APPENDIX E.2. Using the Windows Control Application (C8X_Control)

E.2.1 Overview

To control the DSP from a Windows application, programs must be created for both the host computer and the DSP. The sample host computer Windows application is written in Microsoft Visual C++ 2010. It is assumed that the reader is familiar with the Visual Studio 2010 environment, so the details of using it are not described here. The interface between the host computer and the DSP board can be serial or USB. The software details of this interface are encapsulated in a dynamic link library file (C8X_CONTROL_DLLxx.DLL) that is included with the host computer program. To transfer data to and from the DSP, the host computer must know the addresses where the data is stored on the DSP. To simplify this process, the DLL contains a function that will return the address of a program variable, assuming the variable name has been stored in the symbol table of the executable file. The host computer application can be built for a 32-bit or 64-bit version of Windows. There are two versions of the DLL, one for each Windows version. Complete sample applications of the DSP code for the OMAP-L138 and TMS320C6713 DSK are also included.

The basic Windows control application implements a simple audio talk-through with a gain control. Highlights of the software are shown below. At least a cursory reading of all of the supplied source code is advised for a better understanding of the interface structure.

E.2.2 DSP Hardware Requirements

The sample application supports both the OMAP-L138 and the TMS320C6713 DSK.

To use the OMAP-L138, the board must have been reflashed with the winDSK8 kernel. Details of this process are described in the text. Ensure that the host application's baud rate is set to match the baud rate set by the DIP switches on the OMAP-L138 board. Note that with some OMAP-L138 boards and some USB-to-serial adapters, the 921.6k baud rate may not work.

The TMS320C6713 DSK requires the installation of the Educational DSP, LLC Host Port Interface daughtercard. Communications through the USB and serial interfaces are supported. If using the serial interface, set the baud rate to WINDSK8_BAUD_115200. If using the USB interface, set the baud rate to WINDSK8_BAUD_921600.

If you have difficulty getting the connection to work between the host computer and the DSP board, use the winDSK8 program to help troubleshoot the problem.

E.2.3 The Host Software

The host software is written to include proper initialization software for all supported DSPs. Detailed documentation of the host software DLL function calls (in C8X_CONTROL_DLLxx.DLL) is available later in this document.

The CC8X_CONTROLDlg::OnInitDialog() function is executed when the application's dialog window is created. The C8X_CONTROL_InitializeHostInterface() function is called to initialize the host interface software and select the correct COM port and baud rate. The OnInitDialog() function also configures the application's Gain slider control, and sets the DspIsRunning variable to false indicating that there is no program running on the DSP. After this, the PC application will wait for the user to click the Reset or Run button.

If the Reset button is clicked, the CC8X_ControlDlg::OnBnClickedResetdspbtn() function is executed and causes the DSP to be reset. The DspIsRunning variable is also set to false, indicating that there is no program running on the DSP.

If the Run button is clicked, the CC8X_ControlDlg::OnBnClickedLoadandrunbtn() function is executed, resetting the DSP, loading the program onto it, and starting it running. After the program is loaded, the program gets the address of the Gain variable, which is available in the symbol table of the DSP program. The DspIsRunning variable is also set to true, indicating that there is now a program running on the DSP.

As the gain slider is moved, the CC8X_ControlDlg::OnNMReleasedcaptureSlider1() function is executed when the left button is released. If the DSP program is running, this function gets the current position of the gain slider and maps that position to a floating-point value on a logarithmic scale. This value is then written to the address of the Gain variable on the DSP, controlling the audio gain.

E.2.4 The DSP Software

The DSP software is simply the basic talk-through program, with the addition of a gain variable that is applied to the audio data. Since the Gain variable is a global variable defined in ISRs.c, its name and address will be stored in the executable file. Note that since the variable name stored is at the assembly language level, the name in the file will be the C language name

prepended with an underscore.

The incoming audio data is stored in the variable CodecDataIn, with stereo codec data stored as two 16-bit samples in the 32-bit variable. The outgoing data is obtained by multiplying the individual channels of CodecDataIn by the Gain variable and storing the results in CodecDataOut. CodecDataOut is then sent to the codec.

When rebuilding the CCS projects after customizing the applications for your use, you should use a Release build.

E.2.5 Building and Running the Application

A CCS workspace with complete sample projects for the OMAP-L138 and TMS320C6713 DSK is included under the DSP8_CCS4 directory. These projects already include a Release version of the applications, so they do not need to be rebuilt to run the sample programs. Open the Visual C++ 2010 project. Make any needed changes for the DSP and COM port being used. Note that the only changes you should need to make are as follows;

- In CC8X_ControlDlg::OnInitDialog(), edit the call to C8X_CONTROL_InitializeHostInterface() to use the correct COM port and the correct baud rate.
- In CC8X_ControlDlg:: OnBnClickedLoadandrunbtn() (), edit the call to C8X_CONTROL_ResetAndRunCOFF to use the correct executable file's pathname for your DSP.

Then, select the applicable build configuration (win32 or x64), build and run the program. You will now be able to control the audio gain of the DSP from the host computer.

E.2.6 C8X_CONTROL_DLLxx Function Documentation

The 32-bit and 64-bit versions of C8X_CONTROL_DLLxx.DLL both export the same set of functions, so their interface is described in a common header file, C8X_CONTROL_DLL.h. There are two header files included by C8X_CONTROL_DLL.h.

winDSK8_stdtypes_win.h

Defines the data types Int8, Uint8, etc., as used throughout the text.

