0;
	maximum_value = values[0];
	for (i = 0; i < array_size; i++)
{
	if (values[i] > maximum_value)
	maximum_value = values[i];
}
	return maximum_value;
}
```

# Leçon 3: Les fonctions GEL page 106/548

- **C:\CCStudio_v3.1\tutorial\sim64xx\gelsolid**

```
menuitem "GEL Welcome Tool";
hotmenu Using_Local_Variables()
{
        int i;
        for (i = 0; i < 10; i++)
        GEL_TextOut("GEL is a solid tool.\n");
}
```

- **C:\CCStudio_v3.1\tutorial\sim64xx\gelsolid\welcome.c**

```
// ********************************************
// Sample DSP application used to showcase GEL
// automation
// ********************************************

#include <stdio.h>
void main()
{
            printf("Welcome to the World of DSP");
}
```

- **C:\CCStudio_v3.1\tutorial\sim64xx\gelsolid\mulidspwelcome.c**

```
// ********************************************
// Sample DSP application used to showcase GEL
// components controlling DSP code
// ********************************************
#include <stdio.h>
#define TRUE 1
volatile int counterValue = 0;
void main()
{
            while (TRUE) {
                        printf("Welcome to the World of DSP. %d\n", counterValue );
            }
}
```

- **C:\CCStudio_v3.1\tutorial\sim64xx\gelsolid\projectmanagement.gel**

```
// A GEL Function showcasing GELs ability
// to automate common tasksmenuitem "Project Management Tool";
hotmenu Run_Project()
{
        // Call standard GEL library function to load project
        GEL_ProjectLoad("C:\\source\\hellodsp.pjt");
        // Call standard GEL library function to build project
        GEL_ProjectBuild();
        // Call standard GEL library function to load the DSP program
        GEL_Load("C:\\Source\\Debug\\hellodsp.out");
        // Call standard GEL library function to run the project
        GEL_Run();
        // Indicate that the GEL script is finished
        GEL_TextOut("Finished Executing GEL Script!");

}
```

# Leçon 4: DSP BIOS Log_printf page 316/548

- **C:\CCStudio_V3.1\tutorial\dsk6416\hello1**

```
/*
 *  Copyright 2003 by Texas Instruments Incorporated.
 *  All rights reserved. Property of Texas Instruments Incorporated.
 *  Restricted rights to use, duplicate or disclose this code are
 *  granted through contract.
 *
 */
/* "@(#) DSP/BIOS 4.90.270 01-05-04 (bios,dsk6416-f06)" */
/*****************************************/
/*                                       */
/*    H E L L O . C                      */
/*                                       */
/*    Basic C standard I/O from main.    */
/*                                       */
/*                                       */
/*****************************************/
#include <stdio.h>
#include "hello.h"

#define BUFSIZE 30

struct PARMS str =
{
   2934,
   9432,
   213,
   9432,
   &str
};

/*
 *  ======== main ========
 */
void main()
{
#ifdef FILEIO
   int      i;
   char     scanStr[BUFSIZE];
   char     fileStr[BUFSIZE];
   size_t   readSize;
   FILE     *fptr;
#endif

   /* write a string to stdout */
   puts("hello world!\n");

#ifdef FILEIO
   /* clear char arrays */
   for (i = 0; i < BUFSIZE; i++) {
      scanStr[i] = 0         /* deliberate syntax error */
      fileStr[i] = 0;
   }
   /* read a string from stdin */
```

```
   scanf("%s", scanStr);

   /* open a file on the host and write char array */
   fptr = fopen("file.txt", "w");
   fprintf(fptr, "%s", scanStr);
   fclose(fptr);

   /* open a file on the host and read char array */
   fptr = fopen("file.txt", "r");
   fseek(fptr, 0L, SEEK_SET);
   readSize = fread(fileStr, sizeof(char), BUFSIZE, fptr);
   printf("Read a %d byte char array: %s \n", readSize, fileStr);
   fclose(fptr);
#endif
}
```

- **C:\CCStudio_V3.1\tutorial\dsk6416\hello2**

```
/*
 *  Copyright 2003 by Texas Instruments Incorporated.
 *  All rights reserved. Property of Texas Instruments Incorporated.
 *  Restricted rights to use, duplicate or disclose this code are
 *  granted through contract.
 *
 */
/* "@(#) DSP/BIOS 4.90.270 01-05-04 (bios,dsk6416-f06)" */
/*****************************************************************************/
/*                                                                           */
/*    H E L L O . C                                                          */
/*                                                                           */
/*    Basic LOG event operation from main.                                   */
/*                                                                           */
/*****************************************************************************/

#include <std.h>

#include <log.h>

#include "hellocfg.h"

/*
 *  ======== main ========
 */
Void main()
{
   LOG_printf(&trace, "hello world!");

   /* fall into DSP/BIOS idle loop */
   return;
}
```

- **C:\CCStudio_V3.1\tutorial\dsk6416\volume2**