DSP8_CCS4\\windsk8_enums.h

Enumerates COM port numbers COM1-COM256, and the baud rates used with the DSP board interfaces.

A detailed description of each DLL function is given in the following pages.

C8X_CONTROL_InitializeHostInterface

bool C8X_CONTROL_InitializeHostInterface(Uint32 port, ComSpeeds speed);

Return Value

Returns false on error, indicating a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

port

Select the COM port number to be used for communications. COM1 through COM256 are defined in windsk8_enums.h.

speed

Sets the baud rate to use when communicating with the DSP board. Allowable values are defined in windsk8_enums.h. When using the TMS320C6713 DSK with HPI daughtercard, use WINDSK8_BAUD_115200 for serial communication and WINDSK8_BAUD_921600 for USB communication. When using the OMAP-L138, choose the baud rate to match the DIP switch settings on the OMAP-L138 board.

Remarks

This function MUST be called before any other DLL function.

C8X_CONTROL_getVersionString

char * C8X_CONTROL_getVersionString();

Return Value

Returns a pointer to a null-terminated character string reflecting the winDSK8 kernel version that the DLL was built from, and whether it is a 32-bit or 64-bit implementation.

Parameters

none

Remarks

None.

C8X_CONTROL_getBoardVersion

Uint32 C8X_CONTROL_getBoardVersion();

Return Value

Returns a 32-bit value containing the DSP board version information.

Parameters

none

Remarks

None.

C8X_CONTROL_ResetDsp

Int32 C8X_CONTROL_ResetDsp(void);

Return Value

Returns 0 if the reset command fails, non-zero on success.

Parameters

none

Remarks

Causes a reset of the DSP board.

C8X_CONTROL_Run

bool C8X_CONTROL_Run();

Return Value

Returns false on error, indicating a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

none

Remarks

This function is only to be called after using C8X_CONTROL_LoadCOFF to load a program to the DSP. This can be done to allow some additional configuration of the DSP before allowing the loaded program to run. However, in general, the C8X_CONTROL_ResetAndRunCOFF function should be used to load and run a DSP program.

C8X_CONTROL_LoadCOFF

bool C8X_CONTROL_LoadCOFF(const Int8 *filename);

Return Value

Returns false on error, indicating the file could not be loaded or there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

filename

ASCIIZ string with the pathname of the file to load.

Remarks

Loads a file onto the board, but does not start it running. The DSP board must be reset before this function is called. The C8X_CONTROL_Run must be called to start the program running. The C8X_CONTROL_LoadCOFF function should only be used when there is a specific reason to not start the program running right away. In general, the C8X_CONTROL_ResetAndRunCOFF function should be used to load and run a DSP program.

C8X_CONTROL_ResetAndRunCOFF

bool C8X_CONTROL_ResetAndRunCOFF(const Int8 *filename);

Return Value

Returns false on error, indicating the file could not be loaded or there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

filename

ASCIIZ string with the pathname of the file to load.

Remarks

Resets the DSP board, then loads the specified program onto the DSP and starts the program running.

C8X_CONTROL_Write

bool C8X_CONTROL_Write(Uint32 address, Uint32 count, Uint32 *pdata);

Return Value

Returns false on error, indicating there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

address

The starting address to write data to.

count

The number of 32-bit words to write.

pdata

Pointer to an array containing the data to be written to the DSP board.

Remarks

Writes *count* sequential words of data starting at the specified address.

C8X_CONTROL_WriteFloat

bool C8X_CONTROL_WriteFloat(Uint32 address, Uint32 count, float *pdata);

Return Value

Returns false on error, indicating there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

address

The starting address to write data to.

count

The number of 32-bit floating-point data words to write.

pdata

Pointer to an array containing the floating-point data to be written to the DSP board.

Remarks

Writes *count* sequential words of floating-point data starting at the specified address.

C8X_CONTROL_Read

bool C8X_CONTROL_Read(Uint32 address, Uint32 count, Uint32 *pdata);

Return Value

Returns false on error, indicating there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

address

The starting address to read data from.

count

The number of 32-bit words to read.

pdata

Pointer to an array of adequate size to store the data read from the DSP board.

Remarks

Reads *count* sequential words of data starting from the specified address. The programmer must ensure that pdata points to an array of adequate size.

C8X_CONTROL_ReadFloat

bool C8X_CONTROL_ReadFloat(Uint32 address, Uint32 count, float *pdata);

Return Value

Returns false on error, indicating there was a problem communicating with the DSP board.

Otherwise, returns true.

Parameters

address

The starting address to read data from.

count

The number of 32-bit floating-point data words to read.

pdata

Pointer to an array of adequate size to store the data read from the DSP board.

Remarks

Reads *count* sequential words of floating-point data starting from the specified address. The programmer must ensure that pdata points to an array of adequate size.

C8X_CONTROL_GetSymbolValue

bool C8X_CONTROL_GetSymbolValue(const char *symbol, Uint32 *address);

Return Value

Returns false if the symbol was not located in the symbol table. Otherwise, returns true.

Parameters

symbol

Pointer to a string containing the name of the symbol. Note that C language symbols are prepended with an underscore in the executable file.

address

Pointer to a variable to store the symbols address in.

Remarks

None.