```
/*
 *  Copyright 2003 by Texas Instruments Incorporated.
 *  All rights reserved. Property of Texas Instruments Incorporated.
 *  Restricted rights to use, duplicate or disclose this code are
 *  granted through contract.
 *
 */
/* "@(#) DSP/BIOS 4.90.270 01-05-04 (bios,dsk6416-f06)" */
/*************************************************************************/
/*                                                                       */
/*    V O L U M E . C                                                    */
/*                                                                       */
/*    Audio gain processing using CLK ISR as data source, and a software */
/*    interrupt for processing routine.                                  */
/*                                                                       */
/*************************************************************************/

#include <std.h>

#include <log.h>
#include <swi.h>

#include "volumecfg.h"

#include "volume.h"

/* Global declarations */
Int inp_buffer[BUFSIZE];       /* processing data buffers */
Int out_buffer[BUFSIZE];

Int gain = MINGAIN;            /* volume control variable */
Uns processingLoad = BASELOAD;  /* processing routine load value */

/* Functions */
extern Void load(Uns loadValue);

Int processing(Int *input, Int *output);
Void dataIO(Void);


/*
 * ======== main ========
 */
Void main()
{
   LOG_printf(&trace,"volume example started\n");

   /* fall into DSP/BIOS idle loop */
   return;
}

/*
```

```
 *  ======== processing ========
 *
 * FUNCTION: Called from processing_SWI to apply signal processing
 *          transform to input signal.
 *
 * PARAMETERS: address of input and output buffers.
 *
 * RETURN VALUE: TRUE.
 */
Int processing(Int *input, Int *output)
{
   Int size = BUFSIZE;

   while(size--){
      *output++ = *input++ * gain;
   }

   /* additional processing load */
   load(processingLoad);

   return(TRUE);
}

/*
 *  ======== dataIO ========
 *
 * FUNCTION: Called from timer ISR to fake a periodic hardware interrupt that
 *          reads in the input signal and outputs the processed signal.
 *
 * PARAMETERS: none.
 *
 * RETURN VALUE: none.
 */
Void dataIO()
{
   /* do data I/O */

   SWI_dec(&processing_SWI);   /* post processing_SWI software interrupt */
}
```

- **C:\CCStudio_V3.1\tutorial\dsk6416\volume3**

```
/*
 *  Copyright 2003 by Texas Instruments Incorporated.
 *  All rights reserved. Property of Texas Instruments Incorporated.
 *  Restricted rights to use, duplicate or disclose this code are
 *  granted through contract.
 *
 */
/* "@(#) DSP/BIOS 4.90.270 01-05-04 (bios,dsk6416-f06)" */
/*************************************************************************/
/*                                                                       */
/*    V O L U M E . C                                                    */
/*                                                                       */
/*    Audio gain processing using CLK ISR as data source, and a software */
/*    interrupt for processing routine.                                  */
/*                                                                       */
/*************************************************************************/

#include <std.h>

#include <log.h>
#include <swi.h>
#include <clk.h>
#include <sts.h>
#include <trc.h>

#include "volume.h"

#include "volumecfg.h"

/* Global declarations */
Int inp_buffer[BUFSIZE];       /* processing data buffers */
Int out_buffer[BUFSIZE];

Int gain = MINGAIN;            /* volume control variable */
Uns processingLoad = BASELOAD;  /* processing routine load value */

/* Functions */
extern Void load(Uns loadValue);

Int processing(Int *input, Int *output);
Void dataIO(Void);

/*
 * ======== main ========
 */
Void main()
{
   LOG_printf(&trace,"volume example started\n");

   /* fall into DSP/BIOS idle loop */
   return;
}
```

```
/*
 *  ======== processing ========
 *
 * FUNCTION: Called from processing_SWI to apply signal processing
 *          transform to input signal.
 *
 * PARAMETERS: address of input and output buffers.
 *
 * RETURN VALUE: TRUE.
 */
Int processing(Int *input, Int *output)
{
   Int size = BUFSIZE;

   while(size--){
      *output++ = *input++ * gain;
   }

   /* enable performance instrumentation only if TRC_USER0 is set */
   if (TRC_query(TRC_USER0) == 0) {
      STS_set(&processingLoad_STS, CLK_gethtime());
   }
   /* additional processing load */
   load(processingLoad);
   if (TRC_query(TRC_USER0) == 0) {
      STS_delta(&processingLoad_STS, CLK_gethtime());
   }

   return(TRUE);
}

/*
 *  ======== dataIO ========
 *
 * FUNCTION: Called from timer ISR to fake a periodic hardware interrupt that
 *          reads in the input signal and outputs the processed signal.
 *
 * PARAMETERS: none.
 *
 * RETURN VALUE: none.
 */
Void dataIO()
{
   /* do data I/O */

   SWI_dec(&processing_SWI);   /* post processing_SWI software interrupt */
}
